CARL VON OSSIETZKY universität OLDENBURG

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Abteilung Assistenzsysteme und Medizintechnik

Abschlussbericht

Projektgruppe Mobilitäts-Assessments mit körpernahen Sensoren für zuhause SoSe17 - WiSe17/18

Themensteller: Prof. Dr.-Ing. Andreas Hein
Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autoren: Beatrice Coldewey, Eugen Lange, Marius Leyh, Jan-Frederik Scharnowski, Maren Steinkamp, Alexander Thomas, Felix Van Der Ahe, Patrick Warszewik, Marlon Willms

16. April 2018

Zusammenfassung

Der folgende Abschlussbericht der Projektgruppe PGMAMKSFZ legt alle wichtigen Informationen und Dokumentation zum Thema *Mobilitäts-Assessments mit körpernahen Sensoren für zuhause* dar. Aufbauend auf die letzte Projektgruppe wurde die MAMKS-Software um einige Funktionen erweitert und optimiert. Es wurden unter anderem neue Funktionen zum Labelbearbeiten hinzugefügt, ein Statistik Modul in Form eines B-Charts, sowie die Weitergabe der Berechnung von Klassifikationsalgorithmen an MATLAB und Python. Ein Großteil der Arbeit bestand in der Datenerhebung. So wurden die Bewegungsdaten bei den Probanden im häuslichen Umfeld aufgenommen und darauf basierend verschiedene Klassifikationsalgorithmen antrainiert. Zu den Algorithmen zählen Bagging Trees, Boosted Decision Trees, K-Nearest Neighbor, Lineare und Quadratische Diskriminanzanalyse, sowie Faltende Neuronale Netze. Neben der Klassfikationsberechnung wurde außerdem eine Applikation zur Aktivitätstracking in Form einer Android App entwickelt. Diese wurde auf Tablets installiert und sollte bei der Studiendurchführung bei den Probanden zu Hause zeitgenau die Aktivitäten dokumentieren. Des Weiteren wurden noch zwei Ansätze zum Nachbearbeiten von Klassifikationen verfolgt.

Weiterhin werden in diesem Bericht, neben den Implementierungen auch Anforderungen zur Entscheidungsfindung sowie Konzepte zur Modelloptimierung und Labelvorbereitung und Labelnachbereitung dokumentiert. Zusätzlich werden im Kapitel Biomechanik auf die physiologischen Bewegungsabläufen am Becken eingegangen. Dies dient als Grundlage für die Labelnachbereitung. Im Anhang befinden sich die Ausarbeitungen von den jeweiligen Gruppenmitgliedern, die sich jeweils auf ein Thema im Bezug auf die Projektgruppe beziehen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungen

AEQUIPA Physical activity and health equity: primary prevention for healthy ageing

API Application Programming Interface

CLI Command Line Interface

CNN Convolutional Neural Network

CPU Central Processing Unit

FP False Positive

FN False Negative

GPU Graphics Processing Unit

HOO HyperparameterOptimizationOption

HPC High-Performance Computing

JAXB Java Architecture for XML Binding

KNN K-Nearest Neighbor

PGMAMKS Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren“

PGMAMKSFZ Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren für zuhause“

QDA Quadratische Diskriminanzanalyse

ROC Receiver Operating Characteristic

SHID SeniorHome-ID

TN True Negative

TP True Positive

VERSA Vorhersage zum Erhalt der Selbstständigkeit im Alter

XML Extensible Markup Language

XSD XML Schema Definition

Glossar

Bar-Chart Diagramm Säulendiagramm

Folds Verfahren zur Validierung von Klassifikatoren

FXML XML basierte Sprache in Java-FX zur Spezifizierung von Benutzeroberflächen.

Holdout-Validation Sonderfall der Kreuzvalidierung, bei der gesamte Datensatz zufällig in zwei Datensätze aufgeteilt wird. Die zwei Teilmengen sind Trainings- und Validierungsdaten, wobei die Validierungsdaten meist weniger sind[8].

IMU-Datensätze Sensordatensätze

jar Ausführbare Programmdatei, die in JAVA geschrieben wurde

Konfusionsmatrix Auch Wahrheitsmatrix genannt. Dient zur Beurteilung eines Klassifikators.

kontralateral auf der entgegengesetzten Körperhälfte gelegen

Kreuzvalidierung Grundsätzlich wird bei der Kreuzvalidierung ein Datensatz in mehrere Teile aufgeteilt. Auf Basis eines Teils des Datensatzes wird ein statistisches Modell abgeleitet. Das Modell wird auf den Rest des Datensatzes angewendet und die Abweichung zu den tatsächlichen Werten untersucht. Das Verfahren dient also in erster Linie der Überprüfung der Prognosegüte eines Modells. Es wird unterschieden zwischen der einfachen Kreuzvalidierung, der stratifizierten Kreuzvalidierung und der Leave-One-Out-Kreuzvalidierung.

MAMKS-Software von der PGMAMKS im Zuge der Projektarbeit erstellte Software zu Verarbeitung von Sensordaten des Humotion-Sensorgürtels

MouseDraggedEvent Mausbewegung, bei der die linke Maustaste gedrückt gehalten wird, während die Maus bewegt wird

MouseEventListener Funktion, die auf die Mausbewegungen und Mausklicke des Nutzers achtet.

Path-Umgebungsvariable Umgebungsvariable in Windows.

Präzision Anteil der relevanten Treffer (True Positives) an allen relevanten Treffen (True Positives + False Positives) eines Klassifikators.

Predictor Im Kontext des maschinellen Lernens sind Prädiktoren Eingabedaten oder Variablen, die durch eine empirische Beziehung, die üblicherweise durch die Daten bestimmt wird, auf die Zielvariablen abgebildet werden.

Recall Trefferquote. Anteil der relevanten Treffer (True Positives) an allen relevanten Profilen (True Positives + False Negatives) eines Klassifikators.

sagitto-frontale Achse Körperachse, von Kopf zu Fuß verlaufend

sagitto-transversale Achse Körperachse, in Blickrichtung

ScrollPane Ansicht mit Scroll-Möglichkeit.

TabPane Mithilfe des TabPane ist es möglich mithilfe der Tabs/Reiter zwischen mehreren Panes/Ansichten zu in einer Anwendung zu wechseln.

TableView Java-FX Objekt zur Darstellung von Tabellen.

Transversalebene Körperebene parallel zum Boden

Workspace Ordner, in dem die vom Programm genutzten Dateien gespeichert sind.

1. Einleitung

Im Folgenden wird ein allgemeiner Überblick über die Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren für zuhause“ (PGMAMKSFZ) und ihre Tätigkeit im Rahmen des Masterstudiums in der Abteilung Assistenzsysteme und Medizintechnik der Carl von Ossietzky Universität Oldenburg gegeben.
Zu Beginn wird auf das Projektgruppen-Modul und seine Ziele im Allgemeinen eingegangen. Daraufhin erfolgt eine einleitende Motivation sowie eine Beschreibung des Problems bezüglich des Themas der Projektgruppe. Es folgen die Ziele, die sich die Projektgruppe gesetzt hat, sowie die Bewertungskriterien, anhand derer die Erfüllung der Ziele gemessen werden kann. Abschließend wird ein Überblick über die einzelnen Kapitel des Abschlussberichts gegeben.

1.1. Allgemein zur Projektgruppe

Die Projektgruppe ist ein Pflichtmodul in den Master-Studiengängen Informatik, Wirtschaftsinformatik und Eingebettete Systeme und Mikrorobotik an der Carl von Ossietzky Universität Oldenburg. Diese besteht in der Regel zwischen sechs und zwölf Teilnehmerinnen und Teilnehmern, die innerhalb von zwölf Monaten zusammen ein Projekt im Umfang von vier Modulen durchführen sollen, was einen Arbeitsaufwand von 24 Kreditpunkten entspricht.
Dabei besteht das Ziel, anhand eines gegebenen Problems, dieses durch eine vollständige Problemanalyse, bis hin zur Realisierung des Systems durchzuführen. Neben dem Projekt sollen aber auch andere berufstypische Einblicke gegeben werden, wie das Arbeiten im Team, der Übernahme von Verantwortung, sowie persönliche Fähigkeiten, wie das zielorientierte Argumentieren. Weiterhin werden von jedem Teilnehmer eine Seminar-

arbeit und eine zugehörige Präsentation eines projektnahen Seminarthemas gefordert. Zusätzlich muss die Gruppe als Ganzes eine umfangreiche Dokumentation über das Projekt erstellen und es in einer Abschlusspräsentation vorstellen.
Die Projektgruppe PGMAMKSFZ setzt sich aus folgenden neun Studentinnen und Studenten zusammen: Beatrice Coldewey, Eugen Lange, Marius Leyh, Jan-Frederik Scharnowski, Maren Steinkamp, Alexander Thomas, Felix Van Der Ahe, Patrick Warszewik und Marlon Willms.
Betreuer und Gutachter der hier vorgestellten Projektgruppe sind Prof. Dr.-Ing. Andreas Hein, Dr. rer. nat. Sebastian Fudickar und M.Sc. Sandra Hellmers.

1.2. Motivation

Die Versa-Studie ist ein Teilprojekt des vom Bundesministerium für Bildung und Forschung (BMBF) geförderten Verbundprojektes AEQUIPA. Ziel dieser Studie ist die Entwicklung von sensorgestützten Verfahren zur Bewegungsanalyse, um zukünftig auftretende Probleme bezüglich der Mobilität frühzeitig zu erkennen. Im Rahmen der VERSA-Studie wurden in Zusammenarbeit mit der Abteilung Assistenzsysteme und Medizintechnik der Carl von Ossietzky Universität Oldenburg geriatrische Assessments mit circa 250 Senioren zwischen 70 und 89 Jahren in mehreren Iterationen durchgeführt. Die dabei von den Probanden durchgeführten Bewegungen wurden mithilfe eines Humotion-Sensorgürtels aufgenommen. Dieser zeichnet Rohdaten der Sensoren Accelerometer, Barometer, Gyrometer und Magnetometer auf. Im Anschluss an das Assessment tragen die Senioren den Gürtel für eine Woche im häuslichen Umfeld, um weitere Daten im natürlichen Bewegungsraum zu sammeln.[11]
In Zukunft könnte ein solcher Gürtel in den täglichen Alltag über einen längeren Zeitraum integriert werden. Durch die große Menge an gesammelten Daten und ihre Auswertung könnten dann genauere Aussagen über die Veränderungen der Mobilität des Patienten angetroffen werden. So können frühzeitig präventive Maßnahmen eingeleitet

werden, um einer Verschlechterung entgegenzuwirken oder zu verlangsamen.
Die Projektgruppe PGMAMKS hat im vergangenen Jahr bei der Verarbeitung der aus den Assessments gewonnenen Daten angesetzt und Ihre Ergebnisse präsentiert.
Die diesjährige PGMAMKSFZ verfolgt aufbauend auf den Ergebnissen der alten Projektgruppe einen ähnlichen Ansatz mit den im häuslichen Umfeld gewonnenen Daten.

1.3. Problembeschreibung

Im Zuge der Versa-Studie wurden große Mengen an Bewegungsdaten sowohl während der Assessments als auch im häuslichen Umfeld der Probanden aufgenommen. Die letzte Projektgruppe hat bereits eine Software entwickelt, die es ermöglicht, die gesammelten Bewegungsdaten des Humotion Sensorgürtels einzulesen, zu visualisieren und im Hinblick auf bestimmte Aktivitäten zu analysieren. Allerdings wurde die für die Analyse verwendeten Daten nicht im natürlichen Umfeld bei den Senioren zu Hause aufgenommen. Die Algorithmen wurden lediglich auf der Basis der Assessmentdaten entwickelt und evaluiert. Die Übertragbarkeit auf Daten, die im häuslichen Umfeld aufgenommen wurden, ist nicht geklärt.

1.4. Zielsetzung

Die Arbeit der Projektgruppe PGMAMKSFZ setzt auf den Ergebnissen der Projektgruppe PGMAMKS auf. Ein besonderes Augenmerk liegt dabei auf den Bewegungsdaten, die im häuslichen Umfeld aufgenommen wurden und nicht von der PGMAMKS betrachtet wurden. Die Projektgruppe PGMAMKSFZ hat sich zwei grundlegende Ziele gesetzt, die im Rahmen ihrer Arbeit erreicht werden sollen:
Zum Einen soll die Anwendbarkeit bzw. die Klassifikationsergebnisse des „MAMKS-Modells“ auf den Daten aus dem häuslichen Umfeld überprüft werden. Zum Anderen

sollen weitere Klassifikationsalgorithmen entwickelt werden, um unter Einbeziehung der Bewegungsdaten aus dem häuslichen Bereich (wenn möglich) bessere Klassifikationsergebnisse zu erreichen und weitere Aktivitäten zu erkennen.
Zur Erfüllung dieser Ziele soll neben zusätzlich entwickelten Softwarekomponenten, z. B. einer Applikation zum generieren vom Label mit Zeitstempel, die von der PGMAMKS entwickelte Software um notwendigen Anpassungen erweitert werden und Schnittstellen für ausgelagerte Funktionalitäten, z. B. Maschinelles Lernen mittels Matlab und Python, geschaffen werden.
Analog zum Entwickeltem Standard zum Labeln der Assessmentdaten sollen zudem die Bewegungsdaten aus dem häuslichen Bereich nach einem eigens entwickeltem Standard gelabelt werden.

1.5. Bewertungskriterien

Um die Erfüllung der Ziele zu gewährleisten, wurden zudem folgend Bewertungskriterien festgelegt:
Die Überprüfung des MAMKS-Modells unter Anwendung der Bewegungsdaten aus dem häuslichen Bereich liefert ein quantitatives Maße zur Beurteilung des Klassifikators (z. B. F1-Score), der einen Vergleich zu den Ergebnissen der vorherigen Projektgruppe ermöglicht. Ebenso sind die neu entwickelten Klassifikatoren bezüglich ihrer Qualität anhand quantitativer Maße zu überprüfen. Zudem sollte mindestens ein gleichwertiges oder besseres Klassifikationsergebnis erreicht werden.
Die neu entwickelten Softwarekomponenten sind auf ihre korrekte Funktionalität zu überprüfen. Soweit wie möglich sind alle benötigten Erweiterungen in die MAMKS-Software integriert worden und bei Verwendung externen Softwarekomponenten entsprechende Schnittstellen geschaffen worden.
Der entwickelte Labelstandard ist für eine Anwendung durch Dritte ausreichend zu dokumentieren und wurde erfolgreich im Zuge der Arbeit der Projektgruppe eingesetzt.

1.6. Aufbau des Projektabschlussberichtes

Im folgenden Abschnitt werden die einzelnen Kapitel im Projektabschlussbericht vorgestellt und näher erläutert. Das Kapitel Projektorganisation beschreibt den wesentlichen Aufbau dieser Arbeit, als auch die geführten Tagebücher. Einleitend in die Inhalte der Projektgruppenarbeit werden die Anforderungen beschrieben, resultierend aus der Befragung der Stakeholder, sowie Anwendungsfälle zur Implementierung. Im Kapitel Konzept werden auf die Optimierungen und Vorgehensweise der weiteren Implementierungen mit Hilfe von MATLAB und Python eingegangen, insbesondere zur Labelvorbereitung, manuellen Beschriftung, Aktivitätstracking und Klassifizierung der Labelnachbereitung. Im Abschnitt Biomechanik werden die einzelnen Aktivitäten genauer betrachtet und beschrieben, unter anderem das Stehen/ Sitzen, Treppensteigen, Aufstehen/ Hinsetzen, sowie das Gehen. Im Kapitel Studiendurchführung werden zunächst die Ziele dieser Studie definiert, eingeschlossen die Pilot-Studie, die zuvor durchgeführt wurde. Anschließend wird der Aufbau der Studie vorgestellt, sowie die gewonnenen Erkenntnisse. Im Abschnitt Implementierung wird die eigen entwickelte Android App als Aktivitätstracker näher erläutert, sowie Erweiterungen zur MAMKS-Software. Das Kapitel Optimierungen beschreibt die verschiedenen Algorithmen zur Verbesserung der Klassifikationserkennung der Bewegungsdaten. Jeder Algorithmus wird zunächst grob vorgestellt und einzelne Parameteroptimierungen in der MATLAB und Python Umgebung erläutert. Am Ende werden die Ergebnisse der Optimierungen für jeden Algorithmus aufgelistet und einen Ausblick für die weiteren Möglichkeiten erörtert. Im vorletzten Kapitel Evaluation wird auf die Validierung der Android App, als auch die der Ergebnisse der optimierten Modelle. Zum Schluss erfolgt ein kurzes Fazit, sowie Ausblick über mögliche Verbesserungen und Erweiterungen der Klassifizierung von Bewegungsdaten. Im Anhang befinden sich jeweils die Ausarbeitungen von den jeweiligen Gruppenmitgliedern, die sich auf ein Thema in Bezug auf die Projektgruppe beziehen.

2. Projektorganisation

In diesem Kapitel werden die Projektarbeit und die Organisation beschrieben. Darin handelt es sich um den konkreten Aufbau dieses Projekts und welche Personen und Rollen darin vertreten waren.

2.1. Projektaufbau

Nachfolgend werden die beteiligten Akteure und ihre Rolle innerhalb des Projekts genannt. Die Auftragsgeber stellen die Professoren und Dozenten um Andreas Hein, Sebastian Fudickar und Sandra Hellmers dar. Von ihnen wurde das übergeordnete Ziel dieses Projekts vorgegeben. Als Projektbetreuer standen uns Sebastian Fudickar und Sandra Hellmers während der gesamten Projektlaufzeit als Berater zur Verfügung. Die Studierenden organisierten innerhalb des Projektteams anhand ihrer Interessen selbst, worin sich jedes Mitglied zugehörig fühlte und gerne arbeiten wollte. Die Projektziele gaben dazu im groben die Möglichkeit sich in Bereichen der Daten Analyse von Gürteldaten zu beschäftigen oder im Bereich der Softwareentwicklung Erweiterungen der MAMKS Software zu entwickeln. Bereits früh kristallisierte sich hierbei ein Verhältnis von etwa zwei Dritteln für die Daten Analyse und ein drittel für die Softwareentwicklung heraus. Obwohl diese Einteilung weder von den Projektbetreuern oder dem Team selber verbindlich gesetzt wurde, bestand diese Aufteilung vom Beginn bis Ende der Projektlaufzeit. Zu den bereits vorherigen groben Einteilungen, wurden anfangs weitere Rollen an einzelne Mitglieder vergeben, um die Kompetenzen jedes Mitgliedes zu erweitern und um die uns zu Verfügung stehenden Ressourcen optimal einsetzen zu können. Die Rollen sollten dabei als Expertise eines Projektmitglieds verstanden werden, auf dessen Wissen die restlichen Mitglieder setzen könnten und bei Problemen in diesem Bereich diejenige

Person	**Rolle**	**Aufgabenbereich**
Beatrice	Dokumentationsbeauftragte	Daten Analystin
Eugen	Projektmanagement	Daten Analyst, Softwareentwickler
Marius	Confluencebeauftragter, Projektkoordinator	Daten Analyst
Frederick	Anforderungsmanagement	Daten Analyst
Maren	Raum und Finanzmanagement	Softwareentwicklerin
Alexander	Qualitätsmanagement	Daten Analyst
Felix	Anforderungsmanagement	Daten Analyst
Patrick	Scrum Master	Softwareentwickler
Marlon	HPC und Anforderungsmanagement	Daten Analyst

Tabelle 2.1.: Projektorganisation: Zuordnung Personen, Rollen und Aufgabenbereiche

Person Abhilfe in der Form einer guten und schnellen Lösung beitragen sollte. Die folgende Tabelle bildet ab, in welchen Abteilungen jedes Projektmitglieder arbeitete, und mit welcher Rolle jede Person während der Projektlaufzeit inne hatte.

Die einzelnen Rollenbeschreibungen und damit verbunden Aufgabenbereiche wurden vom Projektteam unmittelbar nach Ernennung der einzelnen Personen im Confluence festgehalten. Im folgenden werden diese Rollendefinitionen wiedergegeben.

- Dokumentationsbeauftragte Der Dokumentationsbeauftragte / LaTeX-Beauftragte beschäftigt sich der adäquaten Dokumentation des Projekts in LaTeX. Die damit verbundenen Aufgaben beinhalten das Erstellen von Vorlagen und deren eventuelle Problembeseitigung. Dazu steht die Person zur Beantwortung aller Fragen zu diesem Thema bereit und überwacht dazu die allgemeine Dokumententation und weißt das Team auf Fehlentwicklungen hin. Als Klare Aufgabentrennung steht die Produzieren und Korrektur von Dokumentationsin-

halten.

- Projektmanagement Das Projektmanagement ist für die organisatorische Leitung des Projekts zuständig und übernimmt das gesamte Projektmanagement. Das Projektmanagement muss auch dafür sorgen, dass das Projekt einen konkreten Ablauf folgt, mit den Zielen eine Transparenz sowie eine effektive und effiziente Gliederung von Aufgaben und deren Einhaltung zu schaffen. Damit sollten mögliche Vermeidungen von Unklarheiten erreicht werden mit einer funktionierenden Vermittlung zwischen allen Beteiligten.
  Die Aufgaben und Verantwortung liegen in der Projektdefinition, indem gemeinsam realistische Projektziele definiert werden. Die Projektorganisation sollen für die klare Rollendefinition und einer optimalen Kommunikationsstrukur bieten. Die Projektplanung umfasst die Erstellung von Netzplänen und die Definition von Meilensteinen die in Arbeitspaketen und Tasks erstellt werden. Als letzte Komponente schließt sich dem des Projektcontrolling an, deren Schnittstelle die Projektkoordination bildet und das Einhalten von Terminen beinhaltet und deren zugehörige Kommunikation zum Projektteam.

- Confluencebeauftragter Der Confluencebeautragter beschäftigt sich mit der Administration und der Struktur der beiden Plattformen Confluence und JIRA. Des weiteren ist er Ansprechpartner bei Fragen und Änderungen an den genannten Systemen. Dazu gehört auch die Anpassung der Systeme an den Ansprüchen der Projektgruppe.

- Projektkoordinator Der Projektkoordinator hat immer einen Überblick über alle aktuellen Aufgaben, welche in der Projektgruppe gemacht werden. Er überwacht die Termine der Aufgaben und sorgt für eine rechtzeitige Erledigung dieser. Ebenfalls kann er als Vermittler dienen, insofern Probleme bzgl. Aufgaben auftauchen. Dazu gehören auch Aufgaben der aktuellen Überwachung des Sprints und der Vermittlung von Problemen mir Aufgaben.

- Anforderungsmanagement Die Verwaltung von Anforderungen dient zur Strukturierung(Hierarchische Strukturierung von Anforderung, um eine bessere Auffindbarkeit zu erleichtern), Identifizierung(Dafür Sorgen dass alle Anforderungen eindeutig Identifizierbar sind.), Nachvollziehbarkeit(Dafür sorgen, dass Anforderungen aus einer Zieldefinition oder aus einem Stakeholder-Interesse entstehen), Widerspruchsfreiheit(Dafür sorgen, dass Anforderungen sich nicht gegenseitig widersprechen), Redundanzfreiheit(Dafür sorgen, dass keine doppelten Anforderungen entstehen), sowie zur Pflege des Lebenszyklus von Anforderungen (u.A. also auch RE Change Management und dafür sorgen, dass Anforderungsänderungen auf angemessene Weise dokumentiert werden). Da Änderungen auf die Einhaltung der oben genannten Kriterien geprüft werden müssen, wurde diese Rolle in unserem Projekt auf drei Personen aufgeteilt. Die Anforderungserhebung wurde jedoch vom ganzen Projektteam durchgeführt.

- Scrum Master Der Scrum Master beschäftige sich mit der Einhaltung sämtlicher Spielregeln des Scrum Prozesses damit SCRUM funktioniert. Dabei steht der Scrum Master nicht als Kommandeur und Auftragsgeber über den Team Mitgliedern, sondern als Coach und Berater innerhalb des Scrum Prozesses dem Team zur Seite. Falls Störungen, Hindernisse, welche den Scrum Prozess beeinflussen, auftreten, gilt es diese zu beseitigen. Wichtig sind dabei die Punkte (Transparenz, Überprüfung und Anpassung) des Scrum Prozesses. Als Aufgabe zählten auch die Moderierung und Führung sämtlicher SCRUM Meetings.(Daily Scrum, Planning, Grooming, Review, Retrospektive).

- HPC Management Das HPC Management beschäftigte sich unter anderem mit der Einarbeitung und der Ansammlung von Expertenwissen für das HPC Cluster, dem Netzlaufwerk und GIT. Damit sollte es im Falle von Problemen das HPC hier als erste Instanz Abhilfe schaffen und zur Problembeseitigung zur Seite stehen.

- Qualitätsmanagement Der Qualitätsbeauftragte beschäftigt sich allgemein mit allen Maßnahmen organisatorischer Art, die die Qualität von Prozessen verbessern.

Dazu werden Standards für verschiedenste Prozesse definiert und in Confluence festgehalten. Qualitätsmanagement im Projekt hat zwei Zielrichtungen: Eine hohe Projektqualität (Verlässlichkeit der Projektprozesse) und auch eine hohe Produktqualität (also im Hinblick auf das Projektergebnis). Die Aufgaben die dieser Rolle angehören ist die Planung von Qualität also die Bestimmung, welche Qualitätsziele für das Projekt notwendig sind und Sicherstellung, wie und dass diese Ziele gemessen werden können (analytische QS). Zusätzlich die Festlegung von Maßnahmen, die präventiv für eine bessere Qualität sorgen (konstruktive QS). Die Sicherung von Qualität, indem eine analytische Qualitätssicherung (QS) stattfindet, die eine ständige Messung der Projektqualität (anhand der in der Planung festgelegten Messgrößen) benötigt und durch konstruktive QS-Maßnahmen die Qualität gesteigert wird. Die Steuerung von Qualität beschäftigt sich mit der "Dosierung"der Qualitätssicherung (z. B. Ressourcenbereitstellung) auf Basis von Qualitätsmessungen. Die Aufgaben die im gesamten Projektteam erfolgte waren das Einhalten der Standards und die Korrektur von Prozessen/Prozessinhalten

- Raum und Finanzmanagement Der Raum-/Finanzmanager beschäftigt sich mit der Organisation von Räumen, falls welche benötigt werden. Fürs erste steht der PG Raum V04 1-138 zur Verfügung. Des weiteren gehört die Verwaltung der Kasse, in die für nicht entschuldigtes Fehlen gezahlt werden muss, zu den Aufgaben des Raum/-Finanzmanagers.

2.2. Projekttagebuch

Um den Projektverlauf zu dokumentieren, wurde in JIRA ein Projekttagebuch erstellt und geführt. Damit sollen wichtige Beschlüsse oder Veränderungen, die in der Vergangenheit durch das Team getroffen wurden, für alle Beteiligten, nachvollziehbar gestaltet werden. Damit soll auch eine Transparenz gegenüber den Projektbetreuern und Auftragsgebern geschaffen werden. Ein zentrales Element stellt dazu die Roadmap

dar (s. Abbildung2.1), die dazu dient, den kompletten Zeitverlauf des Projekts für das Projektteam, zu veranschaulichen, welcher jederzeit über das Confluence Board aufrufbar ist. Dabei ist die Roadmap in zwei grobe Bereiche eingeteilt. Zum Einen in Programmierung(gelb) und Sonstiges (blau). In der Programmierung fallen sämtliche Programmiertechnische Aufgaben wie Softwareentwicklung und die Programmierung bzw. antrainieren von Klassifikationsalgorithmen an, die das Projektpersonal erfordert, welches hierfür die Expertise aufweist. Im blauen Bereich sind die Tätigkeiten genannt, zu denen das ganze Projektteam üblicherweise herangezogen wird bzw. das komplette Projektteam in der Lage zu dieser Aufgabe zu arbeiten.

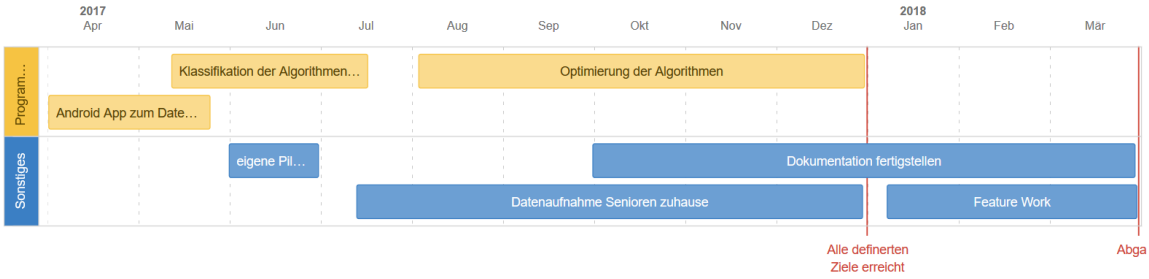


Abbildung 2.1.: Roadmap des Projekts

Im folgenden wird das Projekttagebuch welches in JIRA verfasst und gepflegt wurde, wiedergegeben. Dazu dient als Angabe das Datum, an dem der Beschluss bzw. die Veränderung stattfand. Die Texte bzw. die Organisatorischen Entscheidungen wurden zur besseren Lesbarkeit angepasst. Die Angabe, welcher Bereich des Projektmanagements von dieser Entscheidung betroffen war, wurde ebenfalls entfernt, kann aber in originalen Projekttagebuch im JIRA nachgelesen werden.

Datum	Organisatorische Entscheidung

12.04.2017	Rollenverteilung und Rollendefinition Vorschlag, Stundenzettel zur Projektkontrolle einzuführen wurde abgelehnt. Die Organisatorische Infrastruktur soll unverändert aus der alten Projektgruppe übernommen werden. Das bedeutet, dass JIRA als Werkzeug für die Unterstützung des Vorgehensmodells genutzt, dass von der Gruppe ausgewählte Vorgehensmodell SCRUM ist und zur Dokumentation/Kommunikationen Confluence eingesetzt wird.
19.04.2017	Rollendefinitionen wurden von den Teilnehmern revidiert und verabschiedet.
26.04.2017	• Grobe Ziele des Projektes wurden diskutiert und in Form von Epics im JIRA formuliert. • Zur Verfeinerung der Ziele sollen die identifizierten Stakeholder befragt werden. • Sebastian und Sandra: Interview Auftraggeber • Lana Dasenbrock (Physiotherapeutin): Interview Mediziner/Physiotherapeut
03.05.2017	Der erste Sprint wurde geplant und gestartet. Die Länge wurde probeweise auf eine Woche gesetzt.

10.05.2017	Aus der Retrospektive des ersten Sprints sind weitere organisatorische Entscheidungen entstanden. • Die Aufgaben müssen aus einer Zieldefinition oder aus einer Anforderung heraus entstehen. • Die Aufgabenbeschreibungen müssen ein klares Ergebnis in Form eines Artefakts definieren, sodass das Ergebnis messbar beurteilt werden kann. • Für die Definition der Ergebnis abhängigen Qualitätskriterien wird daher eine neue Rolle Qualitätssicherungsbeauftragter eingeführt. • Für Dokumentationszwecke wird Des Weiteren ein neuer Bereich Aufgabenberichte eingeführt. Die Interviewergebnisse mit den Auftraggebern wurden in der Gruppe besprochen und als Ergebnis eine Liste von Zielen und Anforderungen dokumentiert. Um einen ersten Einblick in die Problemdomäne zu erhalten, werden erste Seminarthemen zur Ausarbeitung vergeben.
17.05.2017	Um entstandene Anforderungen besser verwalten zu können wird eine neue Rolle Anforderungsmanager eingeführt. Als Werkzeug wird Eclipse Plug-In Papyrus eingeführt, das Werkzeug soll für folgende Aufgaben genutzt werden: Reverse Engineering, Anforderungsmanagement und Modellierung.
24.05.2017	Eine Road-Map wurde zur Kontrolle des Projektverlaufes vorgeschlagen und erstellt.

07.06.2017	Um den Projektfortschritt zu bewerten, wurde Vom Auftraggeber ein anonymes Feedback aller Teilnehmer sowie eine Stundentabelle erwünscht. Nach einem intensiven Gespräch mit dem Auftraggeber wurden bereits festgehaltenen Ziele und Anforderungen verworfen, da unterschiedliche Auffassung über die Zielgruppe „Mediziner“ aufgedeckt werden konnte. Die Zielsetzung konzentriert sich ab jetzt auf die Verbesserung der Aktivitätserkennung im häuslichen Umfeld. Die tiefere Untersuchung der Assessments ist nicht mehr Bestandteil der PG.
04.06.2017	Um dem Wunsch des Auftraggebers nach Anwesenheitsinformationen wurde beschlossen, die Abwesenheit der Teilnehmer im Kalender zu dokumentieren. Um das Erledigen von Aufgaben besser verteilen zu können, wurde die Länge eines Sprints probeweise auf eine Woche gesetzt.
21.06.2017	Es wurden Missverständnisse bezüglich Zielsetzungen und dazugehörigen Tasks aufgedeckt und behoben. Es wurde festgestellt, dass die Sitzung der Länge eines Sprints auf eine Woche als Voraussetzung eine genau Schätzung von Tickets erfordert.
05.07.2017	Aufgrund von Änderungen in der Zielsetzung und damit im Umfang der Anforderungen vom 07.06.2017 , werden die Anforderungen ab jetzt nicht in Acceleo, sondern direkt in der Dokumentation gepflegt. Um die Studiendurchführung besser koordinieren zu können, wird von dem Auftraggeber ein Outlook-Kalender organisiert, in dem Termine für die PG-Teilnehmer durch die Abteilungsmitarbeiter organisiert werden.

04.10.2017	Es wurde entschieden, dass eine neue Rolle des Projekt-Controllers/-Manager eingeführt wird, um die Übersicht und Kontrolle über die Leistungen der Projektteilnehmer zu verbessern. Nach gemeinsamer Abstimmung wurde beschlossen, dass Marius Leyh die Aufgaben des Projektmanagers übernimmt.Die Aufgaben müssen von Marius Leyh in Bereich Rollendefinitionen beschrieben werden.

Da zu Beginn eines Projekts der Bedarf an Organisationsaufwänden üblicherweise groß war, ist es nicht verwunderlich, dass das Projekttagebuch in den ersten Monaten viele Einträge enthält. Da der letzte Eintrag vom Oktober 2017 stammt, ist dies als letzter große Beschluss bzw. Veränderung zu nennen. Darauffolgend befand sich das Projekt in der Endphase der Datenaufnahme für Senioren worin der Bedarf an Beschlüssen oder Veränderungen fortan nicht mehr aufkam. Für das Team war die weiterführende Arbeit eindeutig.

2.3. Scrum Nutzung

Im vorherigen Kapitel wurde bereits erwähnt dass PGMAMKSFZ als Vorgehensmodell Scrum gewählt hat.[19] Die Vorteile von SCRUM als Vorgehensmodell liegen in dessen agilem Vorgehen, d.h. das die nächste Projektarbeit in Iterationen sog. Sprints erfolgt. Dazu wird im *Planning* der kommende *Sprint* jeweils geplant indem die Aufgaben verteilt und besprochen werden. Die Dauer eines Sprints kann je nach Projekt und Team auf eine Woche oder mehreren Wochen ausgelegt werden. Für das Controlling dient das *Daily* indem die Projektmitglieder täglich, über ihre gestrige Arbeit und ihr Vorhaben für den bevorstehenden Tag kurz und knapp wiedergeben. Abgeschlossen werden die Sprints in Reviewmeetings indem fertige Zwischenstände für alle anderen Mitgliedern im Projekt und eventuell auch Kunden/Auftraggeber präsentiert werden. Dem Review schließt sich

die *Retrospektive* an, in dem das Team über den letzten Sprint reflektiert und festhält was positiv und negativ aufgefallen war und was man in Zukunft besser machen kann. Dabei können die Probleme allerlei Formen haben: zum Beispiel organisatorische Probleme, zu komplexe Aufgaben oder Kommunikationsprobleme. In all diesen Meetings ist die Aufmerksamkeit und Mitarbeit des gesamten Teams erforderlich um so einen effektiven Mehrwert für den gesamten Scrum Prozess zu schaffen.

Um die Machbarkeit des Scrum Prozesses zu gewährleisten werden die zu erledigten Aufgaben während des Sprints in sogenannten Storypoints geschätzt. Die Schätzung findet im Sprint Plannung oder in extra angesetzten Meetings genannt *Grooming* statt, an dem das gesamte Teams teilnimmt und mithilfe des *Planning-Pokers* eine Zahl der Fibonacci Reihenfolge abgibt, dessen Wert die Komplexität des Task bzw. der Aufgabe repräsentiert. Da jedes Teammitglied seine persönliche Aufwändsschätzung abgibt, wird üblicherweise als finaler Wert der Mittelwert aller Schätzungen genommen. Um aus den Storypoints einen Aufwand abzuleiten, der den Personenstunden oder Tagen entspricht, kann auf sehr unterschiedlichen Wegen erfolgen und den Bedarf einer Berücksichtigung vieler Faktoren auf denen hier nicht weiter eingegangen wird.

2.4. Erfahrung

Da wir uns zu Beginn und die meiste Zeit der Projektdauer einmal wöchentlich trafen, fanden auch an diesem Tag alle SCRUM Meetings statt. Das führte oft zu viel Overhead von Meetings, welcher bei fortgeschrittener Zeit beim Team als schnell zu erledigen angesehen wurde. Es wurde sich häufig zu wenig Zeit dafür genommen, ein ordentliches Review, Retrospektive oder Planning durchzuführen. Eines von SCRUMS Merkmalen, sich in einem Verbesserungsprozess selber zu reflektieren bzw. zu verbessern, gestaltete sich darin schwierig, den Sprint in seinem optimalen Zustand zu kennen und ihn dahingehend zu verbessern. Die Gründe lassen sich dafür nicht eindeutig erklären, aber mangelnde SCRUM Erfahrung im gesamten Team kann ein möglicher Grund

sein. Möglich wäre auch das der Großteil der zu erledigten Aufgaben nicht im Softwareentwicklungsbereich waren, sondern eher Aufgaben aus dem Bereich der Datenanalyse waren. Dadurch dass die Treffen wöchentlich stattfanden, waren die zu erledigten Sprint Tickets als Aufgaben angesehen worden die unabhängig von ihrem Aufwand oder Komplexität meistens zum nächsten Sprint erledigt wurden. Dies hatte zur Folge das die meisten alleine an Ihren Aufgaben arbeiteten und die Arbeit auch nicht im Team stattfand. Vielen fiel zudem schwer in der Retrospektive Aspekte zu nennen, die den Sprint besonders gut oder schlecht machten um so den Sprint besser gestalten zu können. Das erklärt auch den Grund für die immer weniger stattfindenden Retrospektiven. Auch die anderen SCRUM Meetings wurden in der Gruppe immer weniger wahrgenommen und durchgeführt. Letztendlich wurden eher die Reviews und das Planning dazu genutzt um Ergebnisse vorzustellen und die Aufgabenverteilung für die kommenden zwei Wochen zu besprechen. Gegen Ende der Projektlaufzeit wurden auch die Sprints nicht mehr zu festgelegter Zeit abgeschlossen, sondern eher weiterlaufen gelassen, da oft Arbeiten an Dokumentationen und an Modelloptimierungen länger Zeit in Anspruch nahmen als vorher zum Sprint-Planning angenommen. All diese Ereignisse wurden auch von der Gruppe als nicht kritisch oder schlimm empfunden, sondern fanden auch Zuspruch. De Einsatz von SCRUM an dieser Stelle wurde als ungeeignet betrachtet. Ein womöglich besseres Modell wäre KANABAN und würde auch eher dem Vorgehen nahekommen, welches von der PGMAMKSFZ ausgeübt wurde. Dabei werden neue Tickets erst angefangen, wenn das aktuelle Ticket abgeschlossen und gereviewet bzw. abgesegnet worden ist. Dabei spielen Laufzeit eine untergeordnete Rolle. In KANBAN steht vielmehr der Prozessfluss im Vordergrund und dass es nicht zu einem Stau in der Arbeitsphase kommt, in der niemand etwas zu tun hat, weil die entsprechende Voraufgabe nicht beendet worden ist. In unserem Fall würde somit jeder mit einer Aufgabe betraut werden und sobald er damit fertig wäre, würde er es von einer bestimmten Person reviewen lassen oder im kommendem Plenum vorstellen und sich dadurch Feedback von den anderen einholen. Sollte das Feedback negativ ausfallen, würde diejenige Person weiter an dem Ticket arbeiten, ansonsten würde es an einem neuem Ticket anfangen.Die Dauer des Tickets

würde dabei eine untergeordnete Rolle spielen. Leider fand diese Erkenntnis erst zu unserer Dokumentationsphase, also der Endphase des Projektes, statt, sonst hatte dieses Modell womöglich Anwendung gefunden.

3. Anforderungserhebung

Das Projekt PGMAMKSFZ ist ein Folgeprojekt basierend auf Ergebnissen der PGMAMKS, das in deren Rahmen entwickelte System auf Anwendbarkeit im häuslichen Umfeld untersuchen soll. In diesem Kapitel werden die notwendigen Informationen gesammelt, die zur Formulierung konkreter Anforderungen notwendig sind. Da das Projekt ein Folgeprojekt ist, ist es notwendig, die aus der PGMAMKS entstandenen Artefakte, also die Dokumentation, sowie die entstandenen Software auf Verbesserungsbeziehungsweise auf Erweiterungsmöglichkeiten zu untersuchen. Um genauere Anforderungen formulieren zu können, sollen auch relevante Stakeholder identifiziert und befragt werden.

3.1. Dokumentensichtung

In der Abschlussdokumentation der PGMAMKS konnten in der abschließenden Interpretation der Ergebnisse, sowie im Kapitel zum Ausblick folgende zu verbessernde Bereiche identifiziert werden:

- Erweiterung der zu erkennenden Aktivitäten. Das finale Modell kann lediglich fünf Aktivitäten erkennen. Zur Identifizierung von TUG und 6-Minuten-Gehtest fehlt die Aktivität Umdrehen.

- Plausibilitätsprüfung der Annotationen. Die Teilnehmer der PG schlugen vor, die klassifizierten Label einer logischen Prüfung zu unterziehen. Aktivitäten, die aus logischen Gründen nicht an der Stelle auftreten können sollen entsprechend herausgefiltert werden.

- Verbesserung der Klassifikationsgenauigkeit. Neuronale Netze bieten die Möglichkeit, für das Vorhandensein einer Klasse eine Wahrscheinlichkeit anzugeben. Der Verbesserungsvorschlag der PG besteht darin, die Modelle die zur Angabe von Wahrscheinlichkeiten fähig sind zu nutzen und bei der Angabe von einer geringen Wahrscheinlichkeit den Datenausschnitt nicht zu beschriften. Den Anmerkungen der Betreuenden und des ersten Gutachters war zu entnehmen, dass eine weitere Funktionalität zur Klassifizierung unsicherer Bereiche als UNBEKANNT erwünscht war, die in dem Ansatz der PGMAMKS nicht umgesetzt worden war.

- Automatisierte oder Semi-automatisierte Beschriftung. Zur schnelleren und genaueren Beschriftung der Daten sollen an Stelle von manuellen Beschriftung Konzepte ausgearbeitet werden, mit deren Hilfe ein Datensatz halbautomatisch vorbeschriftet werden kann.

- Entwicklung und Erproben weiterer Klassifikationsalgorithmen, die im Rahmen der PGMAMKS nicht umgesetzt werden konnten. Als mögliche aussichtsreiche Kandidaten wurden im Ausblick-Kapitel Faltende Neuronale Netze sowie Reccurent Neural Networks genannt.

- Untersuchen weiterer Ansätze zur Dimensionsreduktion und Merkmalsanalyse. Hierzu wurde, als eine vielversprechende Möglichkeit, die Lineare Diskriminanzanalyse genannt.

- Untersuchung der Klassifikationsgenauigkeit anhand von Daheim-Daten. Die PG-Teilnehmer vermuteten, dass das auf Assessments-Daten trainierte Modell schlechtere Ergebnisse liefern würde. Das finale Modell soll daher diesbezüglich untersucht werden. Die PG-Teilnehmer weisen hier auf die Schwierigkeiten bei der Beschriftung der Daheim-Daten hin und empfehlen zur Validierung der Beschriftungen ein sicheres Referenzsystem. Als Beispiel für ein mögliches Referenzsystem wurde die Videoaufzeichnung genannt.

3.2. Befragung der Stakeholder

Aus der Dokumentation der PGMAMKS wurde ersichtlich, dass eine Stakeholderanalyse im Rahmen der Anforderungserhebung nicht stattgefunden hat. In den Beschreibungen der Anwendungsfälle wurden als Stakeholder Proband, Humotion-Software, Auswerter, Entwickler (Nutzer), medizinisches Personal (Nutzer) identifiziert . Wie die Abbildung 30 in der PGMAMKS-Dokumentation zeigt, wurde das MAMKS-System mit Nutzungsschnittstellen sowohl für Entwickler, als auch für medizinisches Personal geplant. Die Umsetzung wurde durch ein einheitliches User-Interface realisiert.

Aus der Beschreibung der Anwendungsfälle wurde deutlich, dass kein gemeinsames Verständnis für die Art der zu gewinnenden Informationen zum Mobilitätszustands der Sensorträgers und angemessene Darstellung erarbeitet wurde. Die Interessen der Stakeholder medizinisches Personal wurden nicht untersucht. Aus diesem Grund haben sich die Teilnehmer der PGMAMKSFZ dazu entschloßen, die Stakeholderanalyse durch eine Befragung nachzubessern. Als relevante Stakeholder wurden die Auftraggeber, die Repräsentanten des medizinischen Personals, sowie Entwickler von Datenanalysewerkzeugen identifiziert.

3.2.1. Interview Physiotherapeuten

Das Ziel der Befragung von Physiotherapeuten, als relevante Repräsentanten des medizinischen Personals, war es die deutlichere Herausarbeitung ihrer Interessen in Hinsicht auf die Darstellung der Daten.

Für die Anforderungsanalyse wurden zwei Interviews mit Physiotherapeuten durchgeführt. In diesem Gespräch sollte zum einen das Wissen zur Bewegungsanalyse und Beurteilung der Mobilität vermittelt werden. Zudem sollten mögliche Anforderungen und Erwartungen zur technikgestützten Bewegungsanalyse erfragt werden. Folgende Fragen

wurden im Interview behandelt:

1. **Wie erfolgt zur Zeit die Beurteilung der Mobilität im häuslichen Umfeld?**

   Zum einen über die Kraft. Das heißt wie schnell kann ein Patient von einem Stuhl aufstehen. Zum anderen über die Wegstrecke, dass heißt, wie viel Meter schafft derjenige zu gehen. Ein weiterer Punkt ist die Koordination und das Gleichgewicht. An diesen wird gemessen wie sicher der Patient steht (zum Beispiel beim Kaffee zubereiten).

2. **Welche Werte wären interessant, um eine Beurteilung über die Bewegung machen zu können?**

   Eine Möglichkeit wäre die Gehstrecke, die der Patient schafft, um einen Vergleich zu ziehen. Des weiteren wäre die Zeit interessant, die er braucht, um vom Stuhl aufzustehen und ob die Hände zur Stabilisierung gebraucht werden oder nicht.

3. **Welche Bewegungen haben eine größere Aussagekraft?**

   Das Becken, der Rumpf und die Beinstellung. Dann kommt wieder die Zeit zum Aufstehen und die Gehstrecke im Vergleich

4. **Welcher Informationen sind zur Beurteilung der Mobilität für einen Mediziner wichtig/oder eher unwichtig?**

   Für die Mediziner wäre die Gehstrecke eine wichtige Information. Zugleich wäre es Interessant, ob der Patient im Alltag zurecht kommt. Das heißt Alltagsdinge, wie Wäsche waschen.

5. **In welcher Form und auf welchen Wegen erfolgt die Berichterstattung vom Pflegepersonal, medizinischen Assistenten an den behandelnden Arzt?**

   Zurzeit erfolgt die Berichterstattung über den TuG (Timed up and Go) Test. Außerdem wird die Gehstrecke übermittelt sowie die Kraft und Ausdauer, zum Bei-

spiel beim Aufstehen.

6. **Wie kann man ohne Testverfahren das Gleichgewicht testen z.B beim Stehen**

   Eine Möglichkeit wäre der TuG Test. Eine zweite Variante wären Gleichgewichtsübungen. Zum Beispiel sind Schwankungen im Oberkörper sichtbar. Hält der Patient sich dabei fest oder macht er einen Ausfallschritt.

7. **Welche Erwartungen/Befürchtungen haben Sie an die technikgestützte Bewegungsanalyse?**

   Es sollte bei der Arbeit unterstützen und keinen extra Aufwand generieren. Es sollte möglichst übersichtlich in der Handhabung sein und eine übersichtliche Ausgabe erfolgen, zum Beispiel in Form von Grafiken.

8. **Welche Information wären wichtig oder interessant beim Sensor Gürtel bzgl. Mobilität?**

   Informationen, wie die Gehstrecke bzw. das Verhältnis vom Gehen und Stehen wäre hilfreich. Es sollte *harmonisch* beim Gehen aussehen. Weitere wichtige Parameter beim Gehen wären: Die Schrittlänge, Kadenz (wie Harmonie, Geschwindigkeit), Schwankungen beim Gehen, wie ist der Abstand der Knöchel, Beckenverlauf muss [harmonisch] sein.

9. **Welche Werte sind interessant bei Physiotherapeuten um eine Beurteilung über die Bewegung aussagen zu können?**

   Wie schnell ist die Schnellkraft. Dabei wären bereits ausgewertete Parameter hilfreich, woran man einen Leistungsabfall erkennen könnte.

10. **Wie würdest du die Ausdauer im Alltag beurteilen?**

    Die Gestrecke ohne Pause. Eine schwache Leistung wären zum Beispiel 400-500 Meter. Dabei wäre 1 Kilometer normal. Alles unter 300 Meter wäre nicht ausreichend.

3.2.2. Interview Auftraggeber

Für die Anforderungsanalyse wurden zwei Interviews mit den Auftraggebern durchgeführt. In diesem Gespräch sollte zum einen das Wissen zu vorangegangenen Ergebnissen sichergestellt werden. Zum anderen sollten gemeinsame realistische Ziele und Anforderungen formuliert werden. Folgende Fragen wurden im Interview behandelt:

Interview mit Dr. Sebastian Fudickar

1. **Für welchen Verantwortungsbereich bist du in diesem Projekt zuständig?**
   Betreuung von Sandra Helmers. Installation des Messsystems, Programmierung der dazugehörigen Software und Organisation zentralisierte Datensammlung. Managementangelegenheiten (Datenschutzkonzept, Datenmanipulationssicherheit, Berechtigungen unterschiedlicher Nutzer). Probanden-Akquise (z.B. aus Fitnessstudios, Flyer). Vor- und Nachbereitung der Assessments

2. **Einordnung der Arbeit unserer Projektgruppe in die Projektstruktur?**
   Aequipa, Soziologisches Konzept, BIPS (Bremen) Versa ist ein Teilprojekt aus dem Bereich Aequipa Technologie. Arbeiten mit den Daten aus der Versa-Studie.

3. **Ziel des Projekts? (Versa)**
   Entwicklung von Prädiktoren für den funktionellen Abbau im Alter. Versa bedeutet Vorhersage zum Erhalt der Selbstständigkeit im Alter „Ziel der Studie ist es zu überprüfen, ob ein sensorgestütztes Assessment als Zwei-Punkt-Deltamaß über 6 Monate einen geeigneteren Ansatz darstellt als eine Einpunktmessung um bei Seniorinnen und Senioren frühzeitig einen funktionellen Abbau zu erkennen. Etablierte geriatrische Tests der körperlichen Funktionalität und Mobilität, die Messung der Körperzusammensetzung und eine sensorgestützte Bewegungsanalyse werden zur Charakterisierung der Studienteilnehmer an-

gewendet. Die Bewegungsanalyse sowie eine längerfristige Analyse der individuellen körperlichen Aktivität werden mithilfe eines Sensorgürtels (Accelerometer, Gyroskop, Magnetometer und Barometer) möglich.“ Quelle: https://www.uni-oldenburg.de/medizintechnik/forschung/projekte/aequipa/versa/

Assessments zu den verschiedenen Zeitpunkten. Abweichungen bestimmen und beurteilen, ob funktioneller Abbau sichtbar geworden ist. Erste Auswertungen zeigen bei 68 von 246 Teilnehmern bereits zwischen den ersten beiden Assessment einen funktionellen Abbau. SPPB-Index (short physikal pervormance battery) beinhaltet Grenzwerte.

Übergeordnetes Ziel: Es müssen technische Messverfahren entwickelt werden, mit denen überhaupt etwas messbar ist, was ein ausschlaggebender Indikator sein könnte.

4. **Typischer Anwendungsfall für die Ergebnisse der Versa-Studie?**

   Ein optimales Ergebnis wäre die Möglichkeit einer Durchführung von Assessments ohne dem Patienten etwas anzulegen, Metadaten, z.B. weniger SMS schreiben aber 100% sicheres Ergebnis, aufleuchten einer Lampe als Warnung bei erkanntem Risiko. Dieses Szenario beschreibt Sebastian als unrealistisch. Ein guter Mittelweg wäre es, den Probanden ein Messverfahren mit zu geben, mit dem angesprochene Punkte messbar sind. Alternativ auch ambiente Sensoren, dann jedoch nur im häuslichen Bereich messbar und nicht im gesamten Alltag (Kritik: zu Hause nur kurze Wege, keine Höchstgeschwindigkeiten). Aus diesem Grund ergeben körpernahe Sensoren ein umfassenderes Bild.

5. **Wer sind zukünftige Anwender bzw. wer erhält die Auswertung der Daten?**

   Es gibt Modelle von Tele-Healthcare. Momentan schwierig bezüglich Abrechnung, hat gerade erst begonnen. Oder allgemeine Verwaltung und Vergabe über Krankenkassen mit Fallmanager. Wer hat Interesse an Vermarktung? Beispiel: Fit Bits nicht gut – Verbesserung entwickeln, die auch altersgerecht, sensitiver sind. Insge-

samt ist ein konkretes Nutzungsszenario noch offen. Erst soll die Technik entwickelt werden und die Möglichkeiten zur Nutzung geschaffen werden.

6. **Welche Unterschiede sind bezüglich der Aussagekraft der Heimdaten im Vergleich zu den Assessments zu erwarten?**
   Weißkittel-Syndrom, das heißt der Proband verhält sich anders beim medizinischen Assessment. Zuhause (alltagsintegriert) gehen Probanden nicht an ihr Limit. Das hat zur Folge, dass eine ganz andere Art der Performance gemessen wird. Ein weiterer interessanter Aspekt ist das Auftreten der Tragefehler – Gürtel richtig herum? gleiche Platinenposition? ausreichende Ladung? (wichtig für Zwischenfilter).

7. **Was sind für euch die wichtigsten Schlussfolgerungen aus der vorherigen Projektgruppe?**
   Haben ersten Schritt gemacht. Programm hat ca. 90% Erkennungswahrscheinlichkeit. Erkennung vom Stehen im Bereich Sitzen, Ursachen nicht erkennbar. Es ist interessant zu erfahren, ob diese Aktivitäten besser von einander unterschieden werden können.

8. **An die neue PG gestellten Erwartungen?**

   - Algorithmen auf Eignung überprüfen
   - Sinnfrage : Sitzen-Stehen Transition
   - Einbauen von Regeln oder andere Nachbearbeitungsschritte: Kann man durch Nachfiltern mit logischen Modellen die Erkennungsgenauigkeit erhöhen (angucken, in wie weit die bisherigen Ergebnisse Sinn machen)?
   - Algorithmen auf Eignung mit Heimdaten untersuchen. Auch neue Modelle untersuchen.
   - Einführen der nicht kategorisierten Bereiche (Nullkategorie) und Konfidenzangabe für die Klassifikationswahrscheinlichkeit.

- Label-Vorverarbeitung: Automatisierte Vorannotation mit anschließender manueller Korrektur.

- Ist es sinnvoll Bewegungsabfolgen auf Assessments zu matchen?

Interview mit Sandra Hellmers

1. **Für welchen Verantwortungsbereich bist du in diesem Projekt zuständig?**
   Physikerin. Technikbetreuung (Gürtel, Technik im Raum). Datenarchivierung, Strukturierung, Revision. Gürteldaten-/Technikdatenauswertung.

2. **Einordnung der Arbeit unserer Projektgruppe in die Projektstruktur?**
   Versa: zwei Bereiche: Zum einen Assessment, zum anderen Datenaufnahme zu Hause. Wir sind für Heimassessments zuständig.

3. **Ziel des Projekts? (Versa)**
   Alltagsintegrierte Assessmentdaten aufnehmen, z.B. Gehstrecke, sowie bessere Daten, weniger Arbeitsaufwand/Kosten. Deltamaß entwickeln, für insgesamt drei Messungen. Aus ersten beiden Messzeitpunkten ein Deltamaß/die Veränderung ermitteln, um 3. Messung vorhersagen zu können. Einzelmessung oder Deltamaß für Vorhersage besser?

4. **Typischer Anwendungsfall für die Ergebnisse der Versa-Studie?**
   Patient geht zum Hausarzt, bekommt Gürtel für eine Woche. Anschließend bekommt der Hausarzt eine Auswertung/Zusammenfassung (was ist gut/schlecht, wo gibt es Defizite?) und ggf. die Möglichkeit Daten anzugucken (eher theoretisch, haben vermutlich nicht ausreichend Fähigkeiten). Die zu klärende Frage ist: Was ist gewünscht?

5. **Welche Unterschiede sind bezüglich der Aussagekraft der Heimdaten**

im Vergleich zu den Assessments zu erwarten?

Bessere Daten, Probanden fühlen sich nicht beobachtet oder verstellen sich nicht, Probleme fallen im Alltag mehr auf, sensitiver, man sieht mehr/Realität. Längere Aufnahmen, nicht nur Momentaufnahme. Vielleicht auch ganz andere Parameter abfragbar, die im Assessment nicht darstellbar sind (laufen auf unterschiedlichem Untergrund).

6. **Was sind relevante Informationen, die aus den Assessments gewonnen werden können?**

   Veränderung im sppb (ergibt sich aus Assessments, wird langsamer).

7. **Was sind für euch die wichtigsten Schlussfolgerungen aus der vorherigen Projektgruppe?**

   Aktivitätserkennung stand im Vordergrund, haben gut gelöst durch hierarchisches Modell. Analysen am Ende haben nicht mehr geklappt.

8. **An die neue PG gestellten Erwartungen?**

   - Werden Aktivitäten auch zu hause erkannt? Die Liste der Aktivitäten soll erweitert werden.
   - Daten weglassen immer relativ, was wollen wir messen? Was Wollen die Geriater/Mediziner wissen? Sind kurze Gehstrecken überhaupt relevant?
   - Was gibt es für Abweichungen bei unterschiedlichen Stühlen/ unterschiedlichen Untergründen
   - Erweiterung um Fahrradfahren. Diese Tätigkeit hat eine hohe Aussagekraft.
   - Analyse der Aktivitäten bezüglich ihrer Dauer (Geriater: längste Episode mit hoher Geschwindigkeit), Balance, Power, Kraft
   - Unterscheidung der Aktivitäten draußen/drinnen (anderes Gangverhalten ge-

radeaus, Türen, Unterbrechungen?).

- Umgestaltung Tagebuch? (können uns alte angucken (digitalisiert - openClinika)) – genauere Abfragen von relevanten Daten – z.B. zum Ankreuzen.

3.2.3. Interview PG-Teilnehmer

Da aus der Dokumentation der PG-Mamks nicht ersichtlich war, welche Datensätze genau zur Optimierung des finalen Modells genutzt worden sind, wurde Jonas Prellberg diesbezüglich befragt.

Ziele der Befragung waren, die Nutzung gleicher Datensätze zum Anlernen der MATLAB- und Python-Algorithmen, die auch für die Validierung der MAMKS-Algorithmen genutzt wurden, um vergleichbare Ergebnisse zu erhalten.

Um die Binär-Daten der PG-Mamks nach MATLAB importieren zu können waren nähere Informationen zum Binärformat nötig.

1. **Welche Datensätze wurden zur Validierung genutzt?**
   Alle Datensätze, die zur Validierung genutzt worden sind, sind in den Skript-Dataeien git/scripts/*.slurm aufgelistet Demnach sind die Daten aus dem Ordner Binärdaten mit den Ordnernamen 12903, 18430, 15713, 28263, 23880, 24117, 20378, 20714, 47056, 22273, 20502, 12117, 11547, 35632, 39707, 37747, 42529, 33467, 31106, 36166, 40860, 37615, 37376, 29231 Zur Validierung genutzt worden. Aus der Liste ist ersichtlich, dass die Zusatzdaten, die von der PG-Mamks zur Verbesserung der Ergebnisse erhoben worden sind und mit einer 9 im Ordnernamen beginnen, nicht für die Optimierung genutzt worden sind.

2. **In welchem Fomat sind die Binärdateien gespeichert?**
   Die *.bin Dateien enthalten float32-Datenreihen.

3.2.4. Interview mit Fachexperten (Datenanalyse, Machine Learning)

Um die Qualität der Umsetzung des MAMKS-Systems besser beurteilen zu können, sollten ursprünglich Fachexperten aus dem Bereich der Datenanalyse und des Maschinellen Lernens befragt werden. Dies ist leider nicht erfolgt, da keine Fachexperten zu einem Interview bereit waren. Aus diesem Grund wurden statt einer direkten Befragung Datenanlysewerkzeuge untersucht, die in der Domäne der Human Activity Recognition und Signaldatenanalyse eingesetzt werden.

Java als Werkzeug zur Datenanalyse zu benutzen erfordert sehr viel Implementierungsaufwand, da sie eine universelle Programmiersprache ist. Es existiert zwar Deeplearning4j als eine Bibliothek für Maschinelles Lernen, diese nutzt jedoch selbst extern entwickelte Modelle zur Klassifizierung oder Regression. Eine Entwicklung von eigenen Modellen ist mit diesem Werkzeug, verglichen mit Pythons Werkzeugen, wie Keras, SciKit Learn, oder mit MATLABs Machine Learning Toolbox ist unverhältnismäßig aufwändiger.

Ein weiteres Problemfeld ist die Darstellung der Daten. Um die Signaldaten performant darzustellen ist in Java eine aufwändige Implementierung notwendig. Dies wurde in der Dokumentation der PG-Mamks bereits am Beispiel des Rendering-Problems der Signaldaten beschrieben. Um weitere Visualisierungsmöglichkeiten, beispielsweise von Cluster-Darstellungen, oder Darstellung von Separierbarkeit der Daten anhand beliebiger errechneter Merkmale über verschiedenen Achsen würde den Implementierungsaufwand soweit erhöhen, dass für die eigentliche Anlyse der Dateneigenschaften kaum Zeit bleiben würde. Auch dies war das Ergebnis der PG-Mamks, da letztendlich nur ein Ansatz erprobt werden konnte. Sowohl MATLAB, als auch Python sind optimale Datenanalyse-Werkzeuge. Trotz vergleichsweise niedrigeren Performanz, wie in der Dokumentation der

PG-Mamks argumentiert wird, sind viele notwendigen Werkzeuge zur Darstellung von Daten und schnellen Formulierung von Berechnungsformeln bereits enthalten. Das dritte Problem ist die Akzeptanz der Zielgruppe Entwickler. Sowohl MATLAB, als auch Python-Bibliotheken zur Datenanalyse sind etablierte und verifizierte Werkzeuge.

3.3. Herausforderungen bezogen auf die Studiendurchführung

Mit der MAMKS Software steht bereits ein leistungsstarkes Werkzeug zum Anzeigen von Bewegungsdaten zur Verfügung. Mit ihr können die vom Humotion-Sensorgürtel erfassten Daten importiert und anschließend angezeigt werden. Das Ziel ist es die Daten anschließend automatisch zu klassifizieren. Es ist in der MAMKS-Software zwar möglich Algorithmen zu implementieren, allerdings ist dies mit einem sehr hohen Arbeitsaufwand verbunden. Es soll möglich sein angelernte Modelle dynamisch auszutauschen und eventuell im Nachhinein anzupassen. Hierzu wird eine elegante Lösung benötigt, die es mit geringem Aufwand ermöglicht neue Modelle zu implementieren und mit ihnen die Daten zu klassifizieren, um möglichst exakte Ergebnisse zu erhalten.
Hierzu wird in Betracht gezogen die Modelle mit MATLAB oder Python zu erstellen und eine Schnittstelle für MAMKS zu integrieren. Im Laufe des Projekts muss diese Möglichkeit geprüft und eine entsprechende Lösung erarbeitet werden.
Außerdem ermöglicht es die Software die importierten Daten per Hand zu klassifizieren. Dies ist nötig für Klassifizierungsalgorithmen, die dem Muster des „Supervised Learning“ folgen (Daten müssen bereits klassifiziert sein um mit ihnen lernen zu können).

- Bei der Datenerhebung müssen die Daten klassifiziert werden. Hierzu wird ein entsprechendes Tool benötigt. Eine Dokumentation mit einem Zettel oder einem zusätzlichen Programm auf einem Mobiltelefon sind denkbar.

- Die Daten müssen nach dem Klassifizieren in ein entsprechendes Format gebracht

werden, damit sie von MAMKS verarbeitet und als Label angezeigt werden können. Hierzu bietet sich die Benutzung einer APP an, die Labeldaten MAMKS-konform speichert.

- Um entsprechende Klassifizierungsmodelle anlernen zu können muss ein Tool ausgewählt werden, dass einfach zu handhaben ist und Modelle in einem einheitlichen Format erstellt.

- Die in MAMKS bereits implementierten Klassifizierungsalgorithmen wurden mit entsprechenden Assessmentdaten in einem kontrollierten Umfeld angelernt. Es kann davon ausgegangen werden, dass eine Adaption der Klassifizierungsmodelle keine guten Ergebnisse erzielen wird.

- Damit die Modelle die Daten klassifizieren können, muss eine Schnittstelle in MAMKS geschaffen werden, mit der es möglich ist die Modelle auszutauschen und anzuwenden.

- Für ein möglichst gutes Klassifizierungsergebnis müssen verschiedene Algorithmen getestet werden. Hierbei ist eine Parameteranpassung und Bewertung der einzelnen Algorithmen notwendig. Hierbei sollte beachtet werden, dass ein Anlernen eines Modells einen hohen Zeitaufwand kosten kann.

- Damit für die Stakeholder die benötigte Übersicht der Aktivitäten angezeigt werden kann, muss eine entsprechende Anzeigefunktion in MAMKS integriert werden. Sie soll anzeigen wie viele Aktivitäten der jeweiligen Klasse vorhanden sind und wie lange die jeweilige Dauer ist.

- Da die Umgebungen der Probanden sehr unterschiedlich sind, wird es für die Modelle schwer eine genaue Klassifizierung vorzunehmen.

- Es ist denkbar das viele Aktivitäten nicht so häufig durchgeführt werden, da sie im alltäglichen Leben der Probanden nicht oft durchgeführt werden (zum Beispiel

springen). Dies grenzt die Datenbasis zum Anlernen dieser Aktivität erheblich ein.

- Außerdem ist denkbar das Menschen im privaten Umfeld, wenn sie sich nicht beobachtet fühlen, ihre Bewegungen eher entspannter durchführen, als wenn sie beobachtet werden. Dies kann zu sehr ungenauen Sensordaten führen, was eine automatische Klassifizierung zusätzlich erschweren könnte.

3.4. Formulierung der Anforderungen

Basierend auf Dokumentensichtungen, den Verbesserungsvorschlägen der PGMAMKS, den Interviews, den Untersuchungen der gegenwärtigen Software-Lösung, sowie auf den identifizierten Herausforderungen bezogen auf die Studiendurchführung, werden zunächst die in dieser Projektgruppe zu behandelnden Anwendungsfälle und danach, die identifizierten Anforderungssätze formuliert.

3.4.1. Anwendungsfälle

Insgesamt wurden drei Anwendungsfälle identifiziert. Zum einen wird eine automatische, oder eine halbautomatische Vorverarbeiung der Annotationen angestrebt. Der zweite Anwendungsfall ist die Entwicklung und Evaluation neuer Klassifikationsalgorithmen. Schließlich behandelt der dritte Anwendungsfall die Nachbearbeitung, der mit Hilfe von Klassifikatoren erzeugten Label.

Die Abbildung 3.1 zeigt den ersten zu behandelnden Anwendungsfall. Um die automatische Vor-Annotation zu ermöglichen wird zuerst ein Datensatz ausgewählt und anschließend ein Vorverarbeitungsalgorithmus für Labels ausgeführt. Geplant sind zunächst Vorverarbeitungsschritte für die automatisierte Beschriftung von Drehbewegungen, wie Drehen nach links, Drehen nach rechts, Umdrehen nach rechts im Liegen, sowie das

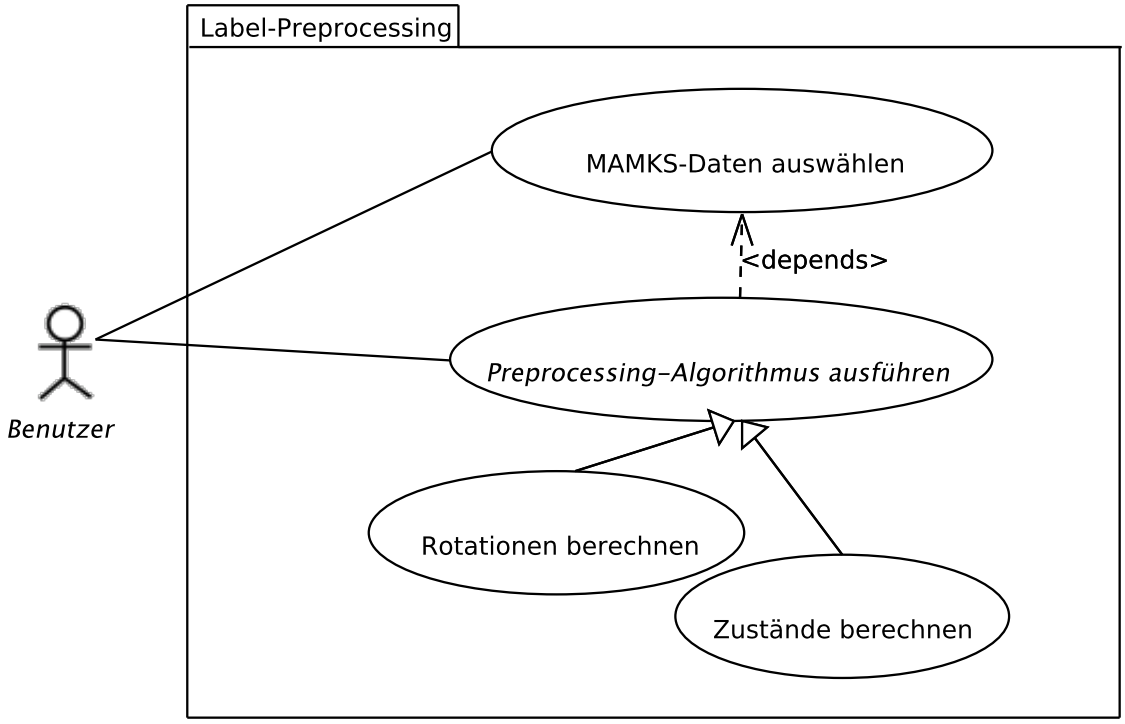


Abbildung 3.1.: Anwendungsfall 1 Vorverarbeitung von Labeln

Umdrehen nach links im Liegen. Diese Vorverarbeitung ist relativ einfach umsetzbar, da sie technisch gesehen leicht zu berechnende Aktivitäten sind. Die Drehungen erfolgen in der Y-Achse und sind über die Dauer einer bestimmten Radialgeschwindigkeit relativ einfach zu beschreiben. Zweite Vorverarbeitungsmöglichkeit ist die automatisierte Segmentierung der Datenbereiche in statische und dynamische Abschnitte. Diese können anhand des Merkmals "Signal Vektor Magnitude" über eine Dauer von 2-3 Sekunden errechnet werden und benötigen im Prinzip keine komplizierten Klassifizierungsmodelle.

Der zweite Anwendungsfall wird in der Abbildung 3.2 beschrieben. Hierbei handelt es sich um die Möglichkeit ein externes Modul zu entwickeln, das die Daten innerhalb des MAMKS-Workspace klassifiziert und die Ergebnisse der Klassifizierung in der, für das MAMKS-Frontend lesbaren Form abspeichert (Label). Die geplanten Umsetzungen sind, wie bereits beschrieben, eine MATLAB-Toolbox sowie Python-Module zur Klassifikation

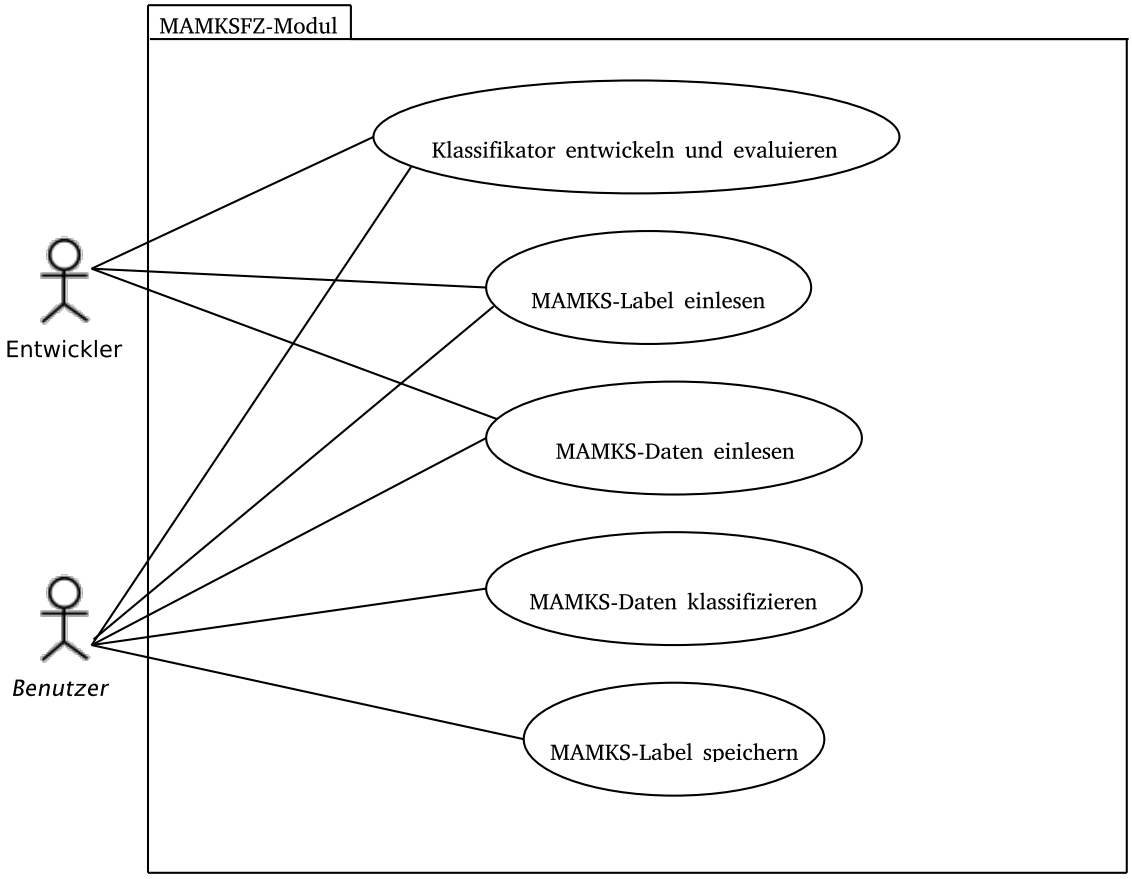


Abbildung 3.2.: Anwendungsfall 2 Erweiterung um Klassifikationsalgorithmen

von MAMKS-Daten. Jedes Modul, das zur Klassifikation von MAMKS-Daten umgesetzt wird, muss die in dem Anwendungsfalldiagramm 3.2 identifizierten Aktivitäten umsetzen.

Der letzte zu behandelnde Problembereich ist die Nachbereitung der Klassifikationsergebnisse. Die Abbildung 3.3 zeigt die identifizierten Aktivitäten in Verbindung mit diesem Anwendungsfall. Zunächst wurden zwei Typen für die Nachbearbeitung der Annotationen identifiziert. Zu einem kann mit Hilfe des sogenannten Mehrheitsentscheid aus den Ergebnissen mehrerer Klassifikationsmodelle ein neues Klassifikationsergebnis erzielt werden. Die Erwartung ist, dass nach multipler Klassifikation die Mehrheit der

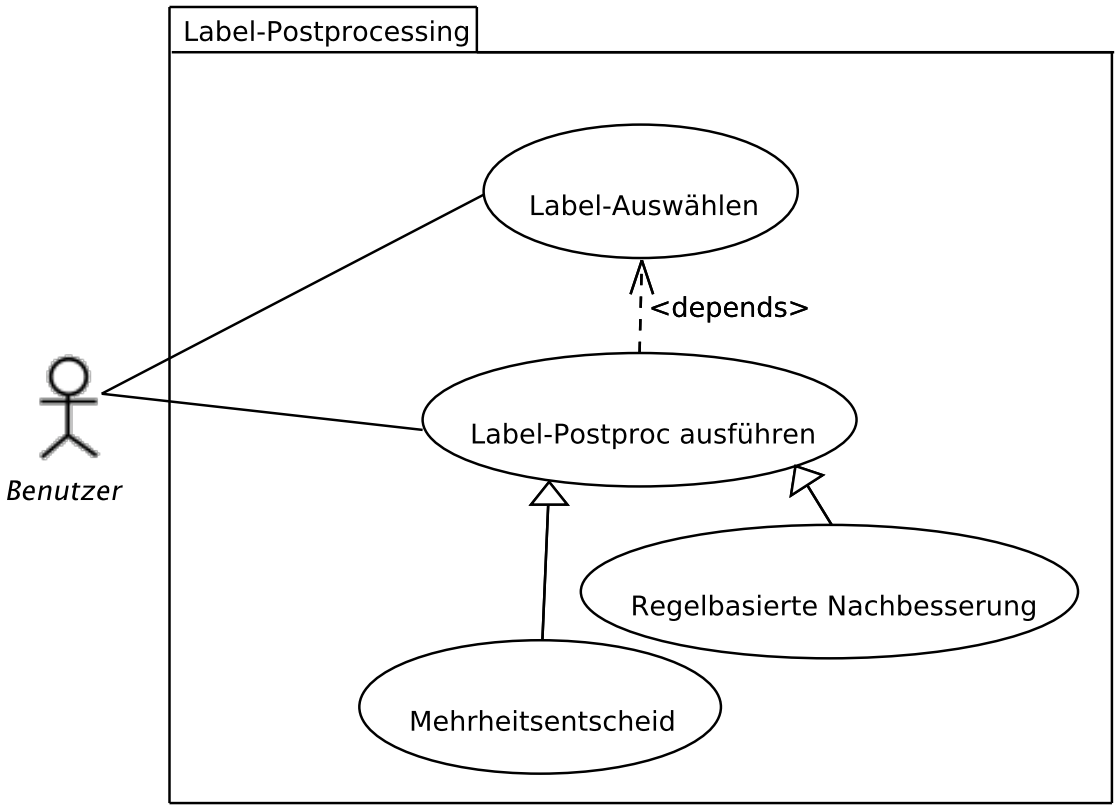


Abbildung 3.3.: Anwendungsfall 3 Nachbearbeitung von Labeln

Modelle an der Stelle ‘n’ im Datensatz eine richtige Schätzung abgeben. Diese Art des Mehrheitsentscheides kann auch als vertikaler Mehrheitsentscheid bezeichnet werden. Eine weitere Erwartung besteht darin, dass auch innerhalb eines Modells, im Datenbereich zwischen ‘n-w’ bis ‘n’, wobei n eine Stelle im Datensatz und ‘w’ eine Fensterbreite ist, die meisten Schätzungen richtig sind. So könnte der gesamte Bereich w der am meisten abgegebenen Schätzung zugeordnet werden. Diese Art des Mehrheitsentscheids wird auch horizontaler Mehrheitsentscheid genannt. Auch eine Mischung aus horizontalen und vertikalen Mehrheitsentscheiden ist denkbar. Wie in dem Abschlussbericht der PG-Mamks, sowie den Interviews zu entnehmen, ist auch eine regelbasierte Nachbereitung denkbar. Es ist möglich, dass einige Klassifikationsfehler durch logische Nachprüfungen eliminiert werden können. Beispielsweise können die maximale und die minimale Dauer

einer Aktivität auf ihre Plausibilität überprüft werden.

3.4.2. Funktionale Anforderungen

Aus den oben betrachteten Anwendungsfällen werden nun konkrete Anforderungssätze formuliert.

A1: Das MAMKS-System wird auf die Möglichkeit einer Anbindung von externen Systemen wie aus MATLAB oder aus Python geprüft und ggf. erweitert.

A2: Es wird eine MATLAB-Umgebung entwickelt, die in der Lage ist, die in einem MAMKS-Workspace enthaltenen Daten zu verarbeiten. Geplant sind Umsetzungen von Klassifikationsalgorithmen. Potenziell soll die Schnittstelle dazu in der Lage sein auch andere Verarbeitungsschritte, wie Filterung oder Sensorfusion zu ermöglichen.

A2.1: Es wird eine Import-Funktion für das Einlesen der Daten für MATLAB aus dem MAMKS-Workspace implementiert

A2.2: Es wird eine Export-Funktion für die Klassifikationsergebnisse der, in der MATLAB-Umgebung klassifizierten Daten zum MAMKS-Workspace implementiert.

A2.3 Es werden Klassifikationsalgorithmen Boosted Decision Trees, Bagged Decision Trees und die Lineare Diskriminanzanalyse auf ihre Klassifikationsgenauigkeit untersucht.

A3: Es wird eine Python-Umgebung entwickelt, die in der Lage ist, die in MAMKS-Workspace enthaltenen Daten zu verarbeiten. Auch hier sind zunächst nur Klassifikationsalgorithmen zur Erkennung von Aktivitäten geplant.

A3.1: Es wird eine Import-Funktion für das Einlesen der Daten aus dem MAMKS-Workspace für die Klassifikation innerhalb der Python-Umgebung implementiert

A3.2: Es wird eine Export-Funktion für die Klassifikationsergebnisse der in Python-Umgebung klassifizierten Daten für den MAMKS-Workspace implementiert.

A3.3 Es werden Klassifikationsalgorithmen Convolutional Neural Networks und K-Nearest Neighbor in der Python-Umgebung auf Klassifikationsgenauigkeit untersucht.

A4: Um Klassifikationsalgorithmen entwickeln und auswerten zu können müssen rele-

vante Daten erhoben werden. Hierzu muss ein Konzept zur Erhebung der Daheim-Daten entwickelt werden.

A5: Zur Erhebung von Daheim-Daten ist ein geeignetes Annotationswerkzeug notwendig. Denkbar wären hier Annotationen mit Videoaufnahmen, mit Hilfe eines Notizblockes, mit Hilfe von Tagebuchstudien, oder eines elektronischen Werkzeugs, wie Smartphone zum Tracken der jeweiligen Aktivität.

A5.1: Das Annotationswerkzeug muss auf seine Wirksamkeit und Genauigkeit untersucht werden.

A6: Das manuelle Annotieren der Daten in MAMKS-System wird verbessert.

A6.1: Um die manuelle Annotierung zu verbessern, müssen die Usability-Aspekte untersucht und optimiert werden.

A6.2: Um die manuelle Annotierung zu erleichtern, müssen Möglichkeiten von automatisierten Nachbesserungen, beziehungsweise Vorbereitungsschritten untersucht werden.

A7: Das Klassifikationsergebnis muss verbessert werden.

A7.1: Um Klassifikatonsergebnisse zu verbessern, müssen Nachbereitungsschritte untersucht werden. Es müssen Möglichkeiten eines Mehrheitsentscheids für mehrere Klassifikationsalgorithmen untersucht werden.

A7.2: Zur Verbesserung der Klassifikationsergebnisse soll ein regelbasiertes Nachbereitungsverfahren entwickelt werden. Das Verfahren muss logisch ausschließbare Klassifikationsfehler beheben.

3.4.3. Qualitätsanforderungen

In diesem Unterkapitel, werden die im Rahmen der Anforderungserhebung gesammelten Qualitätsanforderungen an alle Arbeitsergebnisse (Artefakte) zusammengefasst.

Q1: Qualität der Annotationen (Label-Qualität)

Das Labeln an sich geschieht genau so wie es die PG-MAMKS bereits definiert hat. Da die alte PG aber durch die Assessments genau wusste, welche Aktivitäten der Proband durchführt und auch über welchen Zeitraum diese durchgeführt worden sind, können wir nicht alle Vorgehensweisen beim Labeln übernehmen. Viel mehr müssen wir uns auf die Qualität der von uns erfassten Daten durch die „Activity Tracking App“ verlassen können. Jedoch hat es sich als schwierig erwiesen, die Aktivitäten zeitlich genau zu erfassen. Beim Übertragen der Label aus der App in die MAMKS Applikation muss also auf jeden Fall nachkorrigiert werden. Das heißt es muss auf die Zeitunterschiede geachtet und die jeweilige Aktivität an die tatsächlichen Zeit angepasst werden. Deshalb ist es wichtig, sich mit der App im vorhinein vertraut zu machen, um richtig zu tracken und die Zeitunterschiede minimal zu halten. So wird eine spätere Nachbearbeitung nicht unnötig kompliziert. Auch sollte das Übertragen der Aktivitäten in MAMKS zeitnah nach dem Besuch eines Probanden geschehen, um etwaige Probleme/Schwierigkeiten direkt zu beseitigen. Dies hat außerdem den Vorteil, dass sich die jeweilige Person noch genau an den Bewegungsverlauf des Besuchs erinnern kann.

Transitionen wie „Hinsetzen“, „Aufstehen“, usw. sind ebenfalls schwer zu erfassen, da diese Aktivitäten relativ schnell durchgeführt werden. Diese müssten beim Labeln händisch hinzugefügt werden, insofern sie nicht beim Tracking erfasst worden sind.

Beispiele hierfür sind:

- Der Proband steht und sitzt als nächstes. Dazwischen muss die Transition STAND-SIT manuell eingetragen werden.
- Der Proband sitzt und steht als nächstes. Dazwischen muss die Transition SIT_STAND manuell eingetragen werden.

- Der Proband liegt und setzt sich hin. Dazwischen muss die Transition LIE_SIT manuell eingetragen werden.

- Der Proband sitzt und legt sich hin. Dazwischen muss die Transition SIT_LIE manuell eingetragen werden.

Weiter unten wird noch einmal grafisch erläutert, woran sich solche Übergänge erkennen lassen.

Beschriftungen begrenzen sich zunächst auf die 15 Aktivitäten, die mit der MAMKS-App erfasst werden können. Alle anderen Aktivitäten werden nicht annotiert, oder erhalten die Markierung „Sonstiges“ - gegebenenfalls mit einem kurzen Kommentar. (Die Aktivität „Sonstiges“ ist ebenfalls mit der App vorhanden) Zu beachten ist, dass die annotierten Daten konsistent sind. Das bedeutet, es dürfen keine Daten, die als STAND markiert werden, Komponenten der Aktivität WALK beinhalten. Im Allgemeinen kann man sich bei der Trennung zweier Aktivitäten am Muster der Magnetometer-Werte orientieren.

Die Magnetometer-Werte können je nach Ort sehr unterschiedlich sein, jedoch sind beispielsweise die Übergänge von einer Aktivität A zu Aktivität B oft besser erkennbar. In der oberen Abbildung, wurde die Transition SIT_STAND in drei Unteraktivitäten gegliedert SIT, SIT_STAND und STAND, diese Unterteilung ist notwendig, um die Datenkonsistenz zu verbessern. Die annotierten Daten sind genau dann konsistent, wenn keine gleichen Datensätze mit unterschiedlichen Annotationen versehen sind. Analog soll die Transition STAND_SIT ab jetzt annotiert werden. Mit den anderen Aktivitäten wird es sich genauso verhalten. Es muss erkannt werden, wann bestimmte Aktivitäten ausgeführt wurden bzw. wo sie anfangen und wo sie aufhören. Die Magnetometer-Werte können je nach Ort sehr unterschiedlich sein, jedoch sind beispielsweise die Übergänge von einer Aktivität A zu Aktivität B oft deutlicher erkennbar. In der Abbildung 3.4, wurde die Transition SIT_STAND in drei Unteraktivitäten gegliedert SIT, SIT_STAND

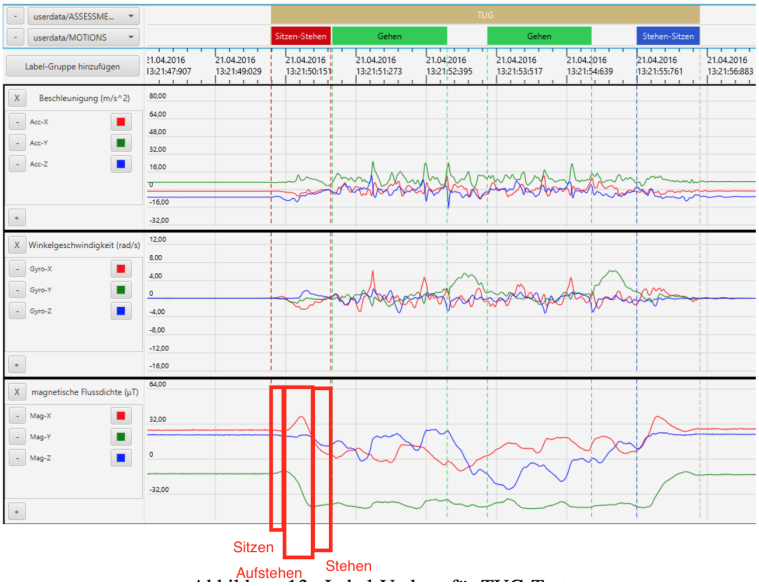


Abbildung 3.4.: SIT_STAND Annotations-Konvention (Reihenfolge der Daten Accelerometer, Gyroskop, Megnetometer)

und STAND, diese Unterteilung ist notwendig, um die Datenkonsistenz zu verbessern. Die annotierten Daten sind genau dann konsistent, wenn keine gleichen Datensätze mit unterschiedlichen Annotationen versehen sind. Analog soll die Transition STAND_SIT ab jetzt annotiert werden. Mit den anderen Aktivitäten wird es sich genauso verhalten. Es muss erkannt werden, wann bestimmte Aktivitäten ausgeführt wurden bzw. wo sie anfangen und wo sie aufhören.

Für die Nachbearbeitung der Labels ist es wichtig, dass jedes PG Mitglied, die Labeldaten auf die gleiche Weise anpasst. Um dies zu erleichtern, werden im folgenden gelabelte Datenbeispiele für jede Aktivität gegeben. Es ist jedoch zu beachten, dass den Daten je nach Datensatz und selbst innerhalb gleicher Aktivitäten Schwankungen unterliegen. Die Beispiele können daher nur als Referenz genutzt und die verschiedenen Aktivitäten nicht eindeutig beschreiben werden. Sind Daten einem Label nicht eindeutig zuordenbar, wird kein Label gesetzt.

Da die Beispielausschnitte aus den jeweiligen Datensätzen sind, ist die Benennung der Datenreihen nicht mit angegeben. Es gibt drei Datenreihen. Jeder davon zeigt die Daten von einem der Sensoren des Gürtels. Die obere Reihe zeigt die Daten des Accelerometers, die mittlere Reihe zeigt die Daten des Gyroskops und die untere Reihe zeigt die Daten des Magnetometers. Die Farben der Linien stehen für die jeweilige Achse des Sensors. Rot ist die x-Achse, grün die y-Achse und blau die z-Achse. Im folgenden werden die Linien nach der relativen Position und der Farbe beschrieben.
Allgemeine Anmerkungen:

- Es besteht die Möglichkeit, dass die Gürtel nicht ordnungsgemäß/falschherum angelegt wurden, dies ist z.B. erkennbar, wenn bei den Accelerometerdaten die x-Linie immer unterhalb der roten und blauen Linie liegt.

- Durch Veränderung der zeitlichen Auflösung sind die Grenzen der verschiedenen Aktivitäten besser zu erkennen.

Konventionen zu einzelnen Aktivitäten werden wie folgt festgelegt.

Beim Sitzen bleiben die Sensordaten zum Großteil konstant. Bei Accelerometerdaten ist die grüne Linie jeweils am Höchsten, die rote und blaue in der Mitte. Je nach Datensatz liegen die rote über/auf oder unter der blauen Linie. Bei den Gyroskopdaten sind alle drei Linien auf der gleichen Höhe. Die Magnetometerdaten unterscheiden sich in verschiedenen Datensätzen. Jedoch sind auch sie zum Großteil konstant. Die Abbildung 3.5 zeigt typische Ausprägungen der Daten.

Beim Stehen sind die Daten aller drei Sensoren, ähnlich wie beim Sitzen, zum Großteil konstant. Allerdings gibt es im Allgemeinen ein vermehrtes Rauschen, sodass die Linien nicht so gerade sind wie beim Sitzen. Die Werte der Daten sind im Mittel fast identisch mit denen beim Sitzen. In der Regel ist bei den Accelerometerdaten die blaue Linie relativ zur roten Linie höher im Vergleich zum Sitzen, d. h. war beim Sitzen wie blaue Linie

Abbildung 3.5.: Beispiel Sitzen (Reihenfolge der Daten; Accelerometer, Gyroskop, Megnetometer)

weit unter der roten Linie, so liegt sie beim Stehen dichter an der roten Linie. Wenn die Linien beim Sitzen fast auf gleicher Höhe lagen, liegt die blaue Linie beim Stehen über der roten Linie. Die Abbildung 3.6 zeigt typische Ausprägungen der Daten beim Stehen. Sie zeigt, dass die Daten vom Sitzen in ihren Mittelwerten kaum zu unterscheiden sind.

Auch beim Liegen auf dem Rücken sind die Linien annähernd gerade. In den Accelerometerdaten ist die grüne die obere Linie. Die rote Linie ist allerdings nur leicht unterhalb der grünen. Die blaue Linie ist deutlich weiter unten. In den Gyroskopdaten sind alle drei Linien auf der gleichen Höhe. In den Magnetometerdaten ist die blaue Linie die oberste, die rote die mittlere und die grüne die untere. Der Abstand dieser Linien unterscheidet sich jedoch je nach Datensatz. Die Abbildung 3.7 zeigt typische Ausprägungen der Daten beim Liegen auf dem Rücken. Sie zeigt, dass die Daten deutlich vom Sitzen und Stehen unterschiedlich sind und daher eine gute Separierbarkeit zu erwarten ist.

Beim Liegen auf der Seite ist bei den Accelerometerdaten die grüne Linie oben, darunter die blaue Linie und darunter die rote Linie. Bei den Gyroskopdaten sind die drei Linien auf der gleichen Höhe. Bei den Magnetometerdaten ist die rote Linie oben, die blaue in

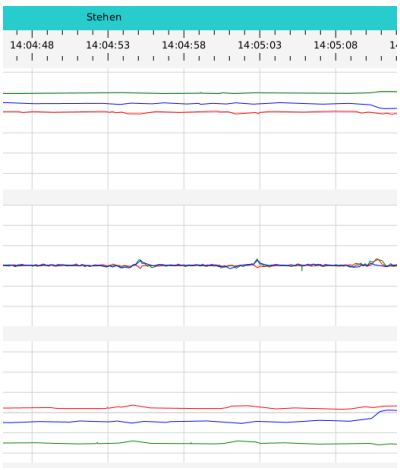


Abbildung 3.6.: Beispiel Stehen (Reihenfolge der Daten Accelerometer, Gyroskop, Megnetometer)

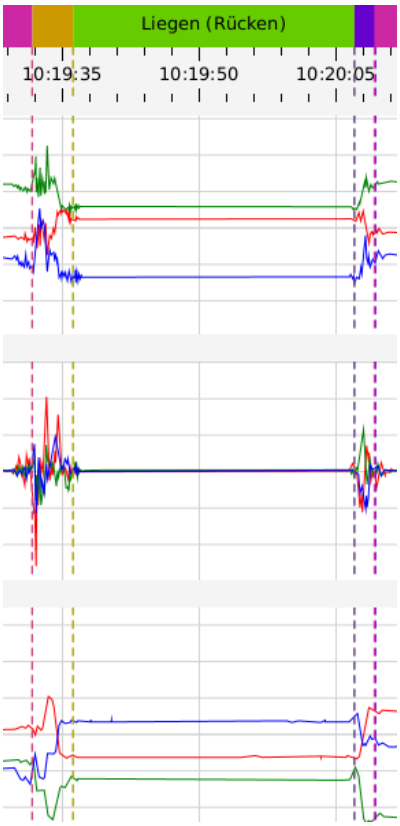


Abbildung 3.7.: Beispiel Liegen auf dem Rücken (Reihenfolge der Daten Accelerometer, Gyroskop, Megnetometer)

der Mitte und die grüne unten. Die Abstände dieser Linien unterscheiden sich jedoch in verschiedenen Datensätzen. Die Abbildung 3.8 zeigt typische Ausprägungen der Daten beim Liegen auf der Seite. Sie zeigt, dass die Daten deutlich vom Sitzen, Stehen und Liegen auf dem Rücken unterscheidbar sind. Im Vergleich zum Liegen auf dem Rücken ist besonders gut beim Wechsel zwischen den beiden Positionen zu erkennen, dass die rote und blaue Linie sowohl beim Accelerometer als auch beim Magnetometer ihre Positionen tauschen.

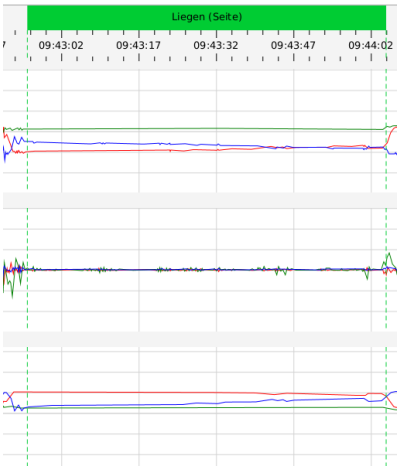


Abbildung 3.8.: Beispiel Liegen auf einer Seite (Reihenfolge der Daten Accelerometer, Gyroskop, Megnetometer)

Bei den Accelerometerdaten lässt sich beim Gehen in allen Linien ein Sinusrhythmus erkennen. Die Linien wechseln sehr schnell zwischen hohen und niedrigen Werten. Die Grundhöhe der Linien bleibt jedoch konstant. Die grüne Linie ist oben, die blaue und rote sind beide auf der gleichen Höhe darunter. Bei den Gyroskopdaten kann man ebenfalls einen Sinusrhythmus erkennen. Die Grundhöhen der Linien sind alle auf der gleichen Höhe. Bei den Magnetometerdaten ändern die Linien deutlich weniger ihre Höhen. Es sind lediglich leichte Zacken zu erkennen. Die Positionen der roten und blauen Linie sind nicht eindeutig. Jedoch ist die grüne Linie immer die unterste. Die Abbildung 3.9 zeigt eine beispielhafte Sequenz von Gangzyklen. Durch zoomen z.B. 2 sek. pro 100 Pixel lassen sich die einzelnen Peaks (Anfang und Ende des Gehens) deutlich erkennen. Ggf. kann für eine bessere Erkennbarkeit der Peaks auch die y-Skalierung angepasst werden.

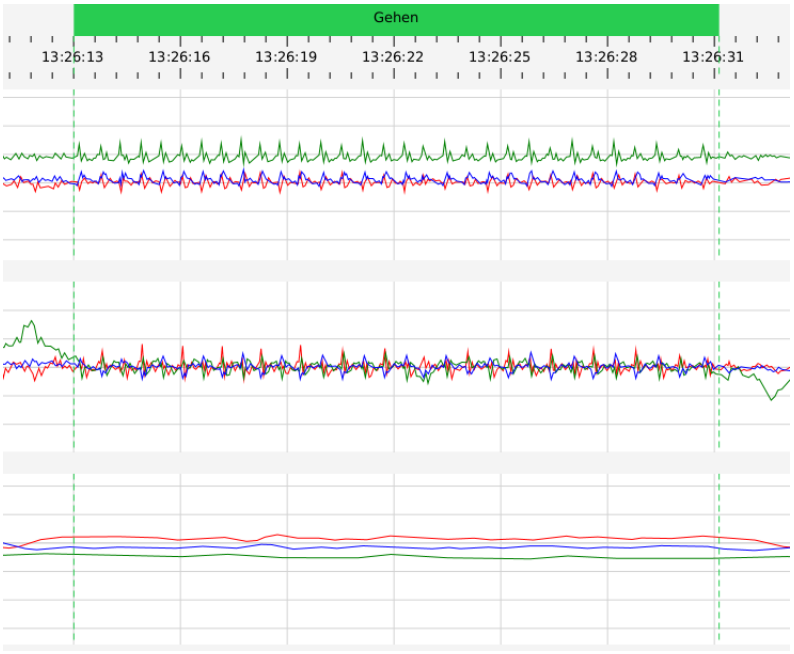


Abbildung 3.9.: Beispiel Gehen (Reihenfolge der Daten Accelerometer, Gyroskop, Megnetometer)

Beim Treppensteigen ist das charakteristischste Merkmal eine Zickzackbewegung in der roten Linie der Mangetometerdaten, welche jedoch nicht immer gut zu erkennen ist. Die Accelerometerdaten und die Gyroskopdaten ähneln sehr denen beim Gehen. Zum Teil sind die Accelerometerdaten etwas anders (langsamerer Wechsel zwischen hoch und niedrig, geringere Ausschlaghöhe), dies ist aber nicht bei allen Beispielen der Fall.
Es lässt sich visuell kein signifikanter Unterschied zum Treppe Hinuntersteigen erkennen. Lediglich in den Druck-Daten lässt ich ein leichter Anstieg beim Heruntersteigen und ein absinken beim Hinaufsteigen erahnen. Die Abbildung 3.10 zeigt eine beispielhafte Datenreihe beim Treppe Hochsteigen.

Das Treppen heruntersteigen ist bei bloßer Betrachtung dem Treppe Hinaufsteigen sehr ähnlich. Es lässt sich kein signifikanter Unterschied erkennen. Lediglich in den Druck-Daten lässt ich ein leichter Anstieg beim Heruntersteigen und ein Absinken beim Hinaufsteigen erahnen. Die Abbildung 3.11 zeigt eine beispielhafte Datenreihe beim Treppe Heruntersteigen. Es lassen sich kaum Unterschiede zum Gehen oder Treppe Hinaufstei-

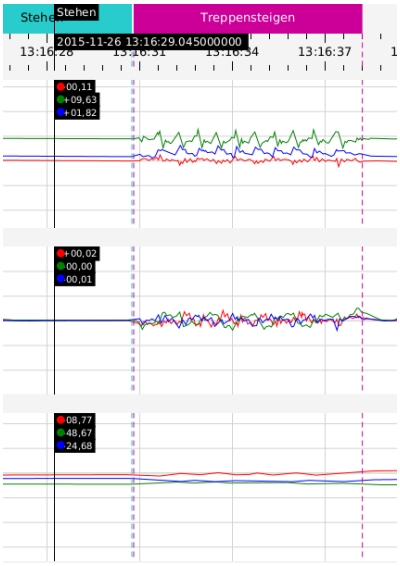


Abbildung 3.10.: Beispiel Treppensteigen nach oben (Reihenfolge der Daten Accelerometer, Gyroskop, Megnetometer)

gen festhalten.

Die Aktivität Drehen ist besonderes leicht und deutlich erkennbar. Bei den Gyroskopdaten ist ein Ausschlag der grünen Linie während der Drehung zu erkennen. Bei einer Rechtsdrehung ist der Ausschlag negativ, bei einer Linksdrehung positiv. Bei den Magnetometerdaten kommt es bei der roten und blauen Linie je nach Winkel der Drehung zu einer Veränderung der Position zum Erdmagnetfeld. Dadurch kommt es zu einem Absinken oder Ansteigen der beiden Linien. Die grüne Linie bleibt nahezu unverändert. Die Abbildung 3.12 zeigt eine beispielhafte Drehung nach rechts. Hier ist die Drehung nach rechts durch die negative Radialgeschwindigkeit auf der y-Achse (grüne Linie) deutliche sichtbar.

Das Hocken wird anhand folgender Werte festgehalten. Bei den Accelerometerdaten steigt die blaue Linie wenn die grüne Linie sinkt. Beide Linien wechseln zwischen einem hohen und einem niedrigen Niveau. Bei den Gyroskopdaten steigt und sinkt die rote Linie abwechselnd. Bei den Magnetometerdaten sinkt die blaue Linie wenn die blaue Linie aus den Accelerometerdaten steigt und umgekehrt. Die grüne Linie verhält sich

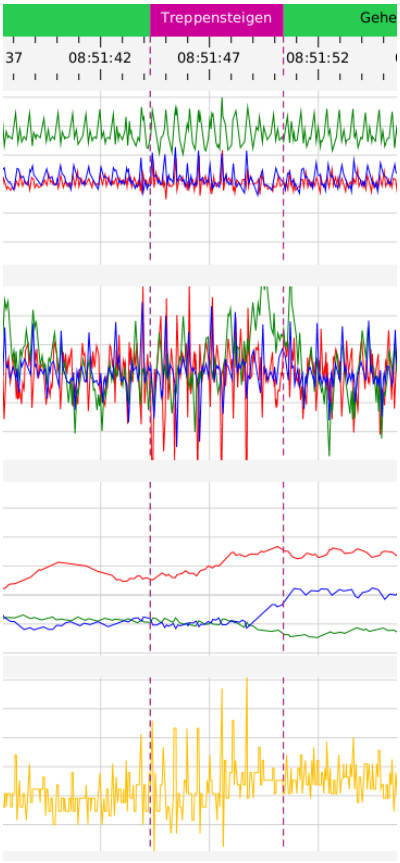


Abbildung 3.11.: Beispiel Treppensteigen nach unten (Reihenfolge der Daten Accelerometer, Gyroskop, Megnetometer)

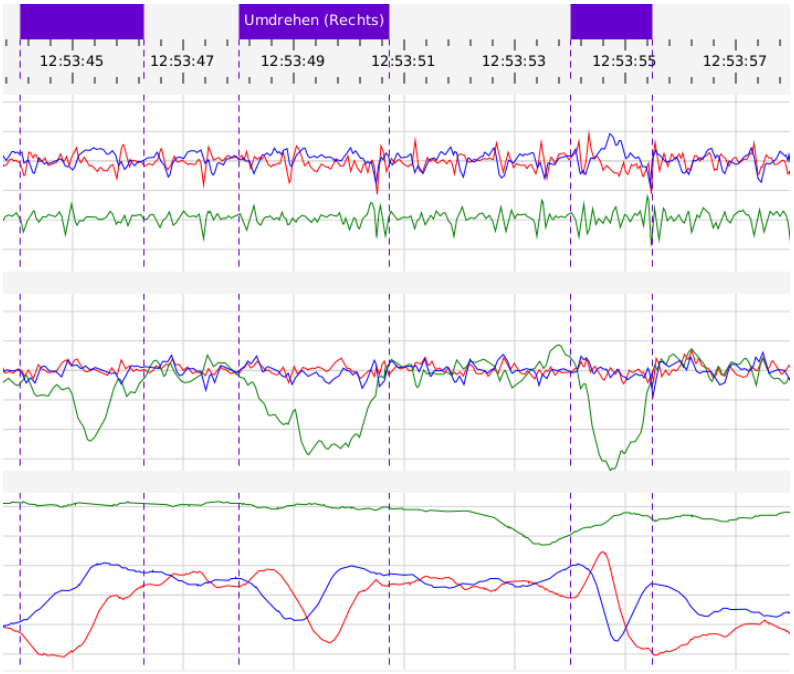


Abbildung 3.12.: Beispiel Umdrehen nach rechts im Stehen (Reihenfolge der Daten Accelerometer, Gyroskop, Megnetometer)

ebenfalls entgegengesetzt der grünen Linie aus den Accelerometerdaten. Die Abbildung 3.13 zeigt das Hocken. Es ist zu sehen, dass durch das Kippen des Beckens um die frontransversale Achse die Werte der Radialgeschwindigkeit um die x-Achse des Gyroskops, sowie die Werte der y-Achse des Accelerometers am dominantesten ausschlagen.

Beim Aufstehen erkennt man einen Abfall der Magnetometer y-Achse (dritter Graph, grün) und der Magnetometer z-Achse (dritter Graph, blau). Bei den Magnetometerdaten (zweiter Graph) erkennt man die Unterschiede am besten, wenn man die y- und die z-Achse ausblendet. Auf der x-Achse (rot) ist trotz Schwankungen insgesamt ein Abfall und ein Anstieg der Winkelgeschwindigkeit zu erkennen.
Beim Labeln ist es besonders wichtig auf die y-Achse des Magnetometers zu achten. Da dies den Höhenverlauf am besten darstellt. Als gute Hilfe lässt sich dann jedoch auch noch die Gyroskop x-Achse verwenden. Am Anfang des Aufstehens ist diese Linie auf dem Nullpunkt und auch am Ende des Aufstehens ist der Nullpunkt wieder erreicht. Der Startpunkt des Labels sollte da liegen, wo die Gyroskop x-Achse einen Nullpunkt hat und

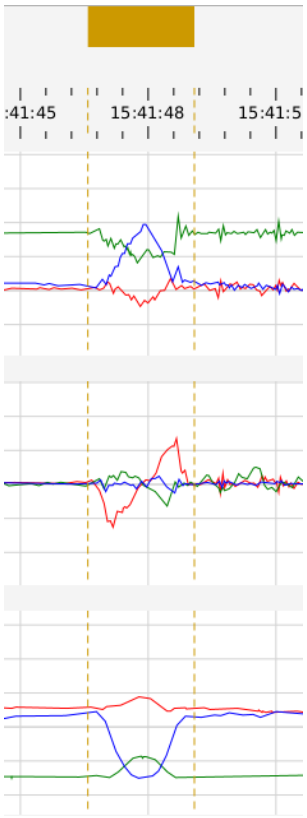


Abbildung 3.13.: Beispiel Hocken (Reihenfolge der Daten Accelerometer, Gyroskop, Megnetometer)

die y-Achse des Magnetometers seinen höchsten Punkt hat. Der Endpunkt des Labels sollte da liegen, wo die Gyroskop x-Achse erneut einen Nullpunkt erreicht hat und die y-Achse des Magnetometers seinen lokalen Tiefpunkt erreicht hat. Die Abbildung 3.14 zeigt ein Beispiel für das Aufstehen.

Abbildung 3.14.: Beispiel Aufstehen (Reihenfolge der Daten Accelerometer, Gyroskop, Megnetometer)

Beim Hinsetzen erkennt man einen Anstieg der Magnetometer y-Achse (dritter Graph, grün) und der Magnetometer z-Achse (dritter Graph, blau). Bei den Magnetometerdaten (zweiter Graph) erkennt man die Unterschiede am besten wenn man die y- und die z-Achse ausblendet. Auf der x-Achse (rot) ist trotz Schwankungen insgesamt ein Anstieg und ein Abfall der Winkelgeschwindigkeit zu erkennen. Beim Labeln ist es besonders wichtig auf die y-Achse des Magnetometers zu achten. Da diese den Höhenverlauf am besten darstellt. Als gute Hilfe lässt sich zusätzlich noch die Gyroskop x-Achse verwenden. Am Anfang des Hinsetzens ist diese Linie auf dem Nullpunkt und auch am Ende des Hinsetzens ist der Nullpunkt wieder erreicht. Der Startpunkt des Labels sollte da liegen, wo die Gyroskop x-Achse einen Nullpunkt hat und die y-Achse des Magnetometers seinen tiefsten Punkt hat. Der Endpunkt des Labels sollte da liegen, wo die Gyroskop x-Achse erneut einen Nullpunkt erreicht hat und die y-Achse des Magnetometers seinen lokalen Hochpunkt erreicht hat. Die Abbildung 3.15 zeigt ein Beispiel für das Hinsetzen.

Das Hinlegen aus dem Sitzen erkennt man bei den Accelerometerdaten an einer sinkenden

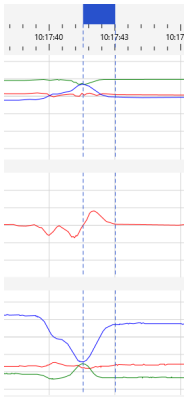


Abbildung 3.15.: Beispiel Hinsetzen (Reihenfolge der Daten Accelerometer, Gyroskop, Megnetometer)

grünen und blauen Linie. Bei den Magnetometerdaten erkennt man es an einer steigenden blauen und grünen Linie. Die Abbildung 3.16 zeigt ein Beispiel für das Hinlegen aus dem Sitzen.

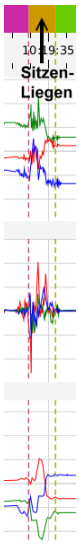


Abbildung 3.16.: Beispiel Hinlegen aus dem Sitzen (Reihenfolge der Daten Accelerometer, Gyroskop, Megnetometer)

Das Aufsetzen aus dem Liegen erkennt man bei den Accelerometerdaten an einer steigenden grünen und blauen Linie. Bei den Magnetometerdaten erkennt man es an einer sinkenden blauen und grünen Linie. Die Abbildung 3.17 zeigt ein Beispiel für das Auf-

setzen aus dem Liegen.

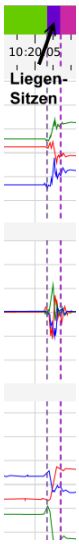


Abbildung 3.17.: Beispiel Aufsetzen aus dem Liegen (Reihenfolge der Daten Accelerometer, Gyroskop, Megnetometer)

Q2: Qualität der MATLAB-Komponenten

Die Qualitätskriterien für die in dieser Projektgruppe entwickelten MATLAB-Komponenten richtet sich nach gängigen Prinzipien der Softwareentwicklung.
Zunächst werden die grundlegenden Namenskonventionen beschrieben, anschließend werden strukturelle Konventionen mit Beachtung der Prinzipien der Abstraktion, Modularisierung, Geheimnisprinzip, Lokalität, Erweiterbarkeit und Wiederverwendbarkeit festgehalten.

Namenskonventionen

Da wir in erster Linie Softwareentwickler und eher weniger Mathematiker sind, halten wir uns an die Konventionen, die im Bereich des Software-Engineering gängig sind. Die Datentabelle hat Systemweit eine einheitliche Struktur. Sie besteht aus den Spalten AccX, AccY, AccZ, GyroX, GyroY, GyroZ, MagX, MagY, MagZ, PressureP, PressureT, La-

bel1, LabelNum1,...,LabelK,LabelNumK. ...,LabelN,LabelNumN. Die Datentabelle soll systemweit den Namen Data tragen. Folgendes Beispiel zeigt die zu verwendende Namenskonvention.

```
accX = Data.AccX
smaAccX = sum(accX(i:j))
```

Die Variablennamen sind in unterschiedlichen Kontexten zu finden. Um die Bedeutung der Variablen zu verdeutlichen, wird hier folgende Konvention vorgeschlagen.
Skalare und Hilfsvariablen: Beginn mit Kleinbuchstaben. Zusammengesetzte Worte fangen mit einem Unterstrich und Kleinbuchstaben an. Das folgende Beispiel veranschaulicht die oben beschriebene Konvention.

```
x=5;
y=42;
arg="retrain"
calc_mode=True
i=0;
t=2
```

Vektoren, Matrizen, Tabellen und andere n-Dimensionale Konstrukte:

```
X=[0 0 0; 0 1 0; 0 0 1];
Y = [0 1 2 3 4 5 6];
CovarianceMatrix = cov(A,B);
```

Mathematische Funktion: Mathematische Funktionen, beginnen mit Kleinbuchstaben. Wortzusammensetzungen Beginnen mit Unterstrich und Kleinbuchstaben. Zur besseren Nachvollziehbarkeit müssen die Worte komplett ausgeschrieben werden.

```
y = sin(x);
signal_magnetude_area = signal_magnitude_area(X)
```

Klassen

Es ist darauf zu achten, dass nach dem Prinzip der Wiederverwendbarkeit, Teilkomponenten als Klassen definiert werden. Die Klassennamen beginnen mit einem Großbuchstaben. Wortzusammensetzungen beginnen ebenfalls mit einem Großbuchstaben. Um die Performanz der Berechnungen zu steigern sollte eine Klasse, wenn sie noch nicht von einer anderen Klasse erbt, von der Klasse "handleërben. Aus Übersichtsgründen wird hier festgelegt, dass obwohl in MATLAB Mehrfachvererbungen möglich sind, diese nach Möglichkeit vermieden werden sollten. Die Membervariablen sollen auf der rechten Seite, per Tabulator getrennt, den Typen enthalten. Nach Möglichkeit sollten potenziell erweiterbare Teilkomponenten von einer übergeordneten abstrakten Klasse erben.

Die Membervariablen sollen nach Möglichkeit der CamelCase-Konvention benannt werden. Ausgenommen davon sind n-Dimensionale Konstrukte, wie Matrizen, Vektoren oder Tabellen.

```
classdef (Abstract) ClassificationAlgorithm < handle
%%CLASSIFICATIONALGORITHM Summary of this class goes here
%   Detailed explanation goes here

properties
model      classification.models.AbstractModel;
end

methods
train(Data);
predict(Data);
```

```
end

end
```

Strukturierte Datentypen (structs) sollten nach Möglichkeit nicht zur Modellierung von Teilsystemen, sondern höchstens als Hilfsstrukturen eingesetzt werden, da sie schwer wartbar oder erweiterbar sind.

Aufzählungstypen sind prinzipiell modellierbar, jedoch im mathematischen Anwendungsbereich oft zu unflexibel. Sie sind daher nur sparsam einzusetzen, zum Beispiel als Methodenparameter.

```
classdef OptimizationMode
enumeration
Entropy, CrossEntropy, GradientDiscent
end
end

...
mode = OptimizationMode.GradientDiscent;
optimizer.train(model,X,Y,mode);
...
```

Strukturkonventionen

Zur besseren Übersichtlichkeit sollen die Teilkomponenten durch das Anlegen von Ordnern, Packages und Klassenordnern strukturiert werden.

Module: Die unterschiedlichen Module der MATLAB-Toolbox werden durch unterschiedliche Ordner unterteilt. Die Ordnernamen werden mit Kleinbuchstaben geschrieben. Die Wortzusammensetzungen beginnen mit Unterstrich.

Beispiel: mamks_data_import, classificaton

Komponenten: Teilkomponenten eines Moduls sind als Packages zu definieren, um Namenskonflikte zu vermeiden. In Matlab werden Ordner mit einem +-Zeichen am Anfang als Packages interpretiert.
Beispiel:+classification
Klassen: Um die Übersichtlichkeit und Erweiterbarkeit zu gewährleisten sollen die Klassen in den dazugehörigen Klassenordnern gespeichert werden. In Matlab werden Klassenordner als @KlassenName definiert.

Ausnahmebehandlung

Matlab ist schwer zu warten und erlaubt nicht erlaubte Typen als Eingabe für Funktionen. Ausnahmen und Fehleingaben sind daher explizit zu behandeln. Alle Eingaben sind auf den erlaubten Typ, erlaubte Randbereiche/Dimensionen zu prüfen. Die fehlerhaften Eingaben müssen mit einer Exception oder mit einem AssertionError beantwortet werden.

```
% Möglichkeit 1
C = magic(5)
>> C =
17    24     1     8    15
23     5     7    14    16
4     6    13    20    22
10    12    19    21     3
11    18    25     2     9


validateattributes(C,{'numeric'},{'size',[5 4]})
>> Expected input to be of size 5x4, but it is of size 5x5.
validateattributes(C,{'numeric'},{'size',[5 5]})
```

```
>>

%Möglichkeit 2
a = 22;
assert(isa(a,'double'),'a is not type double.')
>>

% Möglichkeit 3
if ~isa(a,'double')
ME = MException('MAMKSMATLAB:classification:invalidInputType', ...
'Variable a is not a double');
throw(ME)
end
```

Kommentierung

Die Kommentierung sollen sowohl für Methoden, als auch für Klassen existieren. Dabei orientiert sich hier diese Konvention an MATLAB, um eine angemessene technische Dokumentation generieren zu können.

```
%SUM Sum of elements.
% Input, Output Parameters:
%   S = SUM(X) is the sum of the elements of the vector X. If X is a matrix,
%   S is a row vector with the sum over each column. For N-D arrays,
%   SUM(X) operates along the first non-singleton dimension.
%
%   S = SUM(X,DIM) sums along the dimension DIM.
%
%   S = SUM(...,TYPE) specifies the type in which the
%   sum is performed, and the type of S. Available options are:
```

```
%
%   'double'    -  S has class double for any input X
%   'native'    -  S has the same class as X
%   'default'   -  If X is floating point, that is double or single,
%                  S has the same class as X. If X is not floating point,
%                  S has class double.
%
%   S = SUM(...,NANFLAG) specifies how NaN (Not-A-Number) values are
%   treated. The default is 'includenan':
%
%   'includenan' - the sum of a vector containing NaN values is also NaN.
%   'omitnan'    - the sum of a vector containing NaN values
%                  is the sum of all its non-NaN elements. If all
%                  elements are NaN, the result is 0.
%
%   Examples:
%       X = [0 1 2; 3 4 5]
%       sum(X, 1)
%       sum(X, 2)
%
%       X = int8(1:20)
%       sum(X)             % returns double(210), accumulates in double
%       sum(X,'native')    % returns int8(127), because it accumulates in
%                          % int8 but overflows and saturates.
```

Namenskonventionen

Auch die Qualitätskriterien für die in dieser Projektgruppe entwickelten Python-Komponenten richtet sich nach gängigen Prinzipien der Softwareentwicklung.
Zunächst werden die grundlegenden Namenskonventionen beschrieben, anschließend

werden strukturelle Konventionen beschrieben.

Q3: Qualität der Python-Komponenten

Grundsätzlich sollte sich stets an das 'Zen of Python' gehalten werden. Es schreibt 20 Regeln fest, die beim Schreiben von Code beachtet werden sollen.

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Das ‘Zen of Python’ kann jederzeit in Python durch den Befehl ‘import this’ ausgegeben werden.

Namen

Namen sollten in jedem Fall darauf hindeuten, was das Objekt oder Parameter bewirkt. Eine selbsterklärende Namenskonvention ist obligatorisch. Nachfolgend wird außerdem zwischen vier unterschiedlichen Namensgebungen, je nach Typ, unterschieden:

1. **importierte Bibliotheken** - Für importierte Bibliotheken sollte, wenn möglich, immer ein sprechender Alias benutzt werden. Dies erleichtert das Lesen des Codes. Als Beispiel folgender Import: *import numpy as np*

2. **Klassen** - Klassennamen beginnen immer mit einem großen Buchstaben. Falls der Name aus mehreren Worten besteht, wird jeder Wortanfang groß geschrieben. Als Beispiel folgende Klassennamen: *ClassifieLabelGroups, TestSomeRandomNumbers*

3. **Objekte** - Objekte werden grundsätzlich klein geschrieben. Um eine Worttrennung deutlich zu machen, werden einzelne Wörter durch einen Unterstrich voneinander getrennt. Als Beispiel folgende Objektnamen: *heigt_of_tree, length_of_snake*

4. **Scripte** - Um Scriptdateien zu kennzeichnen sollten sie möglichst einen eindeutigen Namen ihrer Tätigkeit besitzen. Als Beispiel folgender Scriptname, inklusive Dateiendung: *Labelgenerierung.py*

Struktur

Die Struktur des Codes sollte immer so flach wie möglich ausfallen. Eine starke Verschachtelung sollte im besten Fall vermieden werden. Durch dieses Vorgehen wir ein besseres Testen des Codes gewährleistet. Außerdem kann der Code so von Außenstehenden besser gelesen werden. Das innere einer Methode ist mit einem Tabulator einzurücken. Auf Leerzeichen ist in diesem Fall zu verzichten.

Kommentierung

Falls ein Codeabschnitt nicht selbsterklärend durch die Namensgebung ist, muss er entsprechend kommentiert werden. Hierbei ist darauf zu achten eine kurze und prägnante Beschreibung zu liefern. Mehrzeilige Kommentare sollten nur verwendet werden, wenn ein Beispiel dargestellt werden möchte oder eine genauere Beschreibung zwingend notwendig ist. Es ist darauf zu achten, dass Kommentar mit %+'Space'+'Kommentar' zu beginnen. Nachfolgend ein Beispiel:

```
% Initialisierung der Bewegungsdaten
```

Beispiel eines mehrzeiligen Kommentars:

```
'''
:param x: Daten des Sensorguertels
:param window\_size: Fensterbreite mit der das ergebnis geglaettet werden soll
:param step\_size: Schrittweite mit der das Fenster verschoben werden soll
:return: y,label\_map (Sie Attributbeschreibung in AbstractClassifier)
'''
```

Q4: Qualität der Studiendurchführung (auch Pilot-Studie) mit 2 Personen

Die Durchführung der Studie muss aus folgenden Schritten bestehen.

1. Vorbereitung
    - Smartphone als auch Rechner muss mit gleichem Zeitserver synchronisiert werden
    - Sensorgürtel richtig anbringen (Siehe Rahmenbedingungen)
    - Gürtel im Stehen starten
    - Gürtel geladen (90%-100%)?

- Rohdaten ID angeben, um gelabelte Daten den Probanden eindeutig zuordnen zu können

2. Durchführung
   - Dauer: 3-5 Stunden
   - Mindestens 80% der Daten sollen gelabelt sein
   - Den Probanden zum Bewegen ermuntern (möglichst viele verschiedene Aktivitäten.)
3. Nachbereitung
   - Umwandeln in Binärdaten
   - Label prüfen, ob zeitlich korrekt mit Aktivitäten
   - Einzelne Aktivitäten nachbearbeiten

Q5: Qualität des Review-Prozesses für Tasks

- Ein Task ist dann abgeschlossen, wenn
  - die in dem Task beschriebene Aufgabe vollständig und korrekt im Sinne der Anforderungsdefinition umgesetzt wurde
  - Alle Mängel beseitigt sind, die bei Reviews gefunden wurden
- Bei Programmieraufgaben muss zusätzlich gelten, dass
  - die jeweiligen Standards (JAVA, Matlab) eingehalten wurden
  - der Code durch einen Test abgedeckt ist, wobei die Code-Coverage min. 80% betragen soll und

- weiterhin alle Tests erfolgreich ausgeführt werden

Wenn all diese Punkte gelten, dann ist ein Task fertiggestellt und kann in JIRA auf Done gestellt werden.

Weitere relevante Rahmenbedingungen

Um die Korrektheit der Klassifikationsalgorithmen und Parameterbestimmung gewährleisten zu können, muss eine fest definierte Trageposition des Gürtels eingehalten werden.

- Auf der Innenseite des Gürtels befindet sich eine Gerätenummer. Diese Nummer muss nach oben (kopfwärts) gerichtet sein. Die Gürtelschnalle sollte also, vom Probanden aus gesehen, auf der linken Seite sein.
- Die Hauptplatine mit den Messeinheiten muss mittig, direkt hinter der Lendenwirbelsäule fest anliegen. Seitlich sollte der Gürtel optimalerweise mit gleichem Abstand zum rechten/linken äußeren Beckenkamm fest anliegen.
- Die Aufzeichnung muss im Stehen beginnen.
- Die Aufzeichnung muss im Stehen beendet werden.

4. Konzept

Aus den Ergebnissen der Anforderungserhebung ist ersichtlich, dass die Klassifikationsgenauigkeit noch nicht zufriedenstellend ist und diese daher verbessert werden muss, bevor weitere analytische Schritte, wie Ganganalyse und Parameterbestimmung wie Balance und Power erfolgen können.

Da es sich herausgestellt hat, dass die Software der PG-MAMKS ein festes hierarchisches Modell, sowie eine feste Menge an Labels und Labelgruppen implementiert, ist es sehr aufwändig neue Ansätze in das monolithische Konstrukt zu integrieren. Aus diesen Gründen haben sich die Teilnehmer der PG-MAMKSFZ dazu entschieden, die vorhandene Software für Annotations-, sowie zur Visualisierung der Annotationen zu nutzen. Die Optimierung der Klassifikationsalgorithmen soll hingegen, in dafür optimaleren Umgebungen, wie MATLAB und Python erfolgen.
Insgesamt werden Konzepte zur Verbesserung, beziehungsweise Umsetzung von Labelvorbereitung, manuellen Beschriftung, Aktivitätentracking während der Seniorenstudie, Klassifizierung und Labelnachbereitung erarbeitet.

4.1. Labelvorbereitung

Das semiautomatische Labeln vor der eigentlichen manuellen Annotation hat zum Ziel, die Qualität, sowie die Geschwindigkeit der manuellen Beschriftung zu erhöhen. Eine halbautomatische Annotation hat zudem den Vorteil einer einheitlichen Weise die Daten zu beschriften, nämlich nach einem zuvor festgelegten Kriterium. In der Phase der

Anforderungserhebung und Untersuchung der annotierten Daten hat sich herausgestellt, dass die Drehungen im Stehen und im Liegen, technisch gesehen, sehr einfach zu beschreibende Aktivitäten sind. Des weiteren ist die Unterscheidung zwischen statischen und dynamischen Aktivitäten mittels Berechung der Signal Vektor Magnitude (Siehe dazu Dokumentation der PG-MAMKS) algorithmisch einfach trennbar. Folgende semiautomatische Label-Vorverarbeitungsschritte sollen daher umgesetzt werden.

- Die Berechnung der Drehungen nach links im Stehen, Drehungen nach rechts im Stehen, Drehungen nach links im Liegen, sowie die Drehungen nach rechts im Liegen werden mit hilfe der mittleren Radialgeschwindigkeit von mindestens n rad/sec , über die Dauer von m Sekunden errechnet. Dabei sollen die Parameter n und m als änderbare Parameter definiert werden. Dies ermöglicht eine anwendungsbezogene Änderung der Algorithmuseigenschaften. Beispielsweise ist es denkbar, dass manche Anwendungen eine Mindestdauer einer Drehung von 2 Sekunden benötigt, um beispielsweise kurze Drehungen unter 45 Grad auszuschließen.

- Um die einheitliche Abgrenzung einzelner Aktivitäten zu erreichen kann als erster Vorbereitungsschritt die Trennung nach dynamischen oder statischen Aktivitäten erfolgen. Dies kann erreicht werden, in dem die Merkmale wie SMV oder SE zur Beurteilung über alle Signalkanäle errechnet werden. Um kleine Schwankungen in statischen Aktivitäten auszuschließen kann eine Mindestdauer der dynamischen Aktivität festgelegt werden. Aus diesem Grund wird als Vorbereitungsschritt vor der manuellen Annotation die Trennung zwischen diesen beiden Zuständen umgesetzt. Zur Umsetzung wird das Merkmal Signalvektor Magnitude verwendet, die mit einem Faktor a, über eine Zeitdauer von m Sekunden berechnet wird. Die Parameter a und m sollen dabei frei wählbar bleiben, um sowohl die Dauer, als auch die Stärke der Magnitude anwendungsbezogen wählen zu können.

4.2. Erleichterung der manuellen Beschriftung in MAMKS

Um die manuelle Beschriftung zu verbessern, beziehungsweise zu beschleunigen wird das MAMKS-Annotationswerkzeug um weitere Funktionalitäten erweitert. Der Bedarf nach diesen Nachbesserungen war bei der Beschriftung von Assessmentsdaten nicht so deutlich, da dort alle Aktivitäten strukturiert und deutlich von einander getrennt sind. Die Pilot-Studie mit den, von den PG-Teilnehmern selbst aufgenommen Heimdaten haben die Notwendigkeit dieser Nachbesserung deutlich gemacht.

- Das Hinzufügen einer Aktivität zwischen zwei bereits vorhandenen Aktivitäten, ohne das die rechts und links liegenden Aktivitäten manuell verschoben werden müssen, muss umgesetzt werden.
- Das Hinzufügen einer Aktivität an einer unbeschrifteten Stelle durch Doppelklick. Durch das Doppelte Klicken auf ein leeres Zeitintervall soll dieser automatisch komplett ausgefüllt werden.

Zum Zweck der Datenanlyse sollen die Daten innerhalb eines MAMKS-Workspace statistisch auswertbar und diese Statistiken visuell darstellbar gemacht werden. Um die Verteilung der annotierten Daten besser beurteilen zu können, soll daher eine Möglichkeit zur Visualisierung der Verteilunegn untersucht werden. Da die Entwicklung der Klassifikationsalgorithmen extern umgesetzt werden soll, wird die Darstellung in MAMKS auf die Visualisierung in Form von Bar-Charts beschränkt. Dabei sollen die einzellnen Balken den prozentuellen Anteil der Aktivitäten im gesamten Datensatz darstellen. Eine weitere Funktionalität, die zur Beurteilung der Labelqualität, sowie zum Vergleich von Labels dienen soll, ist das Einblenden mehrere Label-Gruppen innerhalb eineses Charts. Dies ermöglicht eine Vergleichbarkeit zwischen mehreren Klassifikationsergebnissen, beziehungsweise den Annotationsergebnissen.

4.3. Aktivitätentracking

Zur Durchführung der Seniorenstudie musste eine geeignete Annotationsunterstützung entwickelt werden. Für diese Annotationsunterstützung wurden verschiedene Ansätze diskutiert:

- Nutzung der Aktivitätstagebücher der Probanden
- Dokumentation der Aktivitäten mit einem Notizbuch
- Dokumentation der Aktivitäten mit Hilfe einer App
- Dokumentation der Aktivitäten durch das Aufstellen von Kameras

Jeder dieser Ansätze hat Vor- und Nachteile. Die Aktivitätstagebücher sind leider sehr ungenau, da die Probanden aus ihrer Erinnerung heraus grob notieren, welche Aktivitäten sie durchgeführt haben. Die Daten sind nicht für das Training von Klassifizierungsmodellen brauchbar. Die Dokumentation der Aktivitäten in einem Notizbuch während eines Probandenbesuches ist schon deutlich genauer. Ein Mitglied unserer Projektgruppe beobachtet den Probanden und schreibt möglichst mit einer Uhrzeitangabe die ausgeführten Aktivitäten in dem Zeitraum des Besuches mit. Die Daten sind mit einer entsprechenden Nachbearbeitung für das Training der Modelle brauchbar. Jedoch ist kein automatisiertes Importieren der Daten in die MAMKS-Software möglich. Außerdem sind die Daten immer noch ziemlich ungenau, da der Dokumentationsweg lang ist. Der Studienleiter muss die Aktivität bemerken, die Uhrzeit in Erfahrung bringen und die Aktivität aufschreiben. In dem Zeitraum können durchaus schon weitere Aktivitäten erfolgt sein, sodass diese nicht dokumentiert werden können. Die Dokumentation mit Hilfe einer App hilft dabei den Dokumentationsweg zu verkürzen. Sie übernimmt die Aufgabe der Zeitfeststellung und des Notierens. Vom Studienleiter soll lediglich ein Klick auf einen Button nötig sein, um die Aktivität zu dokumentieren. Die Dokumentation mit Hilfe einer App wird durch die Zeitersparnis genauer sein,

als die Dokumentation mit einem Notizbuch. Auch kürzere Aktivitäten können so dokumentiert werden. Außerdem ist ein automatisiertes Übertragen der Daten in die MAMKS Software denkbar. Jedoch sind auch bei diesem Ansatz Ungenauigkeiten in den dokumentierten Daten ein Problem. Der Studienleiter hat eine gewisse Reaktionszeit, die vergeht, bevor die Aktivität eingetragen ist. Außerdem kommt es auf die Fähigkeiten der Person im Umgang mit der App an, wie schnell diese den passenden Button findet. Diese drei Ansätze haben den Vorteil, dass sie mit vergleichsweise geringem Aufwand verbunden sind und flexibel an jedem Ort, der während der Studie besucht wird einsetzbar sind. Der vierte Ansatz ist da anders. Das Aufstellen von Kameras in der Wohnung der Probanden liefert zwar die genauste Dokumentation von Aktivitäten, aber auch nur für Aktivitäten, die der Proband in seinem Zuhause ausführt. Verlässt er das Haus kann nichts mehr dokumentiert werden. Außerdem ist es aus Gründen der Privatsphäre wahrscheinlich, dass nur wenige Probanden sich zu dieser Art der Studiendurchführung bereit erklären. Hinzu kommt ein hoher Aufwand der Vorbereitung der Studie, die durch das Aufstellen der Kameras entsteht.
Die PG MAMKSFZ hat von diesen vier Möglichkeiten die Dokumentation mit Hilfe einer App ausgewählt, da diese von den ersten drei Möglichkeiten die genauste Dokumentation und eine Möglichkeit des automatisierten Übertragens der Daten in die MAMKS Software zur Nachbesserung erlaubt. Die Aufstellung von Videokameras wurde aus Gründen des Aufwandes und der Privatsphäre ausgeschlossen.

Als nächster Schritt wurde der Benutzungskontext dieser App untersucht. Sie soll von Mitgliedern der PG MAMKSFZ genutzt werden. Die Zielgruppe besteht demnach aus Informatik- und Wirtschaftsinformatikstudenten, die im Umgang mit technischen Geräten geübt sind. Der Einsatz der App soll bei den Probanden zu Hause erfolgen. Dies bedeutet wechselnde Einsatzorte, die sowohl in Gebäuden als auch draußen sein können. Während des Einsatzes muss der Probandenbesuch gestaltet werden. Es muss sich mit den Probanden unterhalten werden. Teilweise unterstützt der Studienleiter den Probanden bei seinen alltäglichen Aufgaben, zum Beispiel Taschen tragen beim Einkaufen. Der

Studienleiter ist also möglicherweise zwischenzeitlich von der Benutzung der App abgelenkt. Das Ziel der App ist es Aktivitäten, die der Proband während der Studie ausführt zeitgenau zu dokumentieren. Aus diesem Benutzungskontext ergibt sich, dass die App eine Dokumentationsfunktion erfüllt. Sie muss schnell und einfach bedienbar sein muss. Außerdem muss das Layout gut erkennbar sein, sodass auch wechselnde Lichtverhältnisse kein Problem darstellen. Auf dieser Grundlage wurde von der PG MAMKSFZ ein erster Designentwurf erstellt. Dieser ist in Abbildung 4.1 zu sehen. Es soll für jede zu

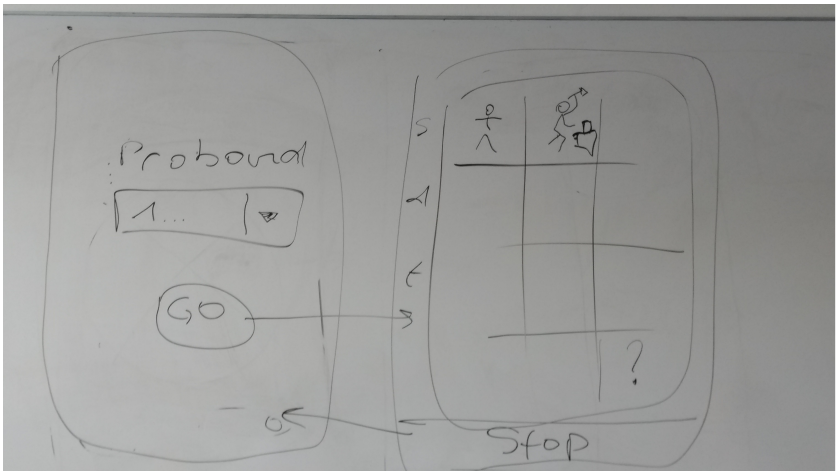


Abbildung 4.1.: Erster Designentwurf

trackende Aktivität einen Button geben. Für eine bessere Übersicht sollen diese nach statischen Aktivitäten, dynamischen Aktivitäten und Transitionen sortiert werden. Es soll möglich sein eine Probanden ID einzugeben, sodass die Daten den Sensorgürteldaten zugeordnet werden können.

Die App soll dem Studienleiter ermöglichen, eine neue Probanden ID einzutragen. Die Probanden erhalten eine spezielle ID, sodass die aufgenommenen Daten nicht mit ihren persönlichen Daten in Verbindung gebracht werden kann. Der Studienleiter soll Aktivitäten dokumentieren können. Außerdem soll die App dem Studienleiter ermöglichen, den Aufenthaltsort des Probanden zu dokumentieren. Dabei soll zwischen Drinnen, Draußen und Zuhause unterschieden werden. Die Dokumentation dieser Informationen soll im weiteren Verlauf der VERSA Studie weitere Analysen der Sensorgürteldaten ermöglichen.

Außerdem soll es dem Studienleiter möglich sein während der Studie zu kontrollieren, welche Aktivitäten wie oft und wie lange ausgeführt wurden, um Entscheidungen über den weiteren Verlauf der Studie treffen zu können. Ein Datenanalyst soll mit die Daten in die MAMKS Software übertragen können. Diese Aufgaben sind in dem Use Case Diagramm in Abbildung 4.2 dargestellt. Der Studienleiter und der Datenanalyst können die gleiche Person sein, dies muss jedoch nicht der Fall sein. Die zu erfassende Aktivitäten

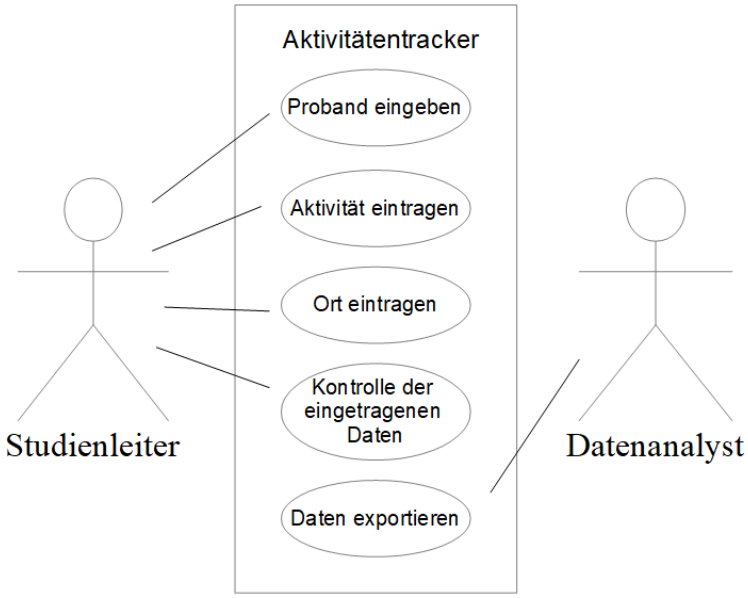


Abbildung 4.2.: Use Case Diagramm

sind Stehen, Sitzen, Liegen auf dem Rücken, Liegen auf der Seite, Aufstehen, Hinsetzen, aus dem Liegen Hinsetzen, aus dem Sitzen hinlegen, Hocken, Vorwärtsgehen, Rückwärtsgehen, Umdrehen, Treppeheruntersteigen, Treppehinaufsteigen, Springen und Sonstiges. Diese Aktivitäten ergeben sich aus den getrackten Aktivitäten, die die PG MAMKS genutzt hat und einer Befragung unserer Auftragsgeberin Sandra Hellmers.
Als Plattform wurde Android ausgewählt, da ein Großteil der Teilnehmer der PG MAMKSFZ das Betriebssystem selbst verwendet. Außerdem war die Entwicklung einer App für dieses Betriebssystem einem der Mitglieder bereits bekannt, sodass eine geringe-

re Einarbeitungszeit notwendig war. Aufgrund der Anzahl der zu trackenden Aktivitäten wurde sich für den Einsatz von Tablets entschieden. Durch den größeren Bildschirm sind die Buttons dort besser zu erkennen.

4.4. Verbesserung der Klassifikation

Zur Verbesserung der Klassifikationsergebnisse, werden die aus den Interview ersichtlichen Problembereiche untersucht. Zunächst wird die Anzahl und die hierarchische Struktur der Label betrachtet und überarbeitet. Anschließend werden weitere Klassifikationsalgorithmen untersucht und zur weieteren Optimierung ausgewählt.

4.4.1. Revision der Annotationshierarchie

Die von der PG-MAMKS ausgearbeitete Label-Struktur soll überarbeitet werden. Die bisherige Struktur benutzt konzeptionell die Terminologie aus dem Bereich der endlichen Automaten. Zudem ist sie so umgesetzt, dass nur genau ein Zustand zu einem Zeitpunkt gelten kann. Wie die Beobachtung der PG-MAMKSFZ gezeigt haben, entspricht diese Vorstellung nicht den physiologischen Bewegungsabläufen eines Menschen. Beispielsweise ist es möglich, dass sich ein Mensch im Stehen, Sitzen, Liegen oder im Gehen umdreht. Datentechnisch entspricht das Umdrehen einer kontinuierlichen Drehung um die Y-Achse des Gyroskops. Hier wird noch ein weiteres Problem deutlich. Obwohl das Umdrehen im stehen und im liegen aus technischer Sicht sehr ähnlich sind, werden in der MAMKS-Software hierfür unterschiedliche Label TURN_AROUND_LEFT, TURN_AROUND_RIGHT, LIE_ON_BACK_LIE_ON_SIDE benutzt. Aus physiologischer Sicht erscheint es nicht sinnvoll, die dynamischen Bewegungen zusätzlich in Dynamisch und Transition zu untergliedern. Auch hier sind Überschneidungen, wie beispielsweise Hinsetzen im Gehen oder ein Übergang aus dem Aufstehen in Gehen denkbar. Es ist wahrscheinlich, dass bei Betrachtung von gut strukturierten Assessmentsabläufen

diese Probleme nicht sichtbar werden. Aus den oben beschriebenen Gründen, wird die Annotationsstruktur für die PG-MAMKSFZ wie folgt angenommen.
Die statischen Zustände Stehen, Sitzen, Liegen werden einzeln betrachtet. Die dynamischen Aktivitäten Gehen, Treppesteigen (hoch und runter), Aufstehen, Hinsetzen, gehören zur Gruppe der dynamischen Aktivitäten. Da die Aktivitäten Umdrehen im Stehen, im Liegen und im Sitzen parallel zu anderen Aktivitäten oder Zuständen auftreten können, werden diese separat betrachtet und annotiert.

4.4.2. Revision des hierarchischen Klassifikationsmodells

Nach der Betrachtung des hierarchischen Annotationsmodells, erscheint auch die hierarchische Struktur des Klassifikationsmodells fraglich. Das hierarchische Modell der PG-MAMKS ist so aufgebaut, dass ein Boosted Decision Tree zuerst die Zustände STATIC, DYNAMIC, TRANSITION klassifiziert. Anschließend werden drei weitere Modelle - hier auch Boosted Decision Trees - benötigt um die jeweiligen Zustände zu klassifizieren. Struktur bedingt hängt die Genauigkeit der untergeordneten Modelle von der Genauigkeit des übergeordneten Modells ab. Wie bereits erlätert, ist die Umstrukturierung des Modells in MAMKS-Umgebung relativ umständlich. Da die PG-Teilnehmer der PG-MAMKSFZ sich für eine separate Label-Nachbereitung entschieden haben, werden zunächst einzelne Modelle angelernt, die für sich alle Aktivitäten klassifizieren und deren Ergebnisse anschließend im Nachbereitungsschritt verbessert werden.
WIe aus der Dokumentensichtung ersichtlich war, empfielen die PG-Teilnehmer den Einsatz von weiteren Algorithmen. Es soll daher untersucht werden, welche Algorithmen potenziell gut geeignet sind. Zur Untersuchung wurden zunächst folgende Algorithmen untersucht.

- Bagged Decision Trees
- Boosted Decision Trees

- K-Nearest Neighbor
- Lineare und Quadratische Diskriminanzanalyse
- Faltende Neuronale Netze

Alle Algorithmen sollen auf individuelle Optimierungsparameter untersucht werden. Individuell bedeutet hier, dass beispielsweise der K-Nearest Neighbor-Algorithmus die Parameter K, Distanzmetrik und Gewichtung besitzt, die zur Verbesserung der Klassfikationsergebnisse optimiert werden können. Auch andere Algorithmen haben ihre indiviuellen Parameter, die im Optimierungskapitel näher erläutert werden.
Eine weitere Optimierungsmöglichkeit ist die Untersuchung unterschiedlicher Merkmale, wie Signalenergie, Singnal Vektor Magnitude, Entropie, oder Mittelwert und Varianz pro Fenster. Eine ausführliche Beschreibung dieser Merkmale ist in der Dokumentation der PG-MAmks zu finden. Weitere denkbare Merkmale können aber auch die aktuelle Änderungsrate in den Daten sein, die beispielsweise in den Gyroskopdaten die radiale Beschleunigung beschreiben würde. Auch in den Magnetometerdaten, die naturgemäß von Ort zu Ort unterschiedlich sien, können durch die Umrechnung zur Änderungsrate mit hilfe des Diferenzquotienten standardisiertwerden. Diese Merkmale könne auf Ihre eignung in Verbindung mit unterschiedlichen Modellen untersucht werden. Eine Ausnahme ist hier das Faltende neuronale Netz, da es die Merkmale automatisiert generiert.
Ein weiterer Optimierungsbereich, der für alle Algorithmen untersucht werden sollte, ist die Notwendigkeit der fensterbasierten Klassifikation. Konkret soll untersucht werden, ob punktbasierte Klassifikation mit anschließenden Glättung dem fensterbasierten Ansatz, wie er in der PG-MAMKS umgesetzt worden ist, überlegen ist.

4.4.3. MATLAB-Toolbox und Pyhton-Module

Zur Anbindung von externen Algorithmen werden Möglichkeiten zur Ausführung von MATLAB-Skripten integriert. Des Weiteren wird eine MATLAB-Toolbox zur Klas-

sifizierung von Daten des Sensorgürtels entwickelt. Innerhalb der MATLAB-Toolbox werden verschiedene Algorithmen zur Berechnung von Features, wie Signalenergie, Signelvektor Magnitude oder zur Berechnung der Änderungsraten in den Signaldaten untersucht. Des weiteren sollen in der MATLAB-Toolbox Werkzeuge zum Enlesen der Daten aus einem MAMKS-Workspace implementiert werden. Nach erfolgter Berechnung sollen entsprechende Daten (Label oder Signaldaten), in der für die MAMKS-Software kompatiblen Form zurückgeschrieben werden, so dass sie schließlich in der MAMKS-Software visuell dargestellt werden können.

Zur Anbindung von externen Algorithmen aus Python wird ebenfalls eine Möglichkeit zur Anbindung untersucht. Für die Entwicklung und Optimierung von Algorithmen in Python wird analog zu MATLAB-Toolbox ein Package entwickelt, das per Command Line Interface die Daten des MAMKS-Workspace auswertet und kompatible Ergebnisse, entweder als Binärdaten oder als Label zurückschreibt.

Konzeptionell sollen beide Subsysteme die gleiche Funktionalität aufweisen. Abbildung 4.3 visualisiert den grundlegenden Aufbau.

Die Abbildung 4.3 ist ein UML-Klassendiagramm. Zu sehen sind vier Teilkomponenten. Um Daten aus dem MAMKS-Workspace einlesen zu können, dient die Komponente BinReader, die eine einheitliche Datenstruktur zur weiteren Klassifizierung bereitstellt. Dieses Datenstruktur unterscheidet sich je nach Umgebung und wird hier daher als die abstrakte Klasse Data dargestellt. Beispielsweise kann es in MATLAB sinnvoll sein die Daten als Datentyp Table zu speichern. In Python wäre diese ein mehrdimensionales Numpy-Array.
Nach der Klassifizierung müssen die Annotationen, in einem für das MAMKS-Frontend kompatiblen Format exportiert werden. Hierzu wird die Komponente LabelExporter entwickelt. Sie erwartet drei Argumente. Zu einem werden die zu exportierenden Annotationen erwartet, zum anderen sind der Pfad zum Zielordner, sowie der Name der

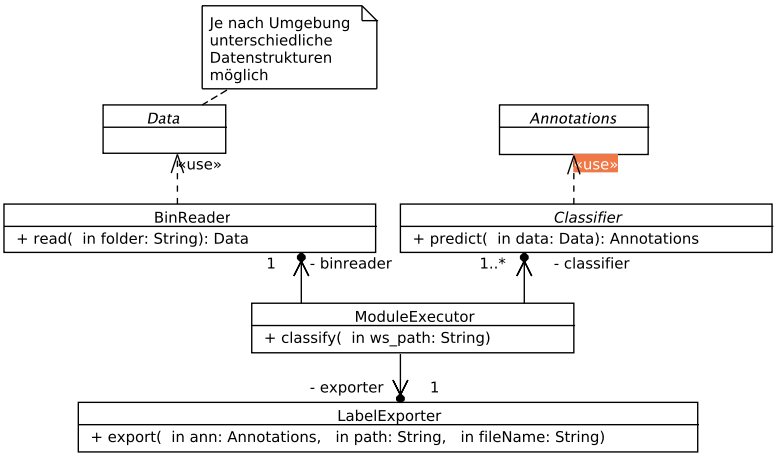


Abbildung 4.3.: Konzeptionelle Struktur eines externen MAMKSFZ-Systems zur Klassifizierung von MAMKS-Daten

Label-Datei notwendig.

Die Abstrakte Klasse Classifier repräsentiert ein antrainiertes Modell, das zur Klassifizierung von Daten eingesetzt werden soll. Die jeweilige Umsetzung, kann sich je nach Modell stark unterscheiden und wird daher erst im Implementierungskapitel näher erläutert. Nach außen sichtbar ist dagegen eine einheitliche Klassifizierungsmethode predict(in data: Data): Annotations. Diese soll eine Datenstruktur Annotations liefern. Diese muss für jedes Signal in den Daten eine Aktivität voraussagen.

Der ModulExecturor ist eine Klasse, die zur Ausführung des Klassfikators dient. Sie nutzt alle anderen drei Subkomponenten, um diese Aufgabe zu erledigen. Die Abbildung 4.4 veranschaulicht den Klassifikationsschritt. Die Abbildung 4.4 zeigt die Klassifizierung in Form eines UML-Aktivitätsdiagramms. Im ersten Schritt 'Lese MAMKS-Workspace Ordner' wird der Workspace untersucht und die zum Einlesen der Daten notwendigen Informationen gesammelt. Danach wird in einer Schleife jeder Datenordner eingelesen 'Lese MAMKS-Daten aus aktuellen Ordner'. Mit dem aktuell gesetzten Klassifikator werden die Daten auf Aktivitäten untersucht. Sobald die Annotationen bereit stehen, werden diese mittels LabelExporter in den jeweiligen Datenordner zurückgeschrieben 'Exportiere Annotationen als MAMKS-Label'.

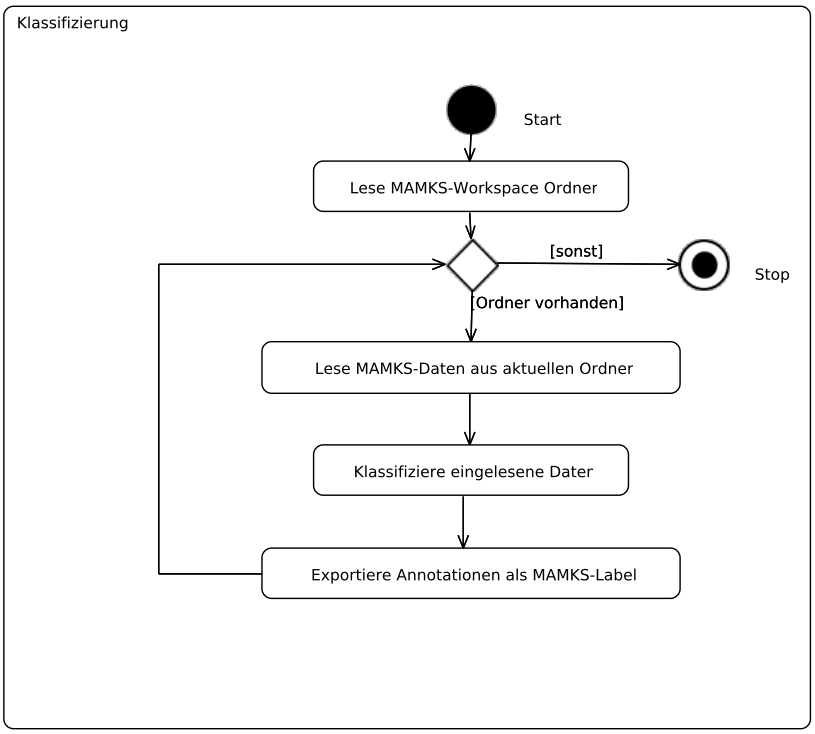


Abbildung 4.4.: Klassifizierung der Daten innerhalb der classify-Methode

4.5. Labelnachbereitung

Um die Ergebnisse der Nachbereitung, beziehungsweise der Postprocessing-Aktivitäten visuell besser beurteilen zu können, werden diese in MAMKS-Frontend umgesetzt. Zur Disskussion stehen, nach der Befragung der Auftraggeber, zunächst zwei grundlegen Ansätze.

- Der Mehrheitsentscheid soll die Ergebnisse mehrerer Klassifikatoren vereinen. Zunächst wurden drei Möglichkeiten der Umsetzung identifiziert. Die erste Möglichkeit ist es, mehrere Ergebnisse mit einander punktweise zu vergleichen. Die Entscheidung wird gefällt, wenn mehr als p Prozent der Ergebnisse eine die gleiche Aussage treffen. Ist die höchste Prozentzahl kleiner als p, so wird die Voraussage als UNKNOWN markiert. Der Parameter p soll dabei vom Benutzer frei wählbar sein. Die oben beschriebene Möglichkeit wird im Folgenden als der vertikale Mehr-

heitsentscheid bezeichnet.

Die zweite Möglichkeit des Mehrheitsentscheides ist die Betrachtung der Voraussagen eines Klassifikator über einen Zeitintervall t. Das Prinzip ist analog zum vertikalen Mehrheitsentscheid, wobei die am meisten vorkommende Voraussage für das gesamte Intervall ausgewählt wird. Auch für diese Möglichkeit soll der Parameter t vom Benutzer frei wählbar sein. Diese Möglichkeit wird als der horizontale Mehrheitsentscheid bezeichnet.

Die dritte Möglichkeit ist die Kombination aus vertikalen und horizontalen Mehrheitsentscheid. Hierbei soll sowohl die Anzahl der Klassifikatoren, des Parameters p, sowie des Intervalls t frei wählbar sein.

- Eine Entscheidung anhand von logischen Plausibilitätsprüfungen soll ebenfalls realisiert werden. Die Idee besteht darin, die Aktivitäten herauszufiltern, die aus logischen Gründen an der fraglichen Stelle nicht auftreten dürften. Dabei ist darauf zu achten, dass die Logikprüfung nicht spekulativ ist. Die Definition der Logikregeln ist daher relativ schwierig. Zur Prüfung der Logik wurden zunächst fünf logische Merkmale einer Aktivität identifiziert.

  - Mindestdauer einer Aktivität. Hier ist es möglich eine Aktivität, die aufgrund ihrer zu kurzen Dauer nicht logisch erscheint zu entfernen, beziehungsweise zu ersetzen.

  - Maximale Dauer einer Aktivität. Aktivitäten wie Hinsetzen oder Aufstehen, können logischerweise nicht über eine Dauer von mehr als 3-5 Sekunden andauern.

  - Valide Vorgänger einer Aktivität. Diese Prüfung ermöglicht die Prüfung auf sequentielle Richtigkeit der Reihenfolge von Aktivitäten. Es muss überprüft werden, ob Regeln gefunden werden können, die ausschließen können, ob die aktuelle Aktivität oder die vorangegangene die logische Reihenfolge einhält.

- Valide Nachfolger einer Aktivität. Diese Regel ist analog zur Prüfung auf valide Vorgänger.

5. Biomechanik

Die Erfahrungen aus der Studiendurchführung und der anschließenden Nachbereitung der Daten haben gezeigt, dass ohne eine genauere Vorstellung von physiologischen Bewegungsabläufen am Becken während einer Aktivität, die Nachbereitung der Daten sich als schwierig erweist. Aus diesem Grund haben sich die Projektgruppenteilnehmer entschieden, als Grundlage für die Labelnachbereitung, die biomechanischen Beschreibungsmittel zu nutzen, um die Beckenbewegungen zu analysieren. Diese Bewegungsabläufe sollen anschließend auf die Daten des Sensorgürtels abgebildet werden.
In der Literatur wird zur Beschreibung von Körperbewegungen oft die Rotation, sowie die Position des zu beschreibenden Körperteils verwendet. Geschwindigkeiten und Beschleunigungen, sowohl radial als auch linear, werden hingegen oft nur qualitativ beschrieben. Beispielsweise deutet eine Beschreibung der Schwungphase beim Gehen zu ihrem Beginn eine positive lineare Beschleunigung in der sagitto-transversalen Achse, was in der Z-Achse des Accelerometers eine positive Beschleunigung sein müsste. Die Daten des Sensorgürtels beschreiben jedoch die Winkelgeschwindigkeit, sowie die lineare Beschleunigung des Beckens quantitativ. Die Herausforderung besteht daher darin diese qualitativen Beschreibungen auf die quantitativen Werte des Sensorgürtels abzubilden.

5.1. Biomechanische Analyse - Stehen/Sitzen

Beschreibung Stehen

Die Hüftposition ist beim Stehen von verschiedenen Faktoren abhängig. Da beim Stehen oft ein Standbein mehr als das andere Belastet wird ändert sich dadurch bereits der Winkel (siehe Abb. 5.1, mitte und rechts). Steht die Person hingegen mit einer gleichen Lastverteilen auf beiden Beinen wie in Abbildung 5.1, links, zu erkennen, so ist die

Ausrichtung weitestgehend waagerecht, insofern beide Beine in etwa die gleiche Länge besitzen.

Das Becken hat während des Stehens leichte Schwingungen insbesondere auf der X und Y- Achse zu verzeichnen. Diese leichten Bewegungen werden z.B. durch ein anlehnen verringert.

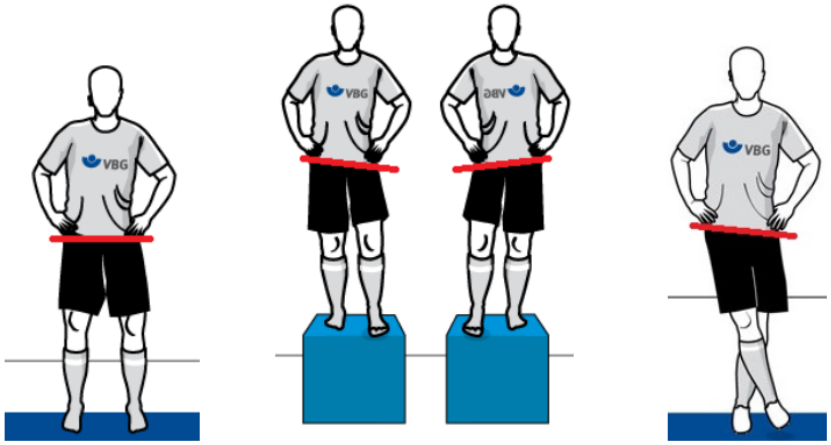


Abbildung 5.1.: Unterschiedliches Stehen

Beschreibung Sitzen

Beim Sitzen ist das Verhalten der Hüfte je nach Sitzposition unterschiedlich. Wie auf Abbildung 5.2, links, zu sehen, ist eine leichte Beugung nach vorne ersichtlich. Entsprechend dieser Position ist der Sensorgürtel etwas nach vorne geneigt. In Abbildung 5.2, mitte, ist eine aufrechte Sitzposition eingenommen und die Hüfte ist gerade ausgerichtet. In Abbildung 5.2, rechts, gleicht die Sitzposition mehr einem liegen. Dieses Verhalten ist vor allem in einem Sessel zu beobachten. Der vom Proband getragene Sensorgürtel weist eine nach hinten gekippte Position auf.

Zusätzlich zu den zuvor aufgelisteten Sitzpositionen kommt eine Variation in der horizontalen Lage in Betracht. Meistens sitzt der Mensch nicht mit einer geraden Hüfte, sondern stützt sich zu einer bestimmten Seite. Diese wird auch häufig während des Sitzens gewechselt.

Differenzierung der Aktivitäten Stehen und Sitzen

Abbildung 5.2.: Unterschiedliches Sitzen

Die beiden Aktivitäten weisen eine sehr identische Sensorenaktivität auf. Jedoch gibt es einige Indizien, mit denen Stehen und Sitzen unterschieden werden kann. Grundsätzlich sollte die zuvor gegebene Aktivität betrachtet werden. Falls ein Gehen ersichtlich ist und keine Transition zum sitzen erfolgt ist, muss es sich um Stehen handeln. Durch diesen Logikfilter lassen sich bereits gute Erfolgsquoten bei einer Unterscheidung erzielen.
Des Weiteren ist das Sitzen eine ruhigere Aktivität. Es wird nicht so oft die Position geändert wie beim Stehen. Bei der Steh-Aktivität wird zum Beispiel des Öfteren das Standbein geändert. Dieser Wechsel wird deutlich durch die Beschleunigungssensoren wahrgenommen. Beim Sitzen besteht ebenfalls die Möglichkeit eines Umpositionierens, aber dies kommt deutlich weniger häufig vor. Durch die Fixierung des Beckens auf einer starren Fläche kommt es somit zu weniger Bewegungen.
Denkbar ist auch eine Veränderung des Barometers, da sich das Becken beim Sitzen und Stehen in unterschiedlichen Höhen befindet kann. Diese Änderungen konnten wir mit dem im Sensorgürtel verbauten Barometer allerdings nicht nachweisen. Wir vermuten, dass der Sensor keine granularen Höhenunterschiede bemessen kann. Ein weiterer Differenzierungspunkt der beiden Aktivitäten wird an nachfolgender Grafik gut ersichtlich.

Wie in Abbildung 5.3 markiert, besitzen die X- und Y-Achse beim Stehen nahezu identische Werte und haben sich deutlich angenähert. Beim Sitzen hingegen ist ein deutlicher Abstand zwischen den beiden Achsen zu erkennen. In der Praxis ist dieser Unterschied nicht immer so klar ersichtlich wie in der vorigen Abbildung. Es sollte sich also nicht nur auf diese Indizien verlassen werden. Es sind also nach Möglichkeit immer mehrere

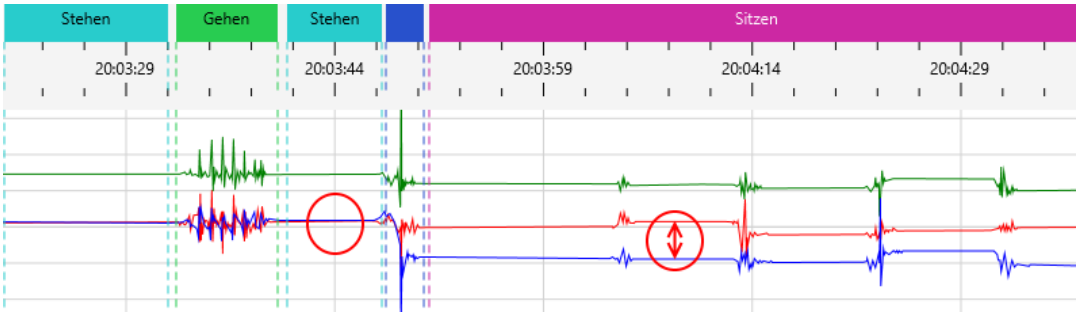


Abbildung 5.3.: Unterschiede zwischen Sitzen und Stehen

Faktoren zu betrachten und vor allem auf die Aktivitätslogik zu achten.

Zusätzliche Informationen In der nachfolgenden Abbildung 5.4 sind zum Nachvollziehen noch manuelle Kippbewegungen mit dem Gürtel händisch erfasst. Die Label zeigen an, um welche Bewegungen es sich handelt.

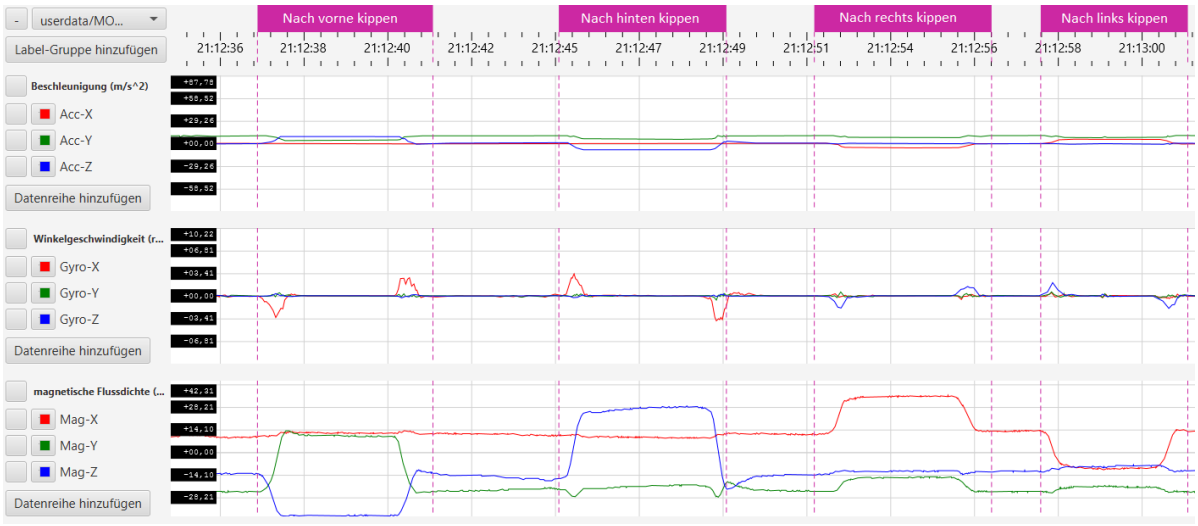


Abbildung 5.4.: Kippverhalten

5.2. Biomechanische Analyse - Treppensteigen

Im folgenden Abschnitt wird auf die Aktivität Treppensteigen näher eingegangen. Insbesondere auf die verschiedenen Phasen und Bewegungsmechaniken beim Treppen auf/- und absteigen.

Treppensteigen Gangzyklus

Grundsätzlich ist das Treppensteigen in 2 Phasen einzuteilen, der Grundhaltung und der Schwungphase. Die Grundhaltung ist beim Treppensteigen in drei weitere Teilphasen unterteilt.

1. Gewichtsaufnahme (Von der Gewichtsaufnahme vom Körper, zur optimalen Position zum Treppenaufstieg)
2. Aufstieg (Der grundsätzliche Treppenaufstieg, von einer Stufe zur nächsten)
3. Weiterer Aufstieg

Abbildung 5.5.: Grundhaltung beim Treppensteigen: Von der Anfangsposition zum weiteren Aufstieg [1]

Dabei wird während der Grundhaltung die Hüfte zwischen 60-30° gebeugt. Die Schwungphase ist in zwei weiteren Teilphasen unterteilt.

1. Fuß Freiheit (Das Bein wird gehoben zur nächsten Stufe, während der Fuß Frei steht zum nächsten Schritt)

2. Fußplatzierung (Fuß setzt auf)

Abbildung 5.6.: Schwungphase beim Treppensteigen [1]

Bei der Schwungphase entsteht eine Hüftbeugung zwischen 10-20° bis 40-60° und eine Hüftstreckung von 40-60°.

Treppensteigen - MAMKS

In der Abbildung 5.7 ist das Treppensteigen dargestellt. Dabei entspricht die oberste Linie die der Beschleunigung, die mittlere vom Gyroskop und die unterste der vom Magnetometer. Die Beschleunigung und Gyroskopdaten ähneln dabei sehr der Aktivität Gehen. Es ist allerdings immer ein signifikanter Abtieg der X-Achse beim Magnetometer zu erkennen.

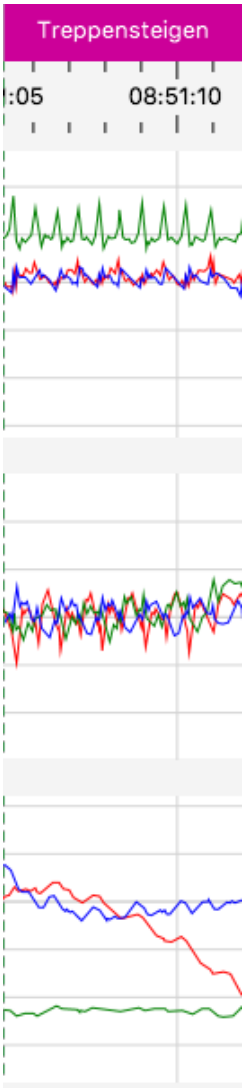


Abbildung 5.7.: Treppensteigen in der MAMKS Software. Oberste Linie: Beschleunigung. Mittlere: Gyroskop. Unterste: Magnetometer

Biomechanische Analyse - Treppenabstieg

Die Grundhaltung beim Treppenabstieg ist wiederum unterteilt in 3 Teilphasen. Dabei entsteht eine Hüftbeugung von 20-30° bis 5°. Bei dem kontrollierten Abstieg wird die Hüfte von 5° bis 10-15° gebeugt.

1. Gewichtsaufnahme

2. Weiterer Abstieg

3. Kontrollierter Abstieg (Von einer Stufe zur nächsten)

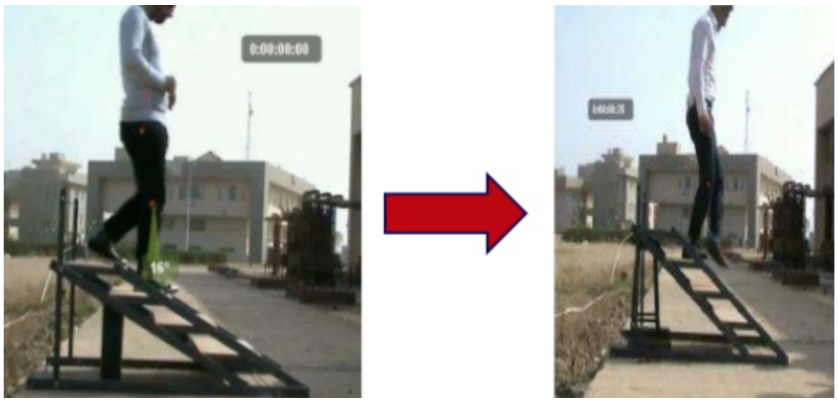


Abbildung 5.8.: Grundhaltung Treppenabstieg: Von Gewichtsaufnahme zum weiteren Abstieg [1]

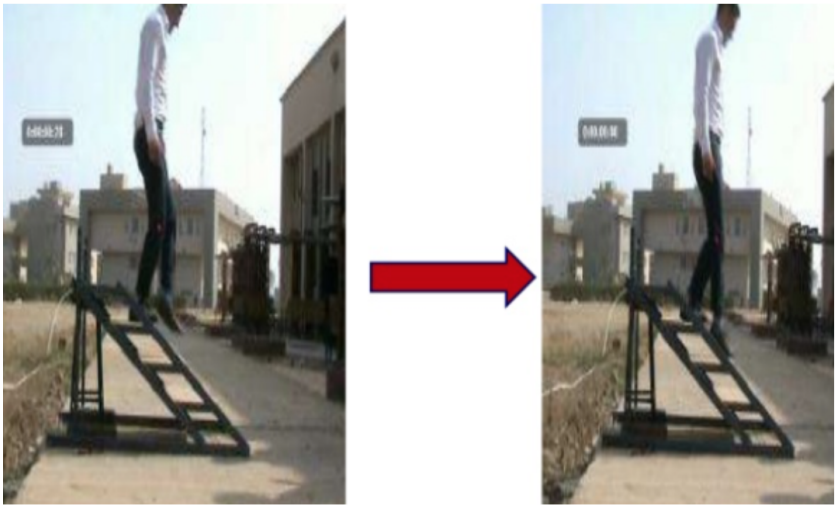


Abbildung 5.9.: Treppenabstieg: Kontrollierter Abstieg [1]

Die Schwungphase beim Treppenabstieg ist nochmals in zwei weiteren Phasen unterteilt. Die Hüfte wird dabei zwischen 40-45° gebeugt.

1. Bein anziehen (Schwungphase durch das Bein)

2. Fußplatzierung

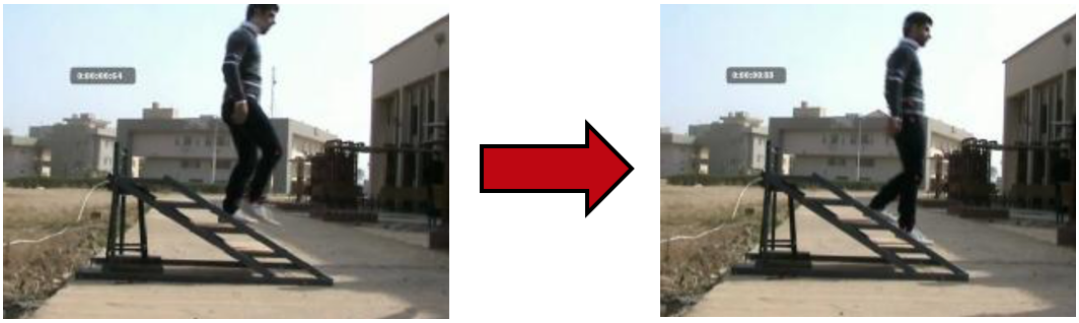


Abbildung 5.10.: Treppenabstieg: Von der Schwungphase zur Fußplatzierung [1]

Treppenabstieg - MAMKS

Bei dem Treppenabstieg verhalten sich die Daten ähnlich wie beim Treppensteigen. Lediglich in den Luftdruckdaten ist ein Unterschied zu erkennen. Dieser ist wiederum sehr gering, da bei einer kleinen Anzahl an Stufen nur sehr schwer ein Luftdruckabfall bzw. Luftdruckzuwachs zu erkennen ist.

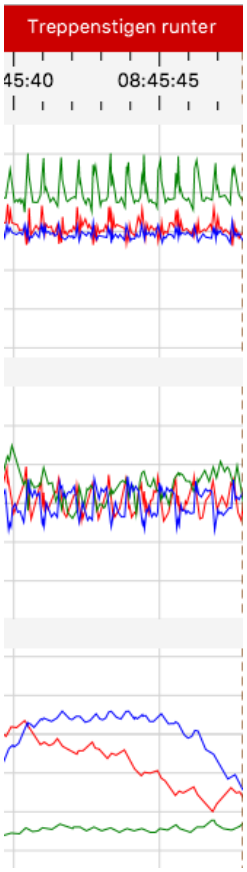


Abbildung 5.11.: Treppenabstieg in der MAMKS Software. Oberste Linie: Beschleunigung. Mittlere: Gyroskop. Unterste: Magnetometer

5.3. Biomechanische Analyse: Aufstehen/Hinsetzen

Der Vorgang des Aufstehens teilt sich in vier Phasen ein. Diese vier Phasen sind in Abbildung 5.12 zu sehen.

1. Flexion Momentum

   Die erste Phase beginnt mit der Initiierung der Bewegung und endet genau bevor das Gesäß sich vom Stuhl hebt. Dabei rotatiert das Becken nach vorne. Dies kann in den Sensordaten an Änderungen der Gyroskop X-Achse und an Änderungen der Magnetometer Z-Achse erkannt werden[14].

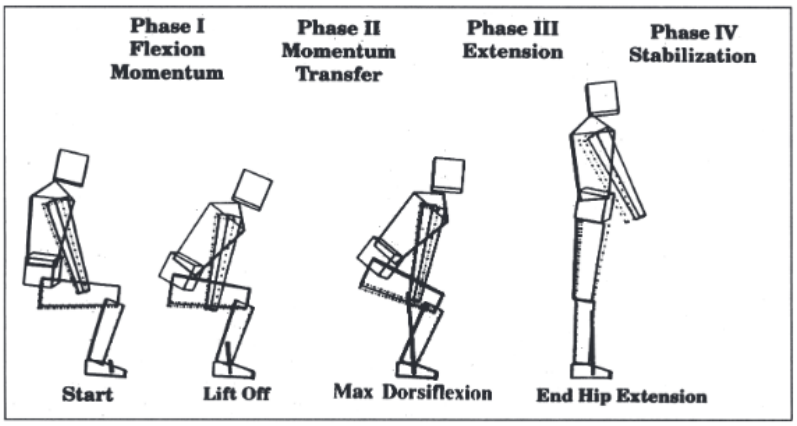


Figure 4. *Four phases of rising marked by four key events. Because the arms are modeled as single segments (using one array), the fact that the forearms are folded across the chest is not reflected in the figure. (Reprinted with permission of the American Physical Therapy Association.[18])*

Abbildung 5.12.: Die vier Phasen des Aufstehens[14]

2. Momentum Transfer

   Die zweite Phase beginnt, wenn das Gesäß vom Stuhl gehoben wurde. Sie endet, wenn die Beugung des Fußgelenks maximal ist. In dieser Phase erfolgt eine Aufwärts-Vorwärts Bewegung des Beckens. Dies hat Auswirkungen auf die Gyroskop X-Achse, die Magnetometer Z- und Y-Achse[14].

3. Extension

   Die dritte Phase beginnt direkt nach der maximalen Fußbeugung und endet, wenn die Hüfte aufhört sich zu strecken. Die Auswirkungen auf das Becken sind in dieser Phase noch genauso wie in der zweiten Phase[14].

4. Stabilisierung Die vierte Phase dient der Stabilisierung. Hier gibt es eine leichte Rotation zwischen Beugung und Streckung der Hüfte. Dadurch müssten leichte Auswirkungen auf die Gyroskop X-Achse entstehen. Die sind allerdings aufgrund ihrer Schwäche vernachlässigbar[14].

Diese vier Phasen lassen sich in unseren Sensorgürteldaten leider nicht optimal erkennen. Allerdings lässt sich anhand der Phasen gut erkennen, welche Sensordaten sich im Laufe

des Aufstehprozesses ändern müssen. Das Magnetometer reagiert auf eine veränderte Position im Magnetfeld. Die Bewegung des Aufstehens geht vorwärts-aufwärts. Wie in Abbildung 5.13 zu sehen ist, müsste die vorwärts-Bewegung Auswirkung auf die Z-Achse und die aufwärts-Bewegung Auswirkungen auf die Y-Achse haben. Außerdem spielt das Gyroskop beim Erkennen des Aufstehens eine Rolle. Das Becken rotiert vorwärts, das heißt es rotiert um die X-Achse. Demnach müssen auch Auswirkungen an den Daten des Gyroskops auf der X-Achse zu sehen sein. Diese Beobachtung lässt sich in den Studi-

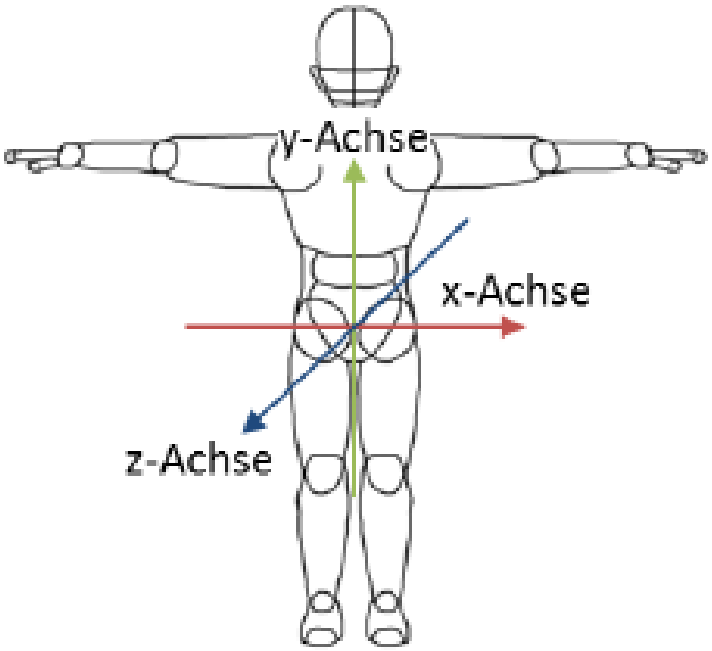
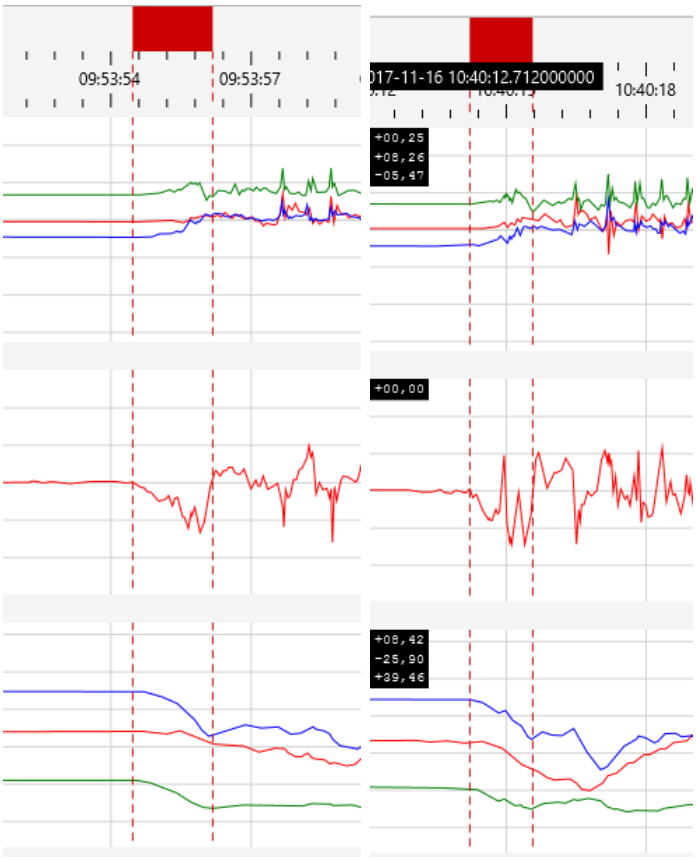


Abbildung 5.13.: Ausrichtung der Sensoren

endaten bestätigen. In Abbildung 5.14 sind Datenbeispiele für das Aufstehen zu sehen. Beim Aufstehen erkennt man einen Abfall der Magnetometer Y-Achse (dritter Graph, grün) und der Magnetometer Z-Achse (dritter Graph, blau). Bei den Gyroskopdaten (zweiter Graph) erkennt man die Unterschiede am besten, wenn man die y- und die z-Achse ausblendet. Auf der X-Achse (rot) ist trotz Schwankungen insgesamt ein Abfall und ein Anstieg der Winkelgeschwindigkeit zu erkennen.

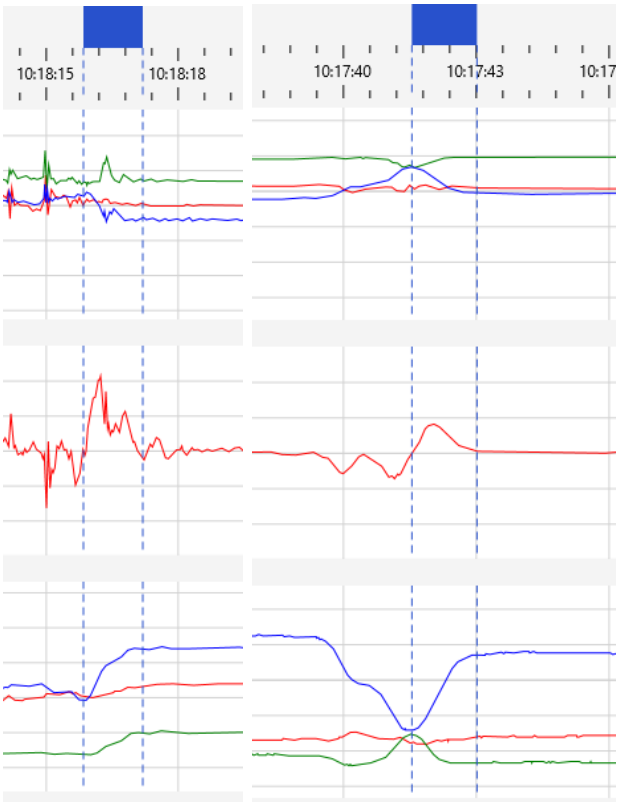
Beim Labeln ist es besonders wichtig auf die Y-Achse des Magnetometers zu achten. Da



(a) Beispiel 1 (b) Beispiel 2

Abbildung 5.14.: Datenbeispiele für das Aufstehen

dies den Höhenverlauf am besten darstellt. Als gute Hilfe lässt sich dann jedoch auch noch die Gyroskop X-Achse verwenden. Am Anfang des Aufstehens ist diese Linie auf dem Nullpunkt und auch am Ende des Aufstehens ist der Nullpunkt wieder erreicht. Der Startpunkt des Labels sollte da liegen, wo die Gyroskop X-Achse einen Nullpunkt hat und die Y-Achse des Magnetometers seinen höchsten Punkt hat. Der Endpunkt des Labels sollte da liegen, wo die Gyroskop X-Achse erneut einen Nullpunkt erreicht hat und die Y-Achse des Magnetometers seinen lokalen Tiefpunkt erreicht hat.
Für das Hinsetzen lassen sich die Phasen in umgekehrter Reihenfolge anwenden. Und auch die Achsen entwickeln sich jeweils in die entgegengesetzte Richtung. In Abbildung 5.15 sind Datenbeispiele für das Hinsetzen abgebildet. Beim Hinsetzen erkennt man ei-



(a) Beispiel 1 (b) Beispiel 2

Abbildung 5.15.: Datenbeispiele für das Hinsetzen

nen Anstieg der Magnetometer Y-Achse (dritter Graph, grün) und der Magnetometer

Z-Achse (dritter Graph, blau). Bei den Gyroskopdaten (zweiter Graph) erkennt man die Unterschiede am besten, wenn man die Y- und die Z-Achse ausblendet. Auf der X-Achse (rot) ist trotz Schwankungen insgesamt ein Anstieg und ein Abfall der Winkelgeschwindigkeit zu erkennen.
Beim Labeln ist es besonders wichtig auf die Y-Achse des Magnetometers zu achten. Da dies den Höhenverlauf am besten darstellt. Als gute Hilfe lässt sich dann jedoch auch die Gyroskop X-Achse verwenden. Am Anfang des Hinsetzens ist diese Linie auf dem Nullpunkt und auch am Ende des Hinsetzens ist der Nullpunkt wieder erreicht. Der Startpunkt des Labels sollte da liegen, wo die Gyroskop X-Achse einen Nullpunkt hat und die Y-Achse des Magnetometers seinen tiefsten Punkt hat. Der Endpunkt des Labels sollte da liegen, wo die Gyroskop X-Achse erneut einen Nullpunkt erreicht hat und die Y-Achse des Magnetometers seinen lokalen Hochpunkt erreicht hat.

5.4. Biomechanische Analyse: Hocken

Die Aktivität Hocken ist eine kombinierte Bewegung und teilt sich in drei Phasen auf. Die erste Phase ist die Bewegung nach unten, die zweite Phase ist das verharren in der Hockposition und die dritte Phase ist das Aufrichten aus der Hockposition zum Stehen. Die erste und dritte Phase haben Auswirkungen auf das Knöchelgelenk, das Kniegelenk, das Hüftgelenk und die Wirbelsäule.
Das Knöchelgelenk wird während der ersten Phase in Körperrichtung gebeugt, der Fuß wird in Richtung Bein angezogen, bis eine Beugung von etwa 20 Prozent erreicht wurde. Diese Beugung wird in der zweiten Phase gehalten. In der dritten Phase wird das Knöchelgelenk gestreckt, bis der Fuß wieder im rechten Winkel zum Bein steht.
Das Kniegelenk wird entlang der sagittalen Ebene bewegt. In Phase eins wird das Kniegelenk bis zu 160 Prozent gebeugt. Die zweite Phase hat keine Auswirkungen auf die Beugung des Kniegelenks. Die Aktivität in dieser Phase ist statisch. In Phase drei wird das Kniegelenk wieder bis zu einer Beugung von 0 Prozent gestreckt. In den

Bewegungen gibt es eine Rotationsbewegung im Knie.
Die Hüfte wird in Phase eins ebenfalls entlang der sagittalen Ebene bewegt. Das Hüftgelenk wird ungefähr um 90 Prozent nach vorne gebeugt, in Phase zwei wird die Beugung gehalten und in Phase drei wieder gestreckt. Es wird also eine Rotation um das Hüftgelenk herum ausgeführt.
Auch die einzelnen Elemente der Wirbelsäule werden entlang der sagittalen Ebene gebeugt und gestreckt während der Aktivität Hocke[15].

Die Bewegung des Hockens hat große Ähnlichkeiten mit der Kniebeuge, weshalb die Abbildung 5.16 zur Verdeutlichung der ersten Phase des Hockens verwendet wird. Es wird im Stehen begonnen. In Bild 1 der Abbildung sind die Knie (A) noch gestreckt und auch das Hüftgelenk ist noch gestreckt. In Bild 2 sind die Knie etwas gebeugt und auch das Hüftgelenk ist leicht nach vorne gebeugt. In den anderen beiden Bildern wird diese Beugung weiter verstärkt. Im Unterschied zur Kniebeuge ist die Bewegung im Hocken oft noch weiter nach vorne gerichtet, da dabei meistens die Fußsohlen vom Boden angehoben werden und nur der vordere Teil des Fußes den Boden berührt. Die Beugung der oben genannten Gelenke bleibt dabei jedoch ähnlich, sodass die ganze Bewegung nach vorne gekippt durchgeführt wird.
Die dritte Phase der Bewegung wird analog in die andere Richtung ausgeführt. Die Gelenke werden nun langsam gestreckt im Laufe der Bewegung.
Die zweite Phase ändert an der Beugung der Gelenke nichts. In dieser Phase wird die in der ersten Phase angenommene Position gehalten.

Auf die Daten des Sensorgürtels bezogen haben diese Bewegungen folgende Auswirkungen. Da der Gürtel die Daten auf Hüfthöhe aufnimmt, kann hauptsächlich die Bewegung des Hüftgelenkes in der Bewegung gemessen werden. In Abbildung 5.17 ist ein Beispiel für das Hocken zu sehen. In der Abbildung sind die drei Phasen eingetragen.
In Phase 1 ist das Herunterbeugen zu erkennen. Die Bewegung verläuft vorwärts-abwärts.

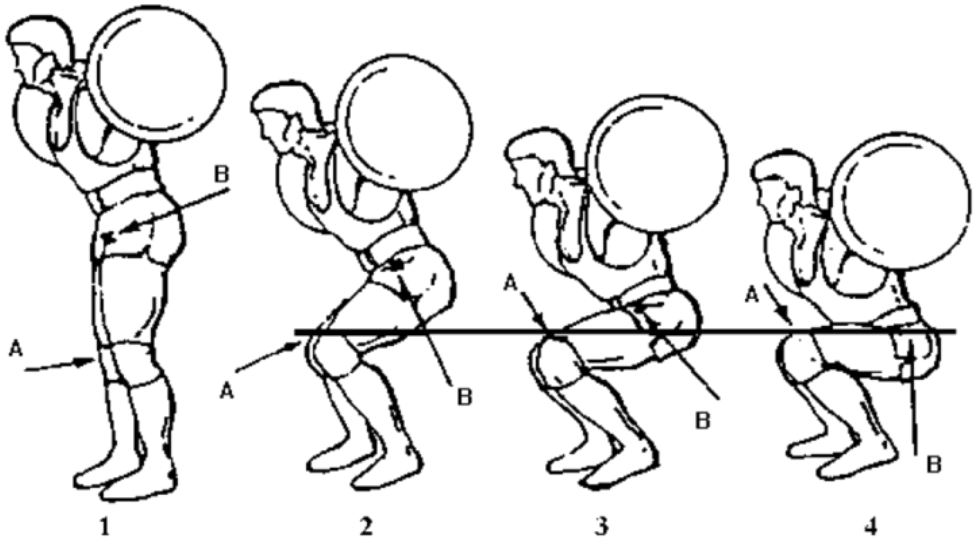


Abbildung 5.16.: Ablauf von Phase 1 des Hockens[17]

Die Vorwärtsbewegung hat Auswirkungen auf die Z-Achse der Magnetometerdaten, die Abwärtsbewegung hat Auswirkungen auf die Y-Achse der Magnetometerdaten. In der Abbildung sind die Magnetometerdaten auf der dritten Datenreihe zu sehen. Die Y-Achse ist die grüne Linie und die Z-Achse ist die blaue Linie. Die Abwärtsbewegung bewirkt einen Anstieg auf der Magnetometer Y-Achse und die Vorwärtsbewegung bewirkt einen Abfall auf der Magnetometer Z-Achse. Gleichzeitig lässt sich die Rotation der Hüfte auf der Gyroskop X-Achse erkennen. Die Gyroskopdaten sind in der zweiten Datenreihe zu sehen. Auf der Abbildung sind die Y- und die Z-Achse ausgeblendet, sodass die rote Linie die X-Achse ist. Die Vorwärtsrotation der Hüfte löst einen negativen Peak auf der Gyroskop X-Achse aus.

In Phase 2 sind alle Linien größtenteils konstant. Es handelt sich um eine statische Bewegung.

Phase 3 verläuft umgekehrt zu Phase 1. Es handelt sich um eine rückwärts-aufwärts Bewegung. Die Rückwärtsbewegung lässt sich durch einen Anstieg der Magnetometer Z-Achse erkennen. Die Aufwärtsbewegung wird durch den Abfall der Magnetometer Y-Achse gekennzeichnet. Das Hüftgelenk rotiert in diese Phase nach hinten. Dies ist durch

einen positiven Peak der Gyroskop X-Achse zu erkennen.

Der Anfang der gesamten Hockbewegung lässt sich besonders gut durch den Peak der Gyroskop X-Achse erkennen. Hier ist der Startpunkt der Punkt, an dem die Daten bei 0 liegen bevor der Peak beginnt. Das Ende der Bewegung lässt sich durch den zweiten Peak der Gyroskop X-Achse definieren. Hier ist der Punkt der Endpunkt, an dem die Daten nach Ende des Peaks wieder den Nullpunkt erreichen.

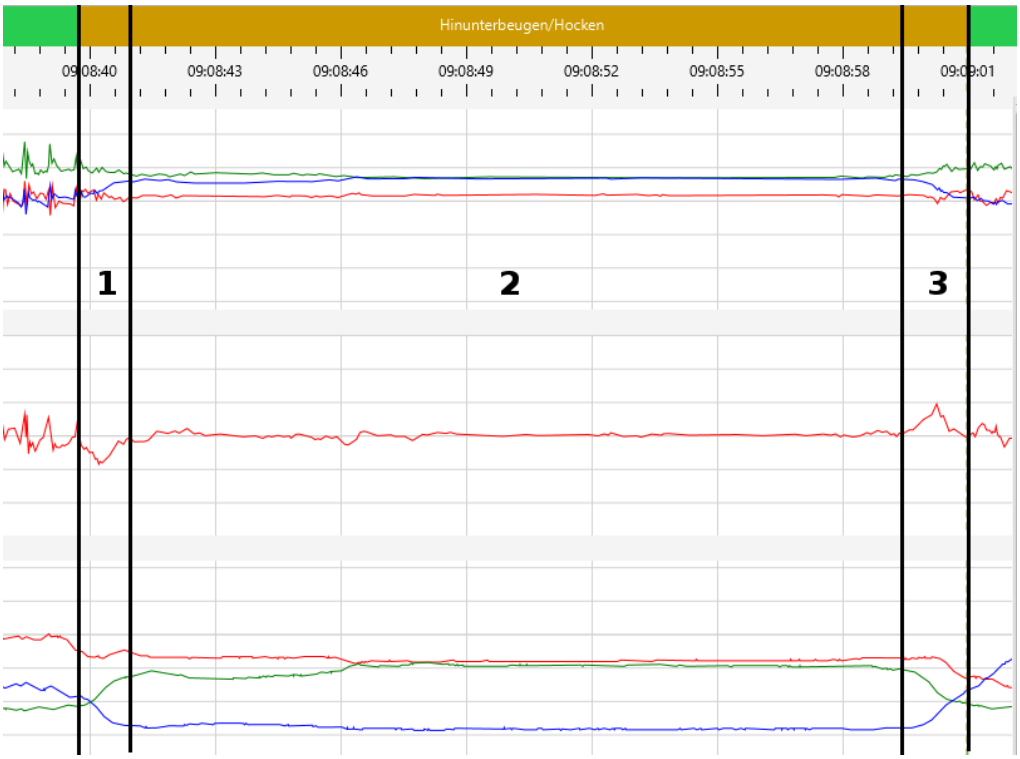


Abbildung 5.17.: Datenbeispiel für die Aktivität Hocken

5.5. Biomechanische Analyse: Gehen

Das Gehen wird in der Literatur oft bezogen auf ein Referenzbein beschrieben. Die Abbildung 5.18 zeigt einen typischen Gangzyklus und die zur Beschreibung notwendigen Namenskonventionen. Die verwendete Grafik hierzu, sowie die Ausführungen zu den einzelnen Phasen, wurden aus [4], sowie aus [3] und [18] übernommen.

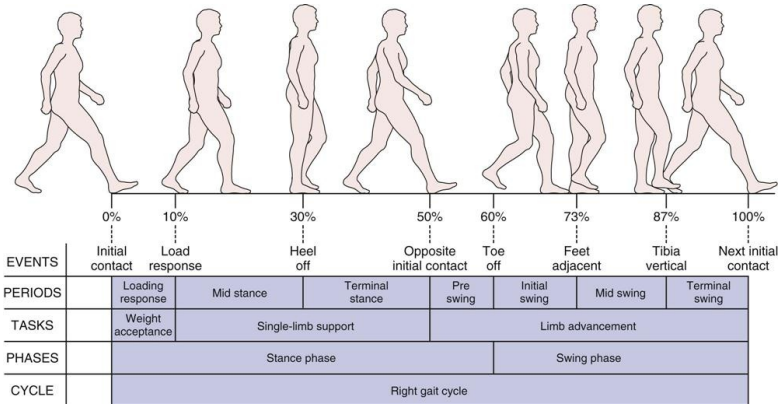


Abbildung 5.18.: Der Gangzyklus [4]

Innerhalb eines Zyklus unterscheidet man Events, Perioden, Phasen und Tasks. Dabei sind Events kurze Momente, die die Perioden voneinander trennen. Die Perioden und Tasks sind ihrerseits in den Phasen des Gangzykluses enthalten.
Festzuhalten ist, dass ein Gangzyklus mit dem Auftreten des initialen Bodenkontaktes des Referenzbeines beginnt und mit dem Auftreten des nachfolgenden Initialen Bodenkontaktes des gleichen Beins endet. Ein Gangzyklus bezieht sich auf ein, dem Betrachter zugewandtes, Referenzbein.

Die Beschreibung des Gangzyklus bezogen auf ein Bein liefert zunächst nur indirekt übertragbare Informationen. Die folgende Beschreibung dient einer direkten Übertragung auf die Daten des Sensorgürtels.
Die Abbildung 5.19 zeigt einen Gangzyklus bezogen auf die Beckenrotation, betrachtet aus der Sagittalebene.

In der Abbildung 5.19 sind zwei Gangzyklen dargestellt. Die schwarze durchgezogene Linie zeigt die Beckenkippung beim Gehen eines männlichen Probanden auf einem festen Untergrund, während die gestrichelte Linie das Gehen auf einem Laufband darstellt. Dabei ist sichtbar, dass beim schnelleren Gehen (Laufband) das Becken um etwa 5-7 Grad stärker nach Anterior (vorne) gekippt ist. Das Kippen findet in beiden Fällen um 3-4 Grad um den jeweiligen Mittelwert (10 Grad beim schnellen Gehen) statt. Dabei

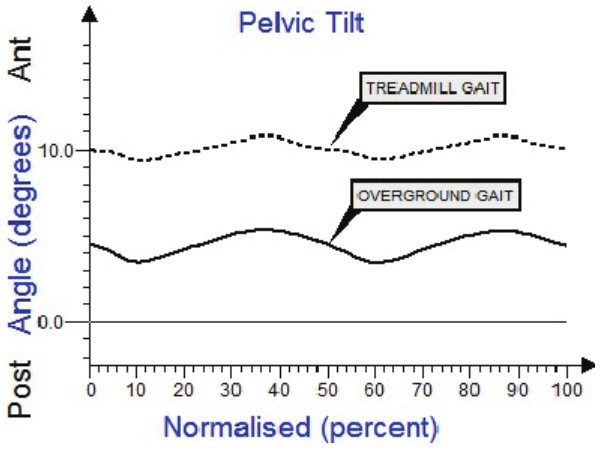


Abbildung 5.19.: Becken aus Sagittalebene betrachtet [18]

sind vier Extrempunkte in der Kurve zu sehen. Der erste Tiefpunkt markiert den Übergang zwischen der Loading-Response und der Mid Stance Periode. Dieser Extrempunkt entspricht in etwa dem Load-Response Event. Auf die Daten des Sensorgürtels bezogen, sollte sich diese Bewegung in der X-Achse des Gyroskops wiederfinden. Dabei sollte der Wert der radialen Geschwindigkeit hier etwa bei 0 liegen, da die Radialgeschwindigkeit der ersten Ableitung des Radialwertes entspricht. Der zweite Extrempunkt ist der Hochpunkt, der bei etwa 35-40 Prozent des Gangzyklus angesiedelt ist. Hier erreicht das Becken seine Maximale Kippung nach vorne und das Referenzbein ist gerade dabei, die Ferse abzuheben, sodass der Hochpunkt das Heel-Off Event markiert. In den Daten des Sensorgürtels müsste die Periode als ein Bogen zwischen zwei Nullstellen auf der X-Achse des Gyroskops zu sehen sein. Der dritte Extrempunkt ist das Toe-Off Event und die dazwischenliegende Periode ist Pre-Swing (Vorschwungphase). Dabei kippt das Becken verstärkt nach vorne aufgrund der raschen Gewichtsumverteilung nach dem Abheben des Vorfußes. In den Daten des Sensorgürtels müsste diese Periode als umgekehrter Bogen zwischen zwei Nullstellen sichtbar sein. Der letzte Hochpunkt liegt zwischen 80-87 Prozent des Gangzyklus und entspricht damit dem Tibia-Vertical Event. Hierbei ist das Becken aufgrund des nach vorne verlagerten und sich im Schwung befindlichen Beins, maximal nach vorne gekippt. Auch hier ist zwischen 60-90 Prozent des Gangzyklus ein Bogen zwischen zwei Nullstellen zu erwarten. Insgesamt ist zu erwarten, dass die Ra-

dialgeschwindigkeit auf der X-Achse des Gyroskops vergleichsweise schwach ausgeprägt sein wird, da in dieser Achse nur leichte Rotationen von 3-4 Grad stattfinden.

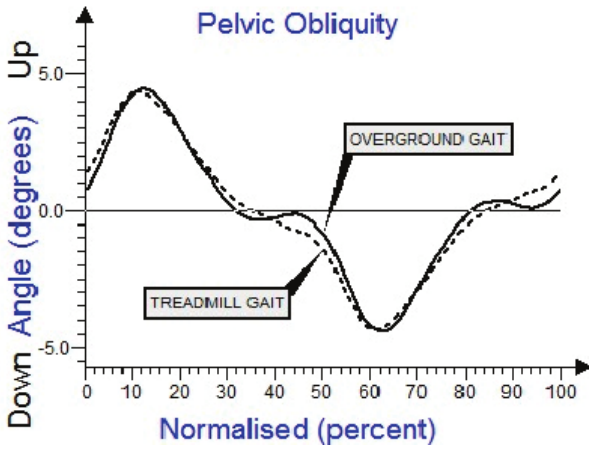


Abbildung 5.20.: Becken aus Frontalebene betrachtet [18]

Abbildung 5.20 zeigt die Rotation des Beckens aus der Frontalebene gesehen, also auf der sagitto-transversalen Achse. Dies entspricht der Rotation auf der Z-Achse in den Sensorgürteldaten. Hier ist anzumerken, dass Die Neigung des Beckens in dieser Achse je nach Alter und Geschlecht sehr unterschiedlich ausfallen kann. Bei weiblichen Geschlecht neigt das Becken sich stärker. Bei älteren Personen, sowie bei Personen mit Gleichgewichtsstörungen ist die Neigung sehr schwach ausgeprägt. Insgesamt zeigt die Abbildung 5.20 drei markante Extrema. Der erste Hochpunkt ist zwischen 10-15 Prozent des Gangzyklus platziert. Er markiert die höchste Neigung des Beckens nach oben nach einem Heel-Strike (Fersenaufprall) und darauf folgenden Gewichtsverlagerung und Ausweichbewegung des Oberkörpers auf die Referenzseite. Die Sensordaten müssten an dieser Stelle einen, nach raschen Abfall der Radialgeschwindigkeit folgenden Nullpunkt in der Z-Achse des Gyroskops aufweisen. Zwischen 40-50 Prozent des Gangzyklus ist ein Plateau bei etwa 0 Grad Rotation zu sehen. Hierbei richtet sich das Becken nach dem Fersenaufschlag wieder in die neutrale Stellung zurück. Zu sehen ist außerdem, dass das Plateau je bei höherer Ganggeschwindigkeit weniger ausgeprägt sein kann. In den

Sensorgürteldaten müsste ein steiler Bogen zwischen zwei Nullpunkten bei 10 und 45 Prozent des Gangzyklus zu beobachten sein. Der dritte Extrempunkt ist ein Tiefpunkt, der eine Beckenneigung auf der kontralateralen Körperseite markiert, die Beschreibung der Bewegung ist dabei anlog zur Referenzseite. In den Sensordaten müsste, umgekehrt wie bei dem ersten Extrempunkt ein steiler Abstieg der Radialgeschwindigkeit auf de Z-Achse des Gyroskops zu sehen sein.

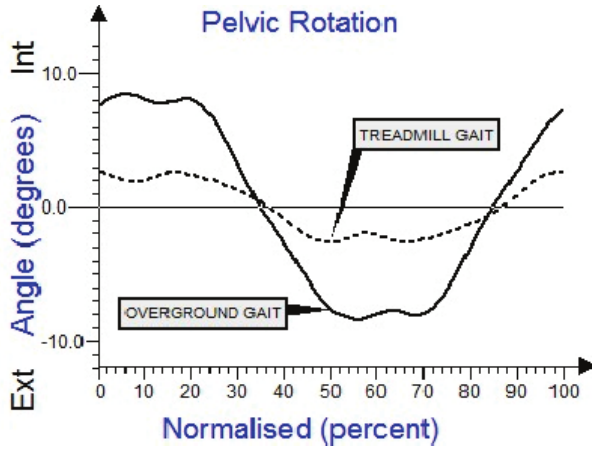


Abbildung 5.21.: Becken aus Transversalebene betrachtet [18]

Abbildung 5.21 zeigt die Beckenbewegung aus der Transversalebene betrachtet, das heißt, die Rotation finden auf der sagitto-frontalen Achse statt. Dies entspricht einer Rotation auf der Y-Achse des Gyroskops im Sensorgürtel. Zwischen 0-25 Prozent des Gangzyklus bleibt das Becken relativ konstant nach innen (interior) rotiert bleibt. Zur Erinnerung - Interior bezieht sich hier auf das Referenzbein. In den Sensorgürteldaten ist hier daher eine Radialgeschwindigkeit von etwa 0 rad/sec im Bereich von 0-25 Prozent des Gangzyklus auf der Y-Achse zu erwarten sind. Zwischen 25-50 Prozent rotiert das Becken zur kontralateralen Seite (Exterior, von der Referenzseite aus gesehen). Auf der Y-Achse des Gyroskops ist dieser Wechsel daher als ein steiler Anstieg und anschließender Abfall der Radialgeschwindigkeit zu erwarten. Auf der kontralateralen Seite verbleibt das Becken konstant bei etwa -10 Grad rotiert. Zwischen 45-70 Prozent

des Gangzyklus ist auch hier daher eine Radialgeschwindigkeit von etwa 0 rad/sec zu erwarten. Schließlich folgt bei 70 bis 100 Prozent des Gangzyklus eine letzte Beckenrotation, sodass das Becken zum weiteren Initialen Bodenkontakt, nach innen rotiert, bereit steht. Hier ist eine steile Geschwindigkeitszunahme auf der kontralateralen Seite, also negative Geschwindigkeit auf der Y-Achse des Gyroskops zu erwarten. Anschließend stellt sich gegen Ende des Zyklus eine neutrale Radialgeschwindigkeit ein. Da die stärkste Beckenbewegung auf der Transversalebene stattfindet sind auch die deutlichsten Wertverläufe auf der Y-Achse des Gyroskops zu erwarten.

5.6. Biomechanische Analyse: Liegen, Hinlegen, Aufsetzen

Der folgende Abschnitt wird der einfacheren Übersicht in zwei Teile unterteilt. Im ersten Abschnitt wird das Liegen beschrieben und die vom Gürtel dazu aufgenommenen Daten gezeigt und analysiert. Im nachfolgenden Teil wird der Ablauf des Hinlegens sowie des Aufsetzens beschrieben. In Abbildung 5.22 ist der Übersicht halber der gesamte Ablauf im MAMKS dargestellt.

5.6.1. Biomechanische Analyse: Liegen

Die Aktivität Liegen wird in der MAMKS Software in zwei verschiedene Aktivitäten geteilt. Zum einen das Liegen auf dem Rücken, sowie das Liegen auf der Seite. Die Abbildungen 5.23 zeigen die hier beschriebenen Liegepositionen. In der Abbildung 5.24 ist das Liegen auf dem Rücken, wie es in MAMKS darstellbar ist, abgebildet. Im Vergleich dazu ist in der Abbildung 5.25 Liegen auf der Seite abgebildet.

Wie in den Abbildungen 5.24 und 5.25 zu sehen, kann anhand der Sensordaten des Gür-

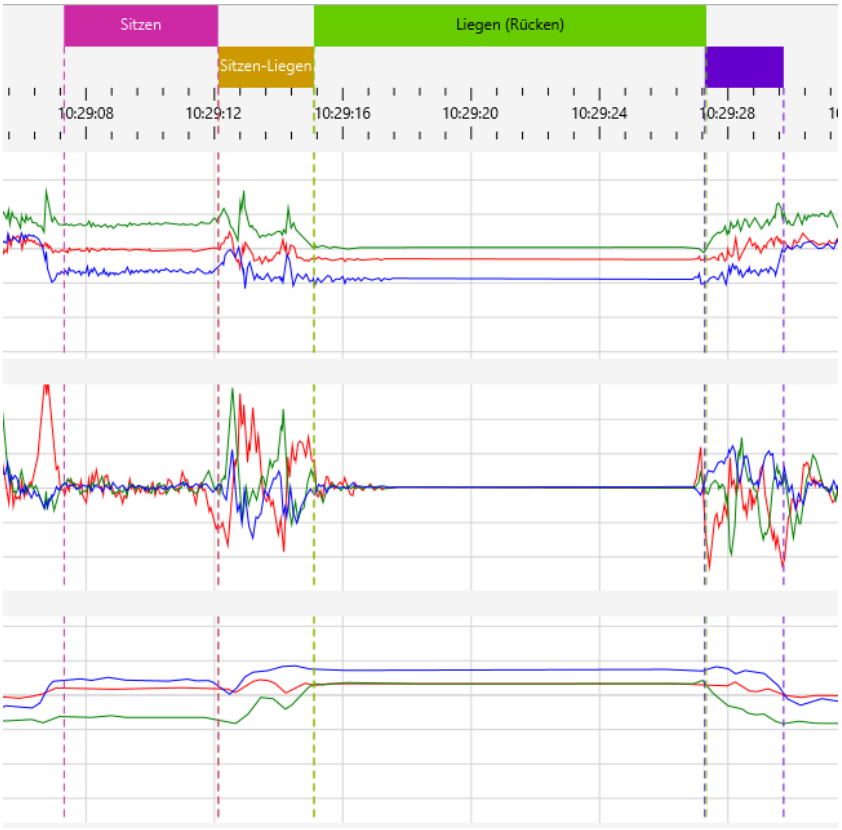
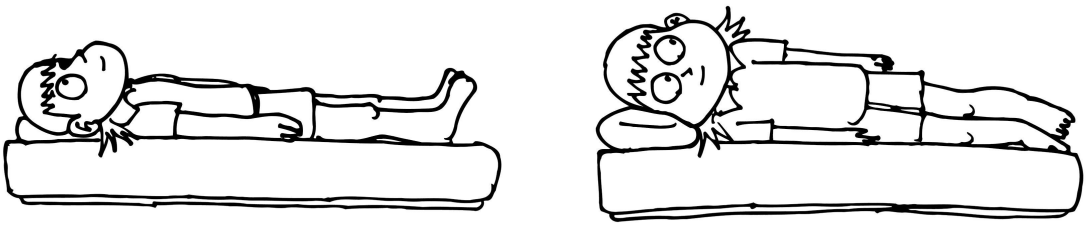


Abbildung 5.22.: Ablauf - Hinlegen - Liegen - Aufsetzen



(a) Skizze - Liegen Rücken - Quelle: https://www.schulbilder.org/malvorlage-auf-dem-ruecken-liegen-i11792.html

(b) Skizze - Liegen Seite - Quelle: https://www.schulbilder.org/malvorlage-auf-der-seite-liegen-i11765.html

Abbildung 5.23.: Skizzen Liegen

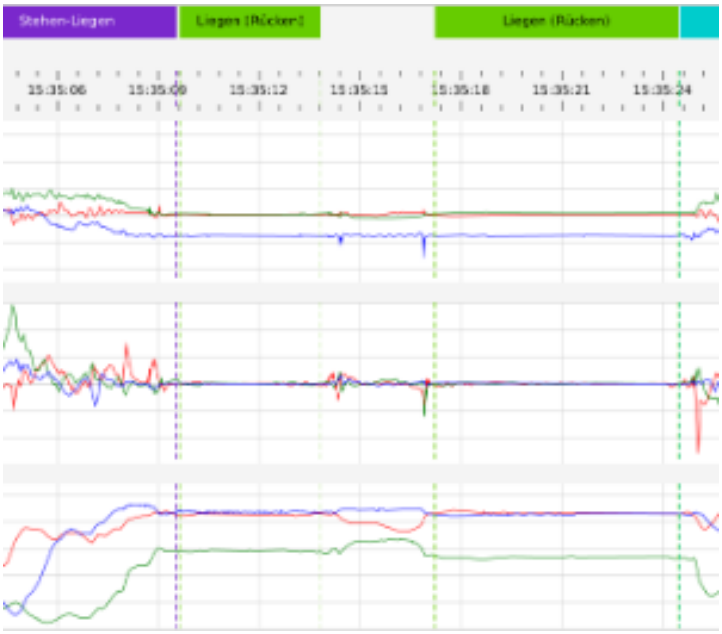


Abbildung 5.24.: MAMKS Screenshot - Liegen Rücken

tels keine eindeutige Unterscheidung stattfinden. Sowohl die Accelerometerdaten (erste Zeile) als auch die Gyroskopdaten (zweite Zeile) verändern sich beim Liegen nicht bzw. nur sehr gering. Einzig die Magnetometerdaten unterscheiden sich in allen Abbildungen. Dies ist auf die relative Position zum Erdmagnetfeld zurückzuführen. Daher kann auch mit Hilfe des Magnetometers (dritte Zeile) keine eindeutige Bestimmung der Liegeposition stattfinden.

5.6.2. Biomechanische Analyse: Aufsetzen / Hinlegen

Im Gegensatz zum Liegen kann man anhand der Gürteldaten unterscheiden, ob sich ein Proband hinlegt oder aufsetzt. In der Abbildung 5.26 ist ein MAMKS Screenshots abgebildet, welcher das Hinlegen und Aufsetzen zeigt.

Die Aktivität des Hinlegens bzw. Aufsetzens besteht im Wesentlichen aus zwei Teilen. Zum einen muss in der Regel der Körper gedreht werden, damit ein Hinlegen ermöglicht wird. Ausgenommen davon ist das Hinlegen, wenn der Proband auf dem Boden sitzt.

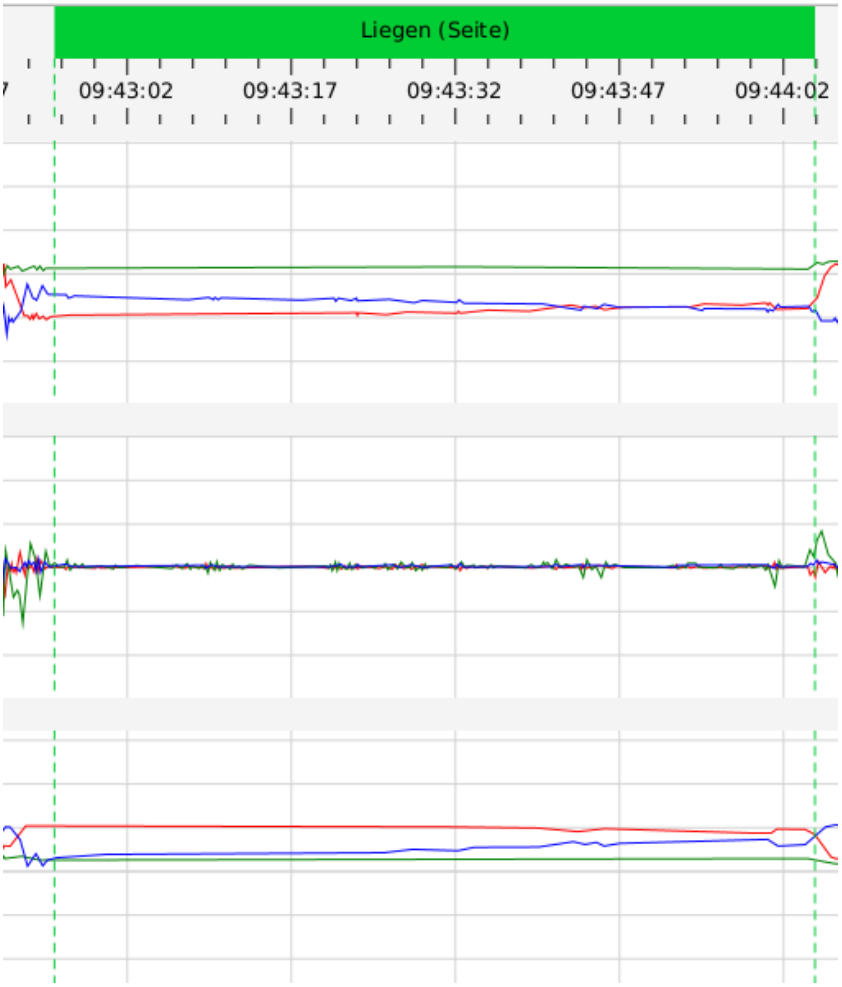


Abbildung 5.25.: MAMKS Screenshot - Liegen Seite

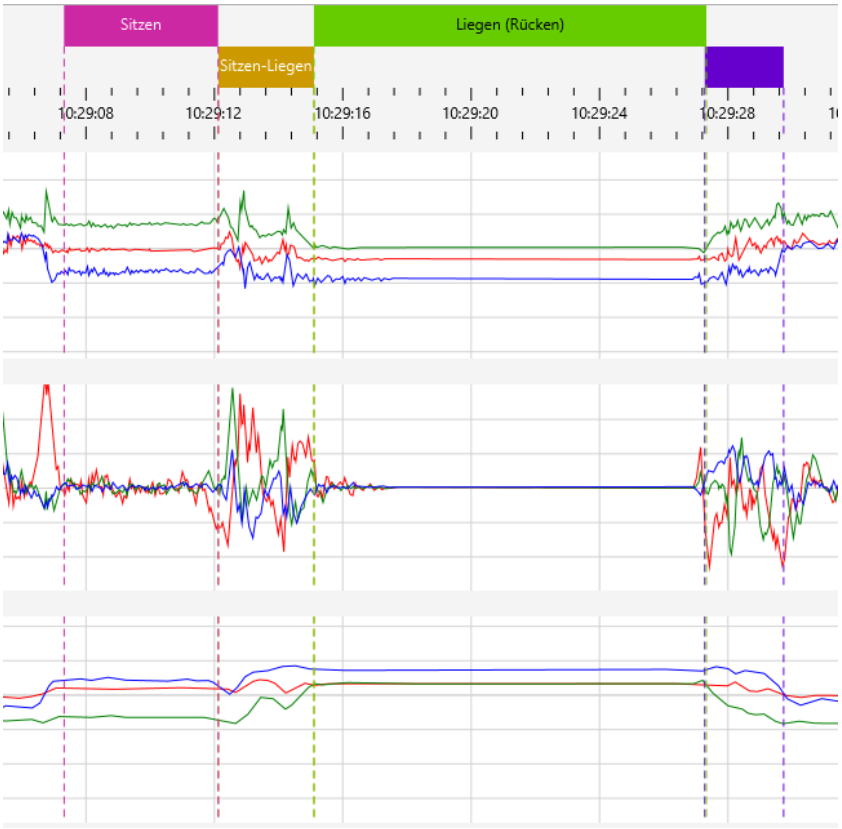


Abbildung 5.26.: MAMKS Screenshot - Hinlegen - Aufsetzen

Andernfalls ist diese Drehung nötig, um die Beine auf das gleiche Level des Beckens zu heben. Der zweite Teil, welcher oftmals gleichzeitig stattfindet, ist das eigentliche Hinlegen. Also das Niederlassen des Oberkörpers und Strecken der Beine. Dabei bewegt sich das Becken auf der X-Achse.

Die hier genannten Aktivitäten können mit Hilfe des Gyroskops gut beobachtet werden. Die erste Rotation findet auf der Y-Achse statt. In Abbildung 5.26 kann dies auf der grünen Y-Achse des Gyroskops (zweite Zeile) beobachtet werden. Die zweite Rotation, welche zum Hinlegen und Aufsetzen benötigt wird, lässt die X-Achse des Gyroskops ausschlagen. In der Abbildung ist diese Achse rot abgebildet. Mit Hilfe dessen kann auch unterschieden werden, ob der Proband vor dem Hinlegen auf dem Boden saß und seinen Körper nur gestreckt hat oder ob der Proband bspw. am Rand eines Bettes saß.

Das Hinlegen und Aufsetzen kann man anhand der Accelerometerdaten bestimmen. Ein Anstieg der Achsen deutet auf ein Aufsetzen hin. Gegensätzlich zeigt ein Abgang der Werte ein Hinlegen des Probanden. Mit Hilfe dieser Beobachtungen, können die Aktivitäten annotiert werden.

6. Studiendurchführung

Für das Training der überwachten maschinellen Lernverfahren, ebenso wie die Überprüfung der Klassifikationsergebnisse, werden gelabelte Daten benötigt. Die PGMAMKS hat sich auf die Annotation der Sensordaten konzentriert, die während der Assessments aufgenommen wurden. Für die Daten, die im häuslichen Umfeld aufgenommen wurden, existiert jedoch kein standardisierter Ablauf wie bei den Assessments, sodass eine alternative Vorgehensweise für das Labeln der Heimdaten entwickelt werden musste.

6.1. Vorbereitung der Studie

Im Zuge der Diskussion bezüglich der Problematik der Annotation der Heimdaten wurden verschiedene Verfahren in Betracht gezogen, die es ermöglichen sollten, möglichst genaue Label zu generieren. Dabei wurde die Verwendung von Stift und Papier zur genauen Dokumentation der Aktivitäten beispielsweise ausgeschlossen. Die durch die Probanden geführten Tagebücher wiesen bereits eine viel zu hohe Ungenauigkeit auf und die benötigten Aktivitäten wie *Gehen*, *Stehen* oder *Hinsetzen* konnten nicht aus diesen abgeleitet werden. Die Aufzeichnung der Aktivitäten mittels Video wurde ebenfalls verworfen, da allein für den häuslichen Bereich zahlreiche Kameras installiert werden müssten und es einen ethisch nicht vereinbarten Eingriff in die Privatsphäre der Probanden darstellt.
Um einen ausreichend detaillierten Einblick in die häuslichen Aktivitäten der Senioren zu erhalten, wurde schließlich eine Android-App (vgl. Kapitel 7.1) entwickelt, die es ermöglicht, die relevanten Aktivitäten auch bei schnellen Abfolgen mit genauen Zeitstempeln zu dokumentieren. Zudem können die Informationen direkt für die Annotation verwendet werden und müssen nicht z. B. nochmals aus einem Video oder niedergeschriebenen Texte extrahiert werden.

Um zu gewährleisten, dass die, mittels der App, dokumentierten Aktivitäten die benötigte Qualität für das Labeln der Daten aufweisen, erfolgt die Aktivitätsdokumentation durch die Projektgruppenteilnehmer persönlich. Zu diesem Zweck werden wie nachfolgend beschrieben Hausbesuche bei den Probanden durchgeführt. Das Vorhaben wurde in einem Ethikantrag dokumentiert und genehmigt.
Um mögliche Probleme und nützliche Anpassungen der MAMKS-Software frühzeitig zu erkennen, wurde von fünf Projektgruppenteilnehmern das Verfahren im Selbsttest in einer Pilotstudie getestet.

6.2. Pilotstudie

Einleitung

Um sich einen Überblick darüber zu verschaffen, worauf geachtet werden muss, bevor das produktive Labeln der Probanden durchgeführt wird, wurde eine Pilotstudie angesetzt. Durch sie sollte ermittelt werden, wo Verbesserungspotenzial besteht und worauf man sich am besten vorbereiten sollte. Des Weiteren sollten Verbesserungsvorschläge für die App gesammelt und ein erster Eindruck ermittelt werden. Nachfolgend eine kurze Zusammenfassung der durchgeführten Studie.

Erkenntnisse vor/während des Erfassens

- Das zuverlässige Labeln erfordert Übung mit der App, da sich die Position der Aktivitäten eingeprägt werden sollte. Somit kann schneller reagiert werden um ein bestmögliches Ergebnis zu erzielen.

- Der Gürtel muss vor der Verwendung stets ausreichend aufgeladen sein.

- Die Messung muss immer im stehenden Zustand gestartet werden, damit das An-

legen des Gürtels nicht als Aktivität gewertet wird.

- Unklarheit beim Labeln der einzelnen Aktivitäten tritt auf. Wann beginnt eine Aktivität und wann endet sie. Wie sollen Transitionen gehandhabt werden. Es muss eine entsprechende Richtlinie entworfen werden.

- Alle Aktivitäten zuverlässig zu erfassen erfordert hohe Konzentration.

- Übermittlung an den Server sollte in der Humotion-Software deaktiviert werden.

Erkenntnisse während der Nachbearbeitung

- die Android-Labeldaten dürfen nicht länger sein als die vom Gürtel aufgenommenen Daten, da sie sonst nicht eingelesen werden können.

- das manuelle Ändern in der ursprünglichen MAMKS-Version dauert sehr lange, da alle Zeiträume manuell angepasst werden müssen.

- der Import von mehreren Android-Labeldaten ist nicht möglich, da MAMKS versucht die Daten unter gleichem Namen zu speichern.

- Magnetometerdaten helfen sehr gut einige Aktivitäten zuverlässig zu erkennen.

- Zwischen den Label-Daten und den Sensorgürteldaten gibt es meistens einen Offset in der Zeit. Dieser muss beim manuellen Bearbeiten korrigiert werden. Als Workaround kann der Zeitraum der Sensordaten um x-Sekunden verschoben werden, indem man in der entsprechenden Datei den Anfangs- und Endzeitpunkt verschiebt.

- Einige Aktivitäten sind leicht verfälscht, da zum Beispiel ein Sitzen bei einigen Personen oft einem Liegen ähnelt (nicht gerade Sitzposition). Dies könnte im weiteren Verlauf eventuell ein Problem bei der Klassifikation durch die Algorithmen darstellen.

6.3. Durchführung der Studie

Für die Studie sollten insgesamt bis zu 50 Probanden über maximal 4 Stunden Zuhause besucht werden. Der Umkreis für die Auswahl sollte in der näheren Umgebung von Oldenburg erfolgen. Des Weiteren sollten die Probanden körperlich und geistig *fit* sein, dass heißt in ihrem Alltag zu recht kommen und keine Probleme bei längeren Gehstrecken oder Treppensteigen haben. Der genaue Ablauf der Terminvergabe wurde in der Verfahrensanweisung VA-Nr. Versa-10 (siehe Anhang A) festgehalten und wurde in Kooperation mit den Studynurses durchgeführt.
Für den Besuch der Probanden wurde der Humotion Sensorgürtel, ein Tablet und ein Entfernungsmessgerät eingesetzt. Die Tablets wurden für die eigen entwickelte Aktivitätstracking-App genutzt. Bei den Terminen selbst waren jeweils zwei Mitglieder der Projektgruppe pro Proband eingeplant. Dabei hatte ein Mitglied die Aufgabe, das Labeln mithilfe der App zu übernehmen und die andere Person, die Aktivitäten zu überwachen sowie den Ablauf des Besuchs zu gestalten. Vor Ort wurden am Anfang alle möglichen Fragen geklärt und der Ablauf erläutert. Nach dem Einführungsgespräch wurde der Gürtel und die App mit der dazugehörigen SeniorHome-ID(SHID) gestartet. Während der Aufnahme sollten die Probanden ihren alltäglichen Erledigungen nachgehen, wie das Einkaufen gehen, Büro arbeiten oder das Kochen. Der genaue Ablauf der Probandenbesuche ist in der VA-Nr. Versa-11 (siehe Anhang A) festgehalten.

6.4. Manuelle Labelbearbeitung

Für eine weitere Verarbeitung der aufgenommenen Daten in der MAMSK-Software müssen sowohl die exportierten Daten des Sensorgürtels, als auch die mittels App dokumentierten Label-Daten in die Software importiert werden. Durch die Erweiterung der MAMKS-Software um die Funktionalität „Label importieren“ können die Label-Daten direkt den Sensordaten zugeordnet und visualisiert werden.

In seltenen Fällen kann es zu einer zeitlichen Verschiebung von Label und Sensordaten kommen, z. B. das Label Gehen ist immer erst 5 Sekunden nach Beginn der Aktivität in den Daten markiert. Für eine einfachere Weiterverarbeitung ist eine Korrektur durch Anpassung der exportierten Sensorrohdaten möglich. Die Angaben Start und Stop der Messung müssen lediglich um die Verschiebung (z. B. 5 Sekunden) in der Datei precalib.ini angepasst werden.
Ebenso ist es möglich, dass versehentlich Label-Daten gespeichert wurden, die über den Messzeitraum des Sensorgürtels hinaus gehen. Dies führt zu einem Fehler beim Import der Daten und erfordert eine Korrektur der Daten. Dies ist durch Anpassung/Löschen einzelner Label in den Label-Rohdaten möglich, sodass diese nur innerhalb des Messzeitraums liegen.

Die während der Studie von den Senioren durchgeführten Aktivitäten waren spontan und oft schwer zu vorhersehen. Zudem erfordert die Bedienung der App ein gewisses Maß an Übung. Die mittels der App dokumentierten Aktivitäten haben daher nur eine begrenzte Genauigkeit, sodass sie lediglich als Referenz für die finalen „SeniorHomeLabel“ genutzt werden können.
Zur Unterstützung der manuellen Labelbearbeitung erfolgt semiautomatisches Labeln als Labelvorbereitung wie in Kapitel 4.1 beschrieben. Durch den Vorverarbeitungsschritt ist es möglich das Label Drehen als finales Label in einer extra Datei ebenso wie die Label „ANY_STATIC“ und „ANY_DYNAMIC“ in der Datei „calculated-Label“ zu erzeugen.

Die weitere Nachbearbeitung der Label erfolgt entsprechend der Standards definiert unter Kapitel 3.4.3. Um möglichst genaue Label zu erstellen, wurde die in Kapitel 5 beschriebene biomechanische Analyse der einzelnen Aktivitäten durchgeführt. In der erstellten Label-Datei „SeniorHomeLabel“ sind analog die folgenden Aktivitäten zu finden:

- Gehen

- Stehen
- Sitzen
- Treppensteigen
- Treppensteigen runter
- Liegen (Rücken)
- Liegen (Seite)
- Hocken
- Stehen-Sitzen
- Sitzen-Stehen
- Liegen-Sitzen
- Sitzen-Liegen

Bereiche, die nicht eindeutig einem der genannten Label zugeordnet werden konnten, wurden frei gelassen oder wenn aus den Androiddaten weiteren Informationen vorhanden waren, z. B. Staubsaugen oder Fahrradfahren, diese als Unbekannt-Label mit Kommentar übernommen. Unklarheiten bezüglich der Daten und möglicher Label wurden zum Erfahrungsaustausch in den wöchentlichen Meetings besprochen. Alle Label wurden durch eine zweite Person überprüft. Um den Fortschritt der manuellen Labelbearbeitung nachvollziehen zu können, wurden Fortschritt und bei Bedarf Anmerkungen in einer Tabelle im Confluence dokumentiert.

Die für die manuelle Labelbearbeitung verwendeten Daten wurden im Netzwerkordner unter /pgimu$/PG2017/Daten/SeniorHome-Studie/ abgespeichert. In den jeweiligen Unterordnern befinden sich nach SHID sortiert die Daten wie folgt:

- In „Rohdaten“ liegen die Originaldaten, welche mittels Sensorgürtel und App aufgenommen und exportiert wurden.

- In „Binärdaten“ ligen die für die manuelle Labelbearbeitung vorbereiteten Daten (in die MAMSK-Software importiert und Label aus semiautomatischem Labeln).

- In „Fertig_gelabelte_Daten“ wurden sowohl die fertigen „SeniorHomeLabel“ abgelegt sowie alle weiteren generierten Label-Dateien.

6.5. Gewonnene Erkenntnisse

Die Vorbereitung, Durchführung und Nachbereitung der Labelstudie war sehr zeitintensiv. Insgesamt wurden in einem Zeitraum von ca. sechs Monaten (Mitte August 2017 bis Ende Januar 2018) Besuche bei 36 Probanden durchgeführt. Von den in Kooperation mit den Studynurses vereinbarten Terminen wurde ca. ein viertel durch die Probanden wieder abgesagt oder verschoben. Durch eine ständige Rücksprache mit den Studynurses bei Problemen oder Unklarheiten konnte insgesamt ein weitestgehend reibungsloser Ablauf gewährleistet werden.
Von den 36 Datensätzen wurden 29 der zuvor beschriebenen manuellen Labelbearbeitung unterzogen und verwertbare S̈eniorHomeLabel”generiert. Zwei Datensätze wurden aufgrund verdrehter Sensordaten nicht weiter verarbeitet und ein Datensatz wurde für das Vorgehen aufgrund ungeeigneter Qualität der Daten und Androidlabel ausgeschlossen. Die letzten vier Datensätze wurden erst im Januar aufgenommen und aufgrund der fortgeschrittenen Zeit bezüglich des Projektplans nur noch als Rohdaten gespeichert und nicht weiter verarbeitet.
Die ausgewerteten 29 Datensätze haben eine Gesamtdauer von etwa 60 Stunden. Die SeniorHomeLabel enthalten die in Tabelle 6.1 aufgelisteten Label.

Die PG hat sich dazu entschieden, bei der Durchführung der Besuche die Probanden nicht bezüglich der durchgeführten Aktivitäten zu beeinflussen, um möglichst wenig Einfluss auf die Daten selbst zu nehmen (z.B. schnelles Gehen aufgrund von Leistungsdruck ähnlich der Assessments). Lediglich bei der Erläuterung der Studie wurde der Wunsch

Label	**Gesamtdauer**	**Prozent %**
Stehen	14H21M56.55S	22.9171
Sitzen	24H30M32.26S	39.0983
Liegen (Rücken)	44M19.38S	1.1785
Liegen (Seite)	8M13.29S	0.2186
Gehen	5H51M29.51S	9.3454
Treppensteigen	10M1.58S	0.2666
Treppenstigen runter	9M10.68S	0.2440
Hinunterbeugen/Hocken	19M22.83S	0.5153
Rückwärts gehen	1M13.48S	0.0326
Springen	2.74S	0.0012
Stehen-Sitzen	5M51.53S	0.1558
Stehen-Liegen	2.12S	0.0009
Sitzen-Stehen	5M7.53S	0.1363
Sitzen-Liegen	36.63S	0.0162
Liegen (Rücken)-Liegen (Seite)	0S	0.0000
Liegen (Seite)-Liegen (Rücken)	0S	0.0000
Liegen-Sitzen	28.54S	0.0126

Tabelle 6.1.: Übersicht der in den 29 SeniorHomeLabel-Dateien enthaltenen Label mit Dauer und Anteil in Prozent

nach möglichst unterschiedlichen Daten geäußert und auf Nachfrage der Probanden, ob noch Wünsche bezüglich bestimmter Aktivitäten bestehen, Beispiele gegeben. Dies hat zur Folge, dass wie in der obigen Tabelle dargestellt, die verwendbaren Daten bezüglich der verschiedenen Label in ihrer Menge sehr unterschiedlich sind. Während für Aktivitäten wie „Gehen“, „Stehen“ und „Sitzen“ mehrere Stunden an Daten gelabelt werden konnten, existieren für Aktivitäten wie „Springen“ oder „Stehen-Liegen“ nahezu keine Daten.

Die Dauer der einzelnen Besuche lag in Abhängigkeit von der Zeit / dem Wunsch der Probanden und der dokumentierbaren Label zwischen einer und vier Stunden. Insgesamt wurde immer versucht, die Probanden in ihrer Privatsphäre nicht zu sehr zu stören und die Besuche so lange durchzuführen wie die Probanden diesen nach empfinden der PG-Teilnehmer auch wünschten. Es wurden beispielsweise nur kürzere Besuche durchgeführt, wenn die Probanden nicht so viel Zeit hatten. Bei einigen Probanden stand dies bereits zu Beginn fest, bei einigen Probanden wurden die Besuche auch frühzeitig von den PG-Teilnehmern beendet, um keinen negativen Eindruck zu hinterlassen, da die Probanden z. B. trotz Erklärung kein Verständnis für die Notwendigkeit langer Datenaufnahmen oder der Besuche an sich hatten. Die Dauer der Besuche war ebenfalls von den Aktivitäten abhängig, die von den Probanden „angeboten“ wurden. War für die PG-Teilnehmer ersichtlich, dass im weiteren Verlauf des Besuchs keine neuen Daten, z. B. nur noch „Sitzen”, aufgenommen wird, wurde die Besuche nicht extra in die Länge gezogen. Einige Probanden haben die Termine auch so gewählt, dass die PG-Teilnehmer sie bei länger andauernden Tätigkeiten wie dem Chorsingen, Einkaufen oder sportlichen Aktivitäten begleiten konnten. Einige Aktivitäten, besonders Bewegungen beim Sport, konnten nicht direkt den gewählten Aktivitäten zugeordnet werden. Daher wurden eine Vielzahl der Aktivitäten, z. B. auch das Staubsaugen, Schuheputzen und Fahrradfahren mit dem Label Sonstiges und einem Kommentar versehen.

Durch die Probandenbesuche konnte für jeden Datensatz ein individueller Ablaufplan analog zu dem der Assessments erstellt werden, welche das Generieren von Referenzlabeln ermöglicht. Da die Daten noch eine gewisse Ungenauigkeit aufwiesen, zum einen durch die unvorhersehbaren Bewegungen der Probanden und die Reaktionszeit der PG-Teilnehmer ebenso wie durch die vorerst als „Sonstiges“ gelabelten Aktivitäten, konnten diese nur als Referenz verwendet werden. Um detaillierte und einheitliche Label erstellen zu können, war eine detaillierte Beschreibung der Aktivitäten und ihrem Einfluss auf die Sensoren notwendig. Um Fehler beim Labeln auszuschließen, wurden die Daten zudem mehrmals überprüft. Dennoch war es nicht möglich alle Bereiche eines Datensatzes zu Labeln. Dies war z. B. bei schnellen Abfolgen von „Gehen“ und „Stehen“ der Fall, da eine klare Trennung der Übergänge kaum möglich ist. Aber auch zahlreiche weitere Bereiche aus „Sonstiges“ oder nicht nachvollziehbarer Sensorwerte wurden auch im Hinblick auf die Verwendung der Daten für das Training der Modelle nicht gelabelt.

Insgesamt war es durch die genaue Festlegung von Standards für die Vorbereitung, Durchführung und manuelle Labelbearbeitung möglich, diese ohne größere Probleme durchzuführen und verwertbare „SeniorHomeLabel“ zu erstellen.

7. Implementierung

Im Laufe der Projektgruppenzeit sind verschiedene Implementierungen umgesetzt worden. Zur Studiendurchführung wurde eine Android App implementiert, es wurden Ergänzungen an der MAMKS Software realisiert und für das Training und das Klassifizieren der Daten wurden Programme in Matlab und Python implementiert.

7.1. Eine Android App als Online-Aktivitätentracker

Als Hilfsmittel bei der Durchführung der Studien wird eine Android Applikation verwendet. Mit dieser App werden die Aktivitäten der Senioren dokumentiert, um daraus zu den Gürteldaten, Labeldaten zu generieren. Im folgenden werden der Aufbau und die Funktionsweise der App beschrieben.
Die App teilt sich in drei Teilbereiche auf, die jeweils ein eigenes Layout haben. In der Android Programmierung spricht man hierbei von Activities. Die Activities der App „Aktivitätsdokumentation“ heißen: participant_info, activity_tracker und delete_activities.

7.1.1. Activity participant_info

Die Activity participant_info wird nach dem Start der App als erstes ausgeführt. Sie dient dazu, die Rohdaten ID des Probanden einzutragen. Dafür hat sie ein Eingabefeld und einen Startbutton. Die Ansicht der Activity ist in Abbildung 7.1 zu sehen.
Wenn der Start Button gedrückt wird, wird zunächst überprüft, ob die beiden Ordner existieren, in denen die Dateien für die Labeldaten gespeichert werden. Wenn sie noch nicht existieren, werden sie in diesem Schritt erstellt. Mit der App werden pro Proband

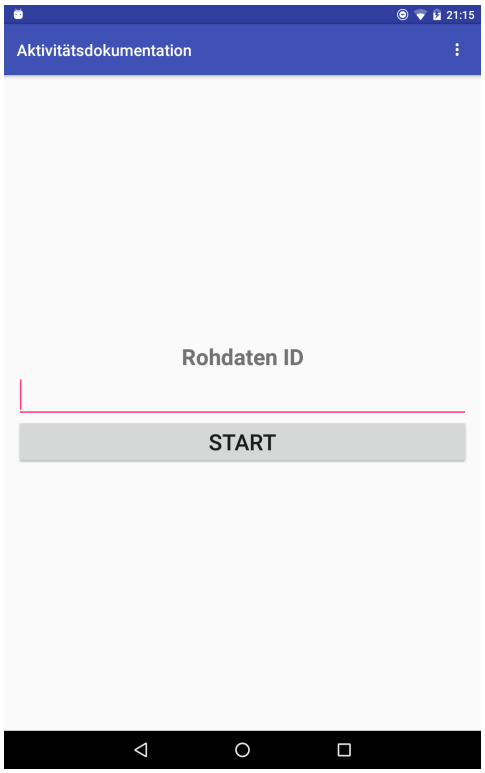


Abbildung 7.1.: Activity participant_info

zwei verschiedene Dateien gespeichert. In der ersten Datei werden die durchgeführten Aktivitäten gespeichert. In der zweiten Datei wird der Standort des Probanden gespeichert. Jede dieser Dateien wird in einem anderen Ordner gespeichert. Die erstgenannte Datei wird im Ordner „ActivityData“ und die zweite Datei wird im Ordner „LocationData“ gespeichert.
Als nächstes wird der Ordner „ActivityData“ auf vorhandene Dateien überprüft. Die Dateinamen werden mit der eingegebenen ID verglichen, um herauszufinden, ob zu einem Probanden bereits Dateien angelegt worden sind. Wenn keine Datei, mit dieser ID übereinstimmt, werden zwei neue Dateien angelegt. Beide Dateinamen bestehen aus der ID und dem Zeitpunkt des Speicherns der Datei. Existieren aber Dateien zu der eingegebenen ID, wird ein Dialogfenster geöffnet. In diesem wird gefragt, ob die Daten zu einer neuen Messung gehören oder nicht. Wenn dies mit ja beantwortet wird, werden ebenfalls zwei Dateien neu erstellt. Lautet die Antwort nein, wird anhand des Zeitstempels im Dateinamen die neuste der passenden Dateien ausgewählt. Anschließend wird die Activity activity_tracker gestartet.

7.1.2. Activity activity_tracker

In der Activity activity_tracker wird die Hauptfunktion der App durchgeführt. Hier werden Start- und Endzeitpunkte der einzelnen Aktivitäten in eine Datei gespeichert, um daraus Labeldaten zu generieren.
Die Activity hat in der Titelleiste einen Button „Löschen“. Unterhalb der Titelleiste sind drei Buttons nebeneinander. Sie haben die Beschriftungen „Zuhause“, „Draußen“ und „Drinnen“. Unter dieser Buttonreihe befinden sich vier Viererreihen mit Buttons, eine für jede der Aktivitäten „Rückwärtsgehen“, „Springen“, „Treppe hinaufsteigen“, „Sonstiges“, „Gehen“, „Drehen“, „Treppe hinaufsteien“, „Hocken“, „Stehen“, „Sitzen“, „Liegen(Rücken)“, „Liegen(Seite)“, „Aufstehen“, „Hinsetzen“, „Sitzen-Liegen“ und „Liegen-Sitzen“. Jeder dieser Buttons hat einen roten Hintergrund. Darauf ist ein Piktogramm der jeweiligen Aktivität zu sehen. In der Mitte des Buttons ist der Name der Aktivi-

tät genannt. Oberhalb davon wird die Anzahl angezeigt, wie oft die jeweilige Aktivität bereits ausgeführt wurde. Unterhalb des Aktivitätsnamen wird die Dauer, welche die jeweilige Aktivität bereits ausgeführt wurde, angezeigt. Links der 16 Button zeigt eine Beschriftung, zu welcher Kategorie, die jeweilige Zeile an Aktivitäten gehört. Die Kategorien sind „statisch“, „dynamisch“ und „Transition“.
Unterhalb der 16 Buttons ist ein „Beenden“-Button. Das Layout dieser Activity ist in Abbildung 7.2 zu sehen.

Beim Start der Activity werden die Anzahl und Dauer der jeweiligen Aktivitäten aus

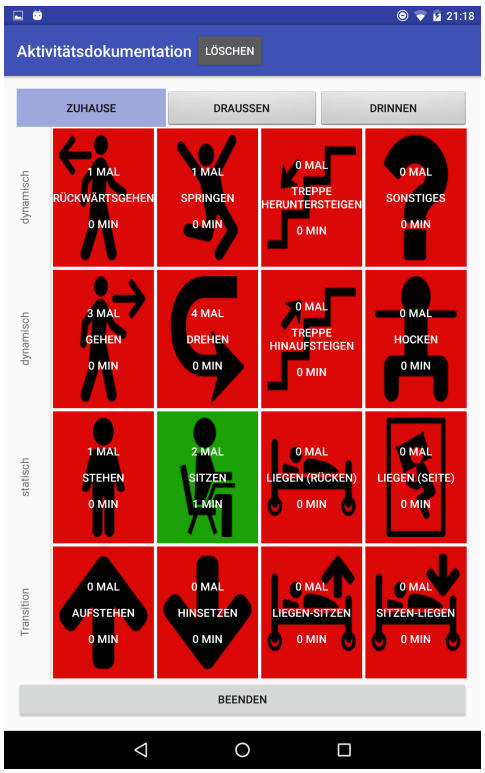


Abbildung 7.2.: Activity activity_tracker

den Dateien des ausgewählten Probanden ausgelesen. Die Dauer wird in Millisekunden berechnet. Um die Darstellung übersichtlicher zu halten, wird sie jedoch auf Minuten gerundet angezeigt.
Falls die Datei nur unterbrochen und nicht neu gestartet wurde, wird aus einem Zwischenspeicher ausgelesen, welcher Proband vor der Unterbrechung ausgewählt war. Eben-

falls wird die Aktivität und der Startzeitpunkt der Aktivität ausgelesen, die vor der Unterbrechung ausgewählt war. Der Hintergrund dieser Aktivität wird dabei grün gefärbt. Beim Klick auf eine der 16 Aktivitätenbuttons wird die zugehörige Aktivität ausgewählt. Es kann zu jedem Zeitpunkt nur einer der 16 Buttons ausgewählt sein. Für diese Aktivitäten gibt es drei Fälle die auftreten können. Im ersten Fall war vorher keine Aktivität ausgewählt. Die angeklickte Aktivität wird die aktive Aktivität. Es wird ein Zeitstempel für den Startzeitpunkt gespeichert und der Hintergrund des Buttons färbt sich grün. Im zweiten Fall war die angeklickte Aktivität vorher bereits ausgewählt. In diesem Fall wird der Hintergrund des Buttons wieder rot. Es wird ein Zeitstempel für den Endzeitpunkt gespeichert und die Aktivität erhält einen Eintrag in die Datei mit dem Aktivitätsnamen, dem Startzeitpunkt und dem Endzeitpunkt. Nun ist keine Aktivität mehr ausgewählt. Im dritten Fall war vorher eine andere Aktivität ausgewählt. Die Hintergrundfarbe der vorher ausgewählten Aktivität ändert sich rot. Der Zeitpunkt wird als Endzeitpunkt gespeichert und ein Dateieintrag erfolgt. Nun ist die neu angeklickte Datei ausgewählt. Der Hintergrund wird grün und der Zeitpunkt wird als Startzeitpunkt gespeichert. Wenn die Aktivität „Sonstiges“ ausgewählt wird, wird im ersten und dritten Fall nach den oben genannten Schritten ein Dialogfenster geöffnet. Hier wird der Nutzer zur Eingabe eines eigenen Aktivitätennamen aufgefordert. Soll kein Name eingegeben werden, kann der Nutzer den Dialog mit Nein schließen.

Die Location Buttons funktionieren analog zu den Aktivitätsbuttons. Auch hier kann jeweils nur einer der drei Buttons aktiv sein.

Beim Klick auf den „Beenden“ Button wird die ausgewählte Aktivität beendet und der Eintrag in die Datei geschrieben. Anschließend wird die Activity participant_info gestartet.

Beim Klick auf den „Löschen“-Button wird die Activity delete_activities gestartet.

7.1.3. Activity delete_activites

Die Activity delete_activities dient dazu einzelne Einträge aus der Datei zu löschen, falls bei der Aktivitätsdokumentation ein Fehler passiert sein sollte.
Hier werden alle bisherigen dokumentierten Aktivitäten mit ihrem Namen und ihrem Start- und Endzeitpunkt in chronologischer Reihenfolge in einer scrollbaren Liste angezeigt. Unterhalb dieser Liste gibt es zwei Buttons: „Löschen“ und „Zurück“. Dies ist in Abbildung 7.3 zu sehen.

Wenn der Nutzer auf einen der Listeneinträge klickt, wird dieser blau markiert und die

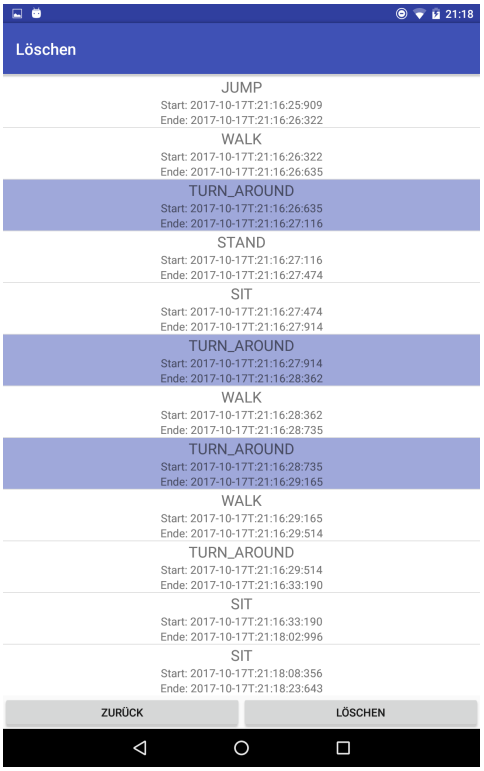


Abbildung 7.3.: Activity activity_tracker

zugehörige Aktivität wird einer zusätzlichen Liste mit zu löschenden Einträgen hinzugefügt. Es können auch mehrere Einträge gleichzeitig markiert sein. Wird ein Eintrag wieder abgewählt, färbt er sich wieder weiß und wird aus der Liste mit zu löschenden Einträgen entfernt.

Beim Klick auf den „Löschen“ Button wird dieser ausgegraut, um einen Mehrfachklick zu vermeiden. Die markierten Aktivitäten werden aus der Datei gelöscht und die Activity activity_tracker wird gestartet.
Beim Klick auf den „Zurück“-Button wird die Activity activity_tracker gestartet.

7.2. Mamks Extension

Im Laufe der Projektgruppe MAMKSFZ wurde mit der Software MAMKS gearbeitet. Im Gebrauch von MAMKS sind ein paar Verbesserungsideen aufgekommen, die im folgenden beschrieben werden.

7.2.1. Labelimport

In unserer Projektgruppe wurden Daten mit Hilfe eines Android Tablets annotiert. Die Tablets generierten dabei XML Dateien, in denen die Label abgespeichert sind. In Listing 8.1 ist ein Beispiel für eine Datei zu sehen. Diese abgespeicherten Label sollen möglichst automatisch in die MAMKS Anwendung integriert werden, sodass sie dort bearbeitet und verwendet werden können.

Listing 7.1: Beispiel einer Labeldatei

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<labelInformation>
<androidData/>
    <label>
        <interval>
            <start>2018-01-09T:10:32:24:545</start>
            <end>2018-01-09T:10:33:44:126</end>
```

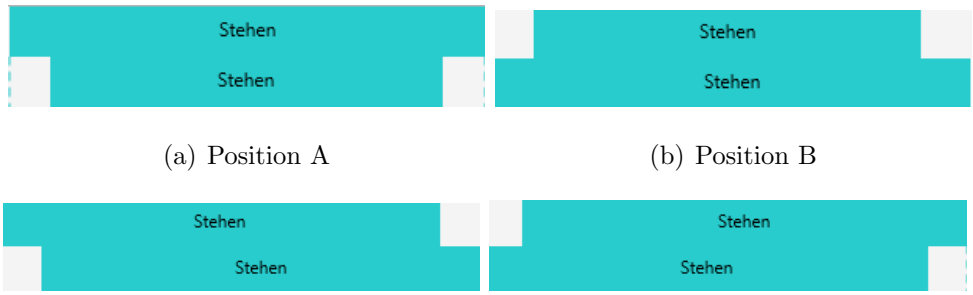
```
        </interval>
        <description>STAND</description>
    </label>
    <label>
        <interval>
            <start>2018-01-09T:10:33:44:126</start>
            <end>2018-01-09T:10:33:52:804</end>
        </interval>
        <description>WALK</description>
    </label>
</labelInformation>
```

Also haben wir die Funktion Labelimport in die Anwendung Mamks integriert. Beim Klick auf "Label importieren"öffnet sich ein Dialogfenster, in dem eine Datei ausgewählt werden kann. Diese ausgewählte Datei wird auf ihre Gültigkeit überprüft. Dabei ist der erste Schritt die Überprüfung des Dateinamens auf die Endung ".xml". Anschließend wird der Aufbau der Datei mit einer Schemadatei (XSD-Datei) kontrolliert. So kann sichergestellt werden, dass die ausgewählte XML-Datei auch wirklich Labeldaten enthält. Die Datei muss mit dem Tag <androidData/> starten. Danach werden die einzelnen Label angegeben. Jedes Label muss ein Intervall, eine Bezeichnung und ein Kommentarfeld enthalten. Jedes Intervall muss einen Start- und Endzeitpunkt enthalten.
Wenn der Aufbau der Datei dem Schema entspricht, wird als letztes geprüft, ob die Label vom Zeitstempel aus zu den Gürteldaten passen. Nur wenn die Datei allen Voraussetzungen entspricht, wird es zugelassen, dass der Nutzer seine Wahl bestätigen kann. Nach der Bestätigung der Datei werden alle Label aus der Datei ausgelesen. Dafür wird eine neue Labelgruppe erstellt, die den gleichen Namen trägt, wie die Datei. Anschließend wird jedes der Label aus der Datei, zu dieser Labelgruppe hinzugefügt und das Dialogfenster wird geschlossen.

7.2.2. Labelbearbeitung

Die Label, die mit den Android Tablets annotiert wurden, sind ungenau und fehlerhaft. Also müssen sie nachbearbeitet werden. Um die Labelnachbearbeitung zu beschleunigen, haben wir einige Funktionen in MAMKS eingefügt.
Mit der Maus können ganze Label oder auch nur die Ränder der Label verschoben werden. Dafür beobachtet ein MouseEventListener die Mausbewegungen des Nutzers. Wenn ein MouseDraggedEvent ausgelöst wird und die Maus sich dabei in einem Randbereich eines Labels befindet, werden die Ränder des Labels verschoben. Ist die Maus dagegen im Bereich eines Labels, aber nicht in einem Randbereich, so wird das ganze Label verschoben. In beiden Fällen wird zunächst der Abstand von der neuen Mausposition zu der ursprünglichen Mausposition vor dem Ziehen der Maus berechnet. Dieser Abstand wird in einen Zeitunterschied umgewandelt, um den der Zeitbereich des Labels verändert werden soll.
Im Fall des Verschiebens eines ganzen Labels wird als nächstes überprüft, in welche Richtung es verschoben werden soll. Wenn es nach links verschoben werden soll, wird vom Startpunkt und vom Endpunkt des Labels, jeweils der Zeitunterschied abgezogen. Soll es nach rechts verschoben werden, wird zum Startpunkt und vom Endpunkt jeweils der Zeitunterschied addiert. An den neuen Start- und Endpunkten des verschobenen Labels wird nun überprüft, ob an den Stellen bereits ein anderes Label vorhanden ist. Wenn dies der Fall ist, wird das Label nicht verschoben. Sonst wird das zu verschiebende Label gelöscht und ein neues Label hinzugefügt, welches den neuen Start- und Endpunkt als Intervall hat.
Im Fall des Verschiebens der Labelränder wird zunächst festgestellt, ob sich die Maus auf dem linken oder rechten Rand befindet. Als nächstes wird die Richtung der Mausbewegung ermittelt. Es werden die neuen Start- und Endpunkte des Labels berechnet. Wenn die Maus auf dem linken Rand ist und nach links bewegt wurde, wird überprüft, ob an dem neuen Startpunkt bereits ein anderes Label vorhanden ist. Wenn dies der Fall ist, wird der neue Startpunkt auf den Endpunkt des anderen Labels plus 1 gesetzt,

sodass das Label an das andere Label anschließt. Wenn kein Label im Weg ist, wird das alte Label gelöscht und das neue Label mit dem neuen Start- und Endpunkt erstellt.
Wird der Labelrand von der linken Seite ausgehend nach rechts verschoben, wird überprüft, ob der neue Startpunkt größer als der alte Endpunkt ist. In diesem Fall wird das weitere Verschieben des Labelrands unterbunden, sodass das Label nicht Start- und Endpunkt tauscht. Wenn der neue Startpunkt kleiner als der alte Endpunkt ist, wird das alte Label gelöscht und das neue Label mit dem neuen Start- und Endpunkt hinzugefügt. Die Verschiebung des rechten Labelrands funktioniert analog.
Zwischen zwei Labeln können Lücken vorhanden sein. Um diese Lücken schnell zu füllen, soll sich durch einen Doppelklick auf den leeren Bereich der Dialog zum Hinzufügen eines Labels öffnen. Der Doppelklick wird durch einen MouseEventListener festgestellt. Dabei muss der primäre Mausbutton gedrückt werden, also die linke Maustaste, und ein Klickzähler muss 2 anzeigen. Wenn ein Doppelklick registriert wurde, wird geprüft, ob sich an der Stelle ein Label befindet. Wenn das nicht der Fall ist, wird nach dem letzten Label vor und nach dem ersten Label nach dem geklickten Punkt gesucht. Aus den Labeln wird das Intervall des leeren Bereiches ermittelt (vom Endpunkt des vorherigen zum Startpunkt des nachfolgenden). Mit dem Intervall wird nun der Dialog zum Hinzufügen eines Labels geöffnet. In den Feldern für das Intervall ist das freie Intervall bereits eingetragen.
Wenn durch die Markierung eines Gürteldatenbereiches ein neues Label erstellt werden soll, kann es vorkommen, dass sich das neue Label mit einem anderen Label überschneidet. Bisher wurde in diesem Fall eine Fehlermeldung angezeigt und das Label nicht erstellt. Durch unsere Änderung wird nun ein Auswahlmenü geöffnet, durch das der Nutzer entscheiden kann, ob das alte Label überschrieben werden soll. Je nach Position des konfliktären Labels muss beim Überschreiben unterschiedlich vorgegangen werden. Auf Abbildung 7.4 sind die möglichen Positionen aufgezeigt. Das alte Label ist stets oben dargestellt und das neue Label ist unten dargestellt. Bei Position A wird das alte Label gelöscht und das neue wird hinzugefügt. Aber es werden auch noch zwei weitere Label hinzugefügt. Das eine geht vom ursprünglichen Start des alten Labels, zum Start



(a) Position A (b) Position B

(c) Position C (d) Position D

Abbildung 7.4.: Positionen des konfliktären Labels

des neuen Labels und das andere geht vom Endpunkt des neuen Labels, zum Endpunkt des alten Labels.

Bei Position B wird das alte Label gelöscht und das neue hinzugefügt. Bei Position C wird das alte Label gelöscht und das neue Label hinzugefügt. Zusätzlich wird ein Label vom Ende des neuen Labels zum Ende des alten Label hinzugefügt. Bei Position D wird das alte Label gelöscht und das neue Label hinzugefügt. Zusätzlich wird ein Label vom Start des alten Labels zum Start des neuen Labels hinzugefügt.

Bei der Labelkorrektur können leicht Fehler passieren. Um eine getätigte Änderung rückgängig zu machen, wurde ein Zurück-Button eingeführt. Beim Hinzufügen, Löschen oder Ändern eines Labels wird jeweils ein Eintrag in einer Liste, der LabelChangelogHistory Liste, hinzugefügt. Dieser Eintrag enthält das alte Label, das neue Label und die zugehörige Labelgruppe. Beim Hinzufügen eines Labels ist das Feld für das alte Label null. Um das Hinzufügen eines Labels rückgängig zu machen, muss also das Label, was im Feld für das neue Label vermerkt ist, aus der angegebenen Labelgruppe gelöscht werden. Beim Löschen eines Labels ist das Feld neues Label null. Um dies rückgängig zu machen muss das alte Label wieder hinzugefügt werden. Beim Ändern eines Labels werden alle Felder ausgefüllt. Um eine Änderung rückgängig zu machen, muss das neue Label gelöscht und das alte Label hinzugefügt werden. Nach dem Rückgängigmachen wird der Eintrag aus der Liste gelöscht. Insgesamt können zehn Einträge gleichzeitig in der Liste sein.

Um noch schneller Label ändern zu können, wird beim Rechtsklick auf ein Label oder

einen markierten Bereich ein Kontextmenü geöffnet. Die zuletzt hinzugefügten oder geänderten Aktivitätstypen werden angezeigt. Beim Klick auf einen Eintrag des Kontextmenüs wird ein Label des ausgewählten Types im ausgewählten Bereich hinzugefügt, falls der Rechtsklick auf dem markierten Bereich erfolgte. Sollten Konflikte auftreten wird wie oben erklärt gefragt, ob das andere Label überschrieben werden soll. Wenn der Rechtsklick auf einem vorhandenen Label erfolgte, wird durch die Auswahl einer Aktivität im Kontextmenü der Aktivitätstyp des Labels geändert.

7.2.3. Labelnachbearbeitung

Nach der Klassifizierung von Daten durch Machine Learning Modelle gibt es auch noch Verbesserungsmöglichkeiten der Labeldaten. Eine Möglichkeit ist die Mehrheitsentscheidung. Hierfür werden Klassifizierungen von mehreren Modellen verglichen. Es werden neue Label gebildet, deren MotionType mit dem der Mehrheit der Klassifizierungen entspricht. Eine weitere Möglichkeit ist eine regelbasierte Untersuchung der Klassifizierungen, genannt Rule Engine. Bestimmte Aktivitäten dürfen zum Beispiel eine bestimmte Dauer nicht unterschreiten oder sie folgen nur auf bestimmten Aktivitäten. Die beiden Möglichkeiten wurden unter dem Begriff Postprocessing oder Labelnachbearbeitung zusammengefasst.

In Abbildung 7.5 ist der Dialog zu sehen, mit dem beide Möglichkeiten angewandt werden können. Für beide Möglichkeiten ist es notwendig, zunächst die Labelgruppen auszuwählen, die für das Postprocessing verwendet werden sollen. Außerdem gibt es die Möglichkeit nur die Rule Engine, die Mehrheitsentscheidung oder beides auszuwählen. Für die Rule Engine muss eine XML-Datei, die die Regeln enthält, ausgewählt werden. Es wird anhand einer XSD Schemadatei überprüft, ob die Datei gültig ist. Nur wenn dies der Fall ist, ist es möglich den Okay Button zu klicken.

Für die Mehrheitsentscheidung kann der Nutzer einen Radius angeben. Der Radius entspricht der Anzahl der Datenpunkte, die für eine Berechnung der Mehrheit gleichzeitig betrachtet werden soll. Des Weiteren kann der Nutzer über das Feld Entscheidungskri-

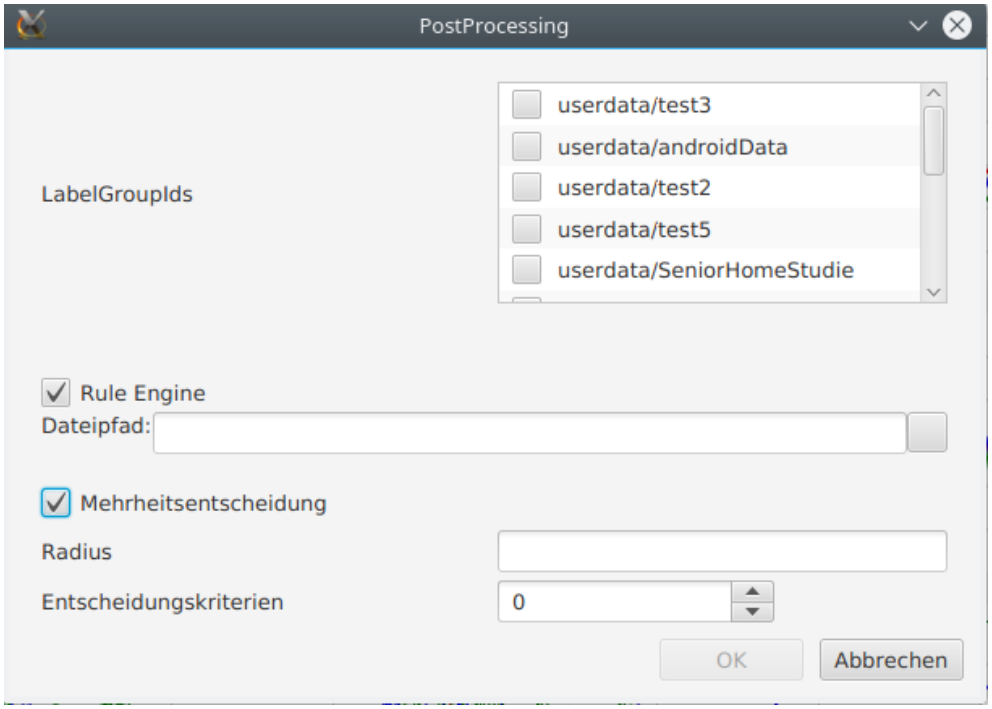


Abbildung 7.5.: Postprocessing Dialog

terium eine Prozentzahl angeben. Die Prozentzahl gibt an, wie viel Prozent der Datenpunkte mindestens mit dem gleichen Label belegt sein müssen. Bei einem Klick auf Okay werden die ausgewählten Möglichkeiten ausgeführt.

Beim Mehrheitsentscheid wird ein Fenster über den ganzen Datensatz geschoben. Die Fenstergröße entspricht dabei dem Radius. Für jeden Fensterausschnitt wird bestimmt, wie viele der enthaltenen Datenpunkte zu welcher Labelart gehören. Das wird in einer Liste gespeichert. Die Liste wird nach Häufigkeit des Vorkommens sortiert. Als neue Labelart wird die Labelart ausgewählt, zu der die meisten Datenpunkte gehören. Gibt es mehrere mit der gleichen Anzahl, wird eine zufällige Labelart ausgewählt. Es wird die relative Häufigkeit der Labelart bestimmt. Ist diese größer als das angegebene Entscheidungskriterium, wird die Labelart für das neue Label verwendet, sonst wird das Label Ünbekannt"verwendet. Ein neues Label wird in einer neuen Labelgruppe erstellt. Das Intervall entspricht dem aktuellen Fensterausschnitt. Wenn das Label zeitlich direkt vor dem neuen Label zur gleichen Labelart gehört, werden die beiden Label zusammenge-

fügt, um nicht zu viele sehr kleine Label entstehen zu lassen.
Bei der Rule Engine wird zunächst die Regeldatei eingelesen. Das passiert mit einem JAXB Provider. Das ist die Implementierung einer Architektur für XML Binding in Java. Der JAXB Provider wird dazu genutzt, um mit Hilfe der XSD Schemadatei die Regeln automatisch aus der XML Datei zu lesen und sie in Instanzen der Klasse Rule zu verwandeln. In Listing 8.2 ist der Aufbau einer solchen Regel zu erkennen.

Listing 7.2: Beispiel einer Regel

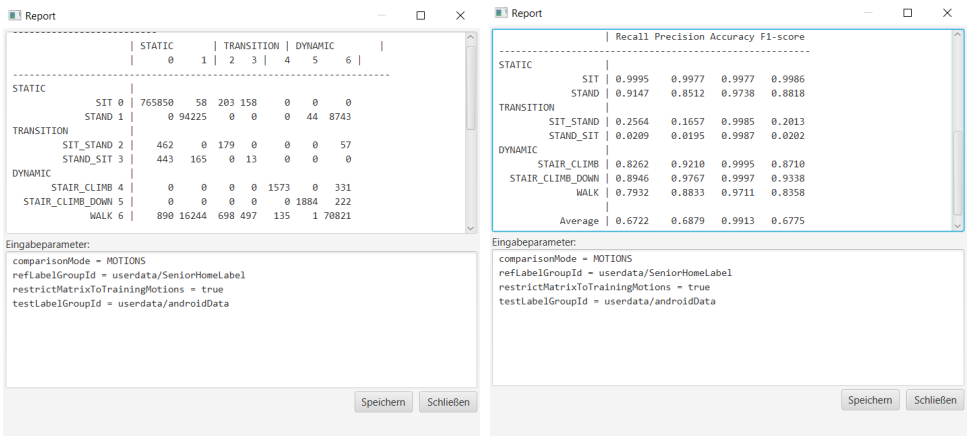
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ruleDefinition>
    <rule>
        <ruleID>001</ruleID>
        <activity>SIT</activity>
        <conditions>
            <minDuration>10</minDuration>
            <maxDuration>2000</maxDuration>
            <validPreviousActivity>
                <activity></activity>
            </validPreviousActivity>
            <validFollowingActivity>
                <activity></activity>
            </validFollowingActivity>
        </conditions>
        <action>
            <newActivity>UNKNOWN</newActivity>
        </action>
        <comment></comment>
    </rule>
</ruleDefinition>
```

Im nächsten Schritt werden die Regeln ausgewertet. Dabei wird für jede der ausgewählten Labelgruppen und für jedes Label dieser Labelgruppen ermittelt, ob eine Regel für diese Labelart vorhanden ist. Dafür wird der MotionType des aktuellen Labels mit dem MotionType, der im Feld activity angegeben ist, verglichen. Wenn dies der Fall ist wird zuerst das zeitlich gesehen vorherige und das nachfolgende Label ermittelt. Es wird überprüft, ob das vorherige und das nachfolgende Label jeweils einem der in der Regel als gültig angegebenen MotionTypes entspricht (validPreviousActivity und validFollowingActivity). Außerdem wird darauf geachtet, ob das betrachtete Label von der Dauer eine angegebene Mindestdauer (minDuration) nicht unterschreitet und eine angegebene Maximaldauer (maxDuration) nicht überschreitet. Wenn alle Regeln erfüllt sind, wird einer neu erstellten Labelgruppe das Label genauso hinzugefügt wie es bereits vorhanden war. Wenn eine Regel nicht erfüllt ist, wird das Label mit einem anderen MotionType hinzugefügt. Dieser ist in der Regeldatei unter action angegeben.

7.2.4. Confusion Matrix Report

Zur besseren Visualisierung wurde die bisherige Ansicht der Konfusionsmatrix überarbeitet. Dazu wurde die bisherige Darstellung, die aus einer reinen Schriftform bestand durch eine Tabelle ersetzt, die einer Konfusionsmatrix mehr gerecht wird. In der Java Klasse MainApplication wird in der showReport Methode nun eine weitere Unterscheidung darüber gemacht ob der Report einer Instanz einer ConfusionMatrixReport Klasse entspricht. Wenn dieser Fall eintritt, wird die neue showConfusionMatrixReportDialog Methode aufgerufen welchen die jeweilige FXML und dazugehörige Controller Klasse instantiiert. Die neue Konfusionsmatrix besteht nun nicht mehr aus einem großen Textfeld das alle Ergebnisse abbildet, sondern auch zwei Tabellen die zum einen die Konfusionsmatrix und zum anderen die Ergebnisse der Genauigkeit abbildet. Zur besseren Übersicht werden die Tabellen in einem TabPane getrennt voneinander abgebildet. Da alle Daten aus der bereits implementiertem ConfusionMatrix Klasse vorhanden waren, bestand die Aufgabe hierbei die Daten in einer ansprechenderen Form abzubilden. Dazu wurden

zur Umsetzung der Tabellen der einfachheitshalber TableView Objekte genutzt dessen TableColumns dynamisch in Abhängigkeit der Größe der Konfusionsmatrix erzeugt werden. Lediglich die Breiten der Spalten der Genauigkeiten sind mit Ausnahme der ersten Spalte statisch festgelegt, da sich die Anzahl der Spalten in diesem Fall nicht ändert. Um den Fokus der diagonalen Werte der Konfusionsmatrix hervorzuheben, werden die Werte in hellgrüner Farbe hervorgehoben. Wenn der Nutzer nun auf den Button Speichern drückt, wird er dazu aufgefordert, einen Ordner zu wählen, in dem nun die beiden Tabellen als Bilddatei abgelegt werden. Dazu werden von den beiden TableViews jeweils Screenshots gemacht und als Bilddatei abgespeichert. Die Namen der Dateien lauten confusionMatrix(aktuelles Datum und Uhrzeit).png und PrecisionMatrix(aktuelles Datum und Uhrzeit).png. Der Zeitstempel dient einfachheitshalber der Kennung, wann die Bilder erzeugt wurden und zur besseren Verwaltung.



(a) alte Konfusionsmatrix (b) alte Genauigkeiten

	SIT	STAND	SIT_STAND	STAND_SIT	STAIR_CLIMB	STAIR_CLIMB...	WALK
SIT	765850	58	203	158	0	0	0
STAND	0	94225	0	0	0	44	8743
SIT_STAND	462	0	179	0	0	0	57
STAND_SIT	443	165	0	13	0	0	0
STAIR_CLIMB	0	0	0	0	1573	0	331
STAIR_CLIMB_...	0	0	0	0	0	1884	222
WALK	890	16244	698	497	135	1	70821

(c) neue Konfusionsmatrix

	Recall	Precision	Accuracy	F1-Score
SIT	0,9995	0,9977	0,9977	0,9986
STAND	0,9147	0,8512	0,9738	0,8818
SIT_STAND	0,2564	0,1657	0,9985	0,2013
STAND_SIT	0,0209	0,0195	0,9987	0,0202
STAIR_CLIMB	0,8262	0,921	0,9995	0,871
STAIR_CLIMB_DOWN	0,8946	0,9767	0,9997	0,9338
WALK	0,7932	0,8833	0,9711	0,8358
Average	0,6722	0,6879	0,9913	0,6775

(d) neue Genauigkeiten

Abbildung 7.6.: Darstellung der alten und neuen Konfusionsmatrix

7.2.5. MATLAB Engine

Während der Phase der Datenerhebung wollten die Projektgruppenteilnehmer gerne die Möglichkeit haben, aus der MAMKS Software heraus, MATLAB Skripte auszuführen. Dazu wurde die Möglichkeit gebraucht die offizielle MATLAB Schnittstelle zu nutzen, welche seit der Version 2016b verfügbar ist. Dazu muss das Root-Verzeichnis von MATLAB als Umgebungsvariable mit den Namen MATLAB-HOME gesetzt und das bin-Verzeichnis in der Path-Umgebungsvariable hinzugefügt werden. Als Dependency wird die Umgebungsvariable in der MAMKS Software eingelesen, mit der die MATLAB API nun genutzt werden kann. Eine umfangreiche Beschreibung kann auf der Website von der MATLAB Java Seite gefunden werden. Da die Anbindung an die MATLAB Engine einige Zeit erfordert, wird diese über einen Thread beim Programmstart ausgeführt und bei erfolgreichem Start die Nachricht im unteren Statustextfeld der MAMKS Software wiedergegeben. Daraufhin wird der Menü Eintrag mit dem Namen MATLAB Engine sichtbar, welcher beim Anklicken zum eigens geschriebenen MATLAB Engine Dialog führt. Über den Dialog wird im obersten Textfeld der Pfad zu den MATLAB Skripten angegeben, die danach über das darunterliegende DropDown Menü ausgewählt werden können. Wenn man den Button ausführen drückt, wird das MATLAB Skript über die Eval Funktion der MATLAB Engine ausgeführt und der Name des aktuellen Kommandos an die MATLAB Engine im untersten Textfeldes angezeigt. Der Dialog ist in Abbildung 7.7 zu sehen. Sobald das Skript komplett durchgelaufen ist und das entsprechende im Skript definierte Ergebnis erzeugt wurde, wird im untersten Textfeld durch die Meldung DONE! angezeigt, dass das Skript erfolgreich ausgeführt wurde. Da im weiteren Verlauf des Projektes eine Anbindung an Python realisiert wurde, wurde in der Gruppe beschlossen, beide Schnittstellen über eine einheitliche Kommandozeile zu bedienen. Daher sollte die oben beschriebene MATLAB Engine keine weitere Anwendung mehr finden, da unter unter anderem die MATLAB Engine nur aus der frontend-Anwendung von MAMKS ausgeführt werden kann und nicht durch das zugehörige Kommandozeilenprogramm. Im Laufe der Implementierung der Kommandozeilenschnittstelle stellte sich jedoch heraus,

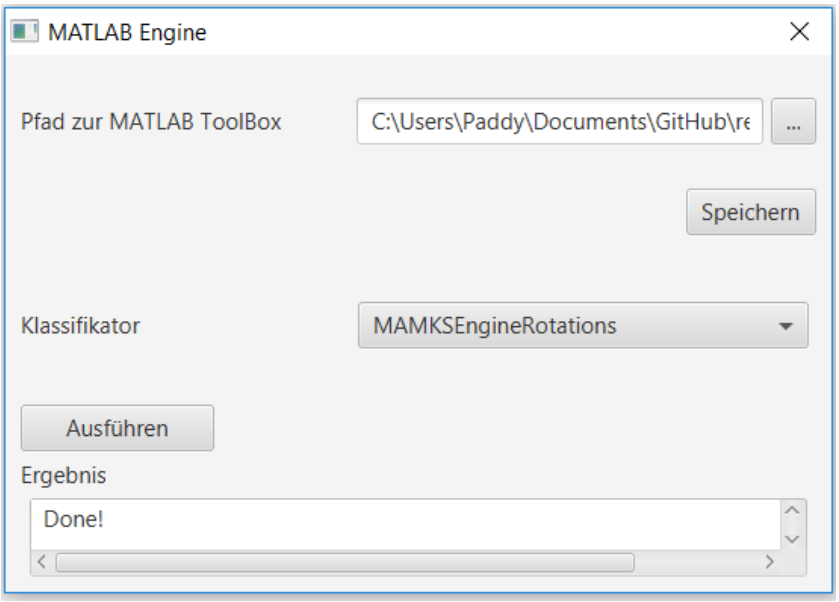


Abbildung 7.7.: MATLAB Engine Dialog mit erfolgreich ausgeführtem Skript

dass dies nicht ohne Probleme auf allen Betriebssystemen umsetzbar war. Aus diesem Grund wird die MATLAB Engine auf allen Betriebssystemen außer Windows weiterhin verwendet.

7.2.6. Statistik Modul

Um den Vorgang der Labelnachbearbeitung effizienter zu gestalten, entstand aus der Projektgruppe der Wunsch, eine Möglichkeit zu haben, sämtliche Label und Labelgruppen gegenüber zu stellen und zu visualisieren. Das Problem war bisher die fehlende Funktionalität in MAMKS, alle Informationen über die Verteilung von Häufigkeiten und Zeiten der Labels und Labelgruppen eines Datensatzes zu erhalten. Deshalb bestand die Lösung darin, ein Statistik Modul zu implementieren und in MAMKS zu integrieren. Aufrufbar wird das Modul, sobald ein Datensatz in MAMKS geladen worden ist. Dazu wird das neue Item in der Menubar entsprechend eingeblendet (s. Abbildung 7.8). Die Anzeige des Moduls selber erfolgt durch ein Modales Fenster. Die Darstellung der Statistik erfolgt zum Einen durch ein Bar-Chart Diagramm, welches in der Mitte des Moduls

Abbildung 7.8.: Statistik Modul

abgebildet ist. Zum anderen werden auf der linken Seite sämtliche Labelinformationen anzeigt, die zu diesem Datensatz gehören (s. Abbildung 7.9). Das Bar-Chart Diagramm enthält auf seiner X-Achse alle vorhandenen Label und auf der Y-Achse die jeweiligen Häufigkeiten im Datensatz. Die Häufigkeit werden abhängig von ihren jeweiligen Labelgruppen betrachtet. Die Labelgruppen und ihre Häufigkeiten werden nebeneinander dargestellt. Damit soll die Möglichkeit erreicht werden alle Informationen innerhalb eines Fensters einsehen zu können. Beim ersten Aufruf wird das Bar-Chart-Diagramm mit allen Labeln aus allen Labelgruppen angezeigt. Um den Bedürfnissen von spezieller Darstellung entgegenzukommen, sind alle Labelgruppen und Label mit Checkboxen versehen, mit denen die Anzeige im Bar-Chart Diagramm ein und ausgeschaltet werden kann. Die Anpassungen erfolgen unmittelbar und können beliebig oft ausgeführt werden. Dadurch sind Anpassungen der Darstellung nach den jeweiligen Bedürfnissen möglich. Auf der linken Seite sind sämtliche Label untereinander in einer VBOX in einem ScrollPane aufgereiht um im Falle von vielen Labels eine Scroll-Funkion zu haben. Unter jedem Label sind die Informationen über die jeweilige Gesamtdauer in der entsprechenden Labelgruppe hinterlegt. Im weiteren Verlauf des Projektes entstand der Wunsch, die

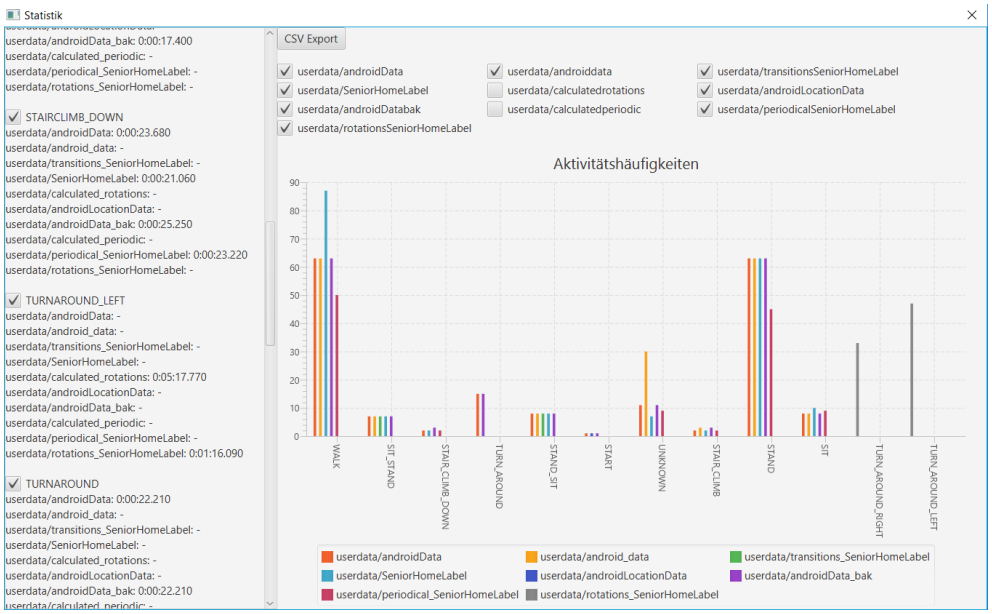


Abbildung 7.9.: Darstellung der Statistiken

Y-Achse der Häufigkeiten eines Labels durch die Gesamtdauer eines Labels zu ersetzen. Um diese Funktionalität auf einfachster Ebene umzusetzen, wurde ein zweites Bar-Chart Diagramm implementiert welches zu Beginn noch deaktiviert ist. Durch einen Mausklick wird die Sichtbarkeit des aktuellen Bar-Chart-Diagramms deaktiviert und stattdessen das Diagramm mit der Angabe der Gesamtdauer in Minuten angezeigt (s. Abbildung 7.10). Für den Benutzer wird dabei der Eindruck geweckt, dass es sich lediglich um eine Statistik handelt, die nur die Y-Achse wechselt.

7.2.7. Ausführung externer Skripte

Für die Ausführung von Klassifizierungsalgorithmen, die auf Python oder Matlab basieren, wird eine Möglichkeit benötigt, diese aus der MAMKS Software heraus auszuführen. Wichtig dabei ist, dass die Ausführung dieser Skripte im mamks-shared Bereich implementiert wird, damit diese Funktion nicht nur über die frontend-Anwendung, sondern auch über das Konsolenprogramm verfügbar ist.
Also wurde für die Ausführung der Skripte je Programmiersprache eine neue Unterklasse

Abbildung 7.10.: Statistiken mit Minutenangabe

der bereits bestehenden Algorithm Klasse hinzugefügt. In der Übersicht der Algorithmen tauchen nun MatlabExecution und PythonExecution auf(s. Abbildung 7.11). In der Algorithm Klasse werden für jeden Algorithmus Eingabe- und Ausgabeparameter spezifiziert. Für die Ausführung von Python Skripten werden die Eingabeparameter Min Score, Step Size, Window Size und die Pfade für den zu benutzenden Workspace, für die zu nutzende Modelldatei und für das auszuführende Skript benötigt. Da die Pythonskripte so konzipiert sind, dass sie einen Workspace Pfad als Eingabe erwarten und dann alle Datensätze aus diesem Workspace in einem klassifizieren, wurde auf das Auswahlmenü für einzelne Datensätze verzichtet, welches bei den anderen Algorithmen eingesetzt wird. Es werden also immer alle Datensätze klassifiziert, die im angegebenen Workspace vorhanden sind. Um dem Nutzer die Auswahl der Parameter zu erleichtern, werden Defaultwerte eingesetzt. Für den Parameter Min Score liegt der Default Wert bei 0,6. Die Window Size und die Step Size haben jeweils den Default Wert 100. Der Workspacepfad ist standardmäßig zunächst der aktuelle Workspace, den die MAMKS Anwendung nutzt. Für die Modelldatei wird als Default, die zuletzt ausgewählte Datei angegeben. Für die Auswahl der Pythonskripte wird im Ordner, in dem die ausgeführ-

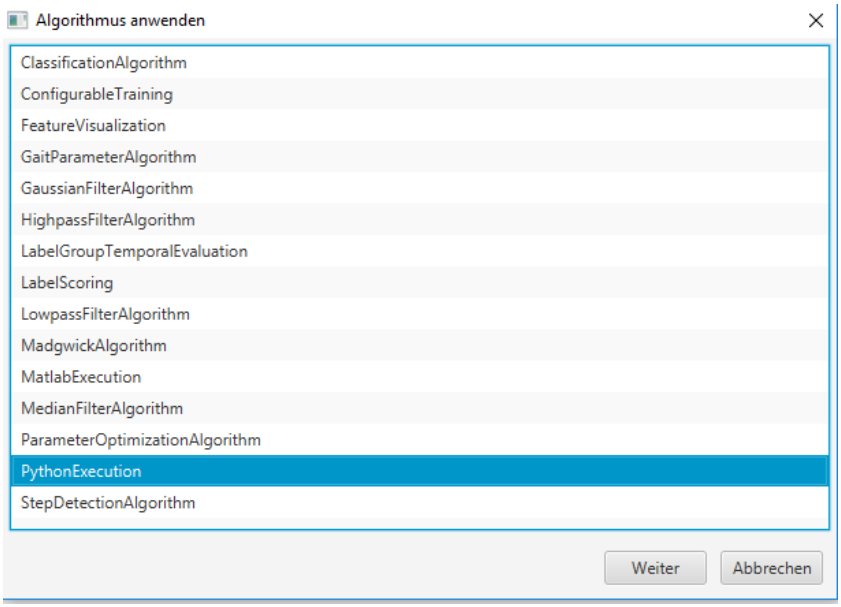


Abbildung 7.11.: Algorithmenauswahl mit neuen Elementen

te jar-Datei liegt nach einem Ordner ”Python”gesucht. Aus diesem Ordner werden alle Dateien, die mit dem String ”classify_mamks_data”beginnen in einem Auswahlmenü angezeigt, sodass der Nutzer nur funktionierende Skripte auswählen kann. Die Auswahl der Pythonskripte funktioniert damit jedoch nur, wenn die jar-Datei im gleichen Ordner ist, wie der Python Ordner mit den Skripten. Das Menü zur Parameterauswahl ist in Abbildung 7.12 zu sehen. Aus den eingetragenen Parametern wird nach dem Klick auf Okay ein String erstellt, der dem Konsolenbefehl für die Ausführung des Skriptes gleicht. Für die Ausführung eines Pythonskriptes ist dieser in Listing 7.3

Listing 7.3: Pythonbefehl

```
python $(Pythonskriptpfad)
       ws_path="$(Workspacepfad)"
       model_path="$(Modellpfad)"
       window_size="100"
       step_size="100"
       min_score="0.6"
```

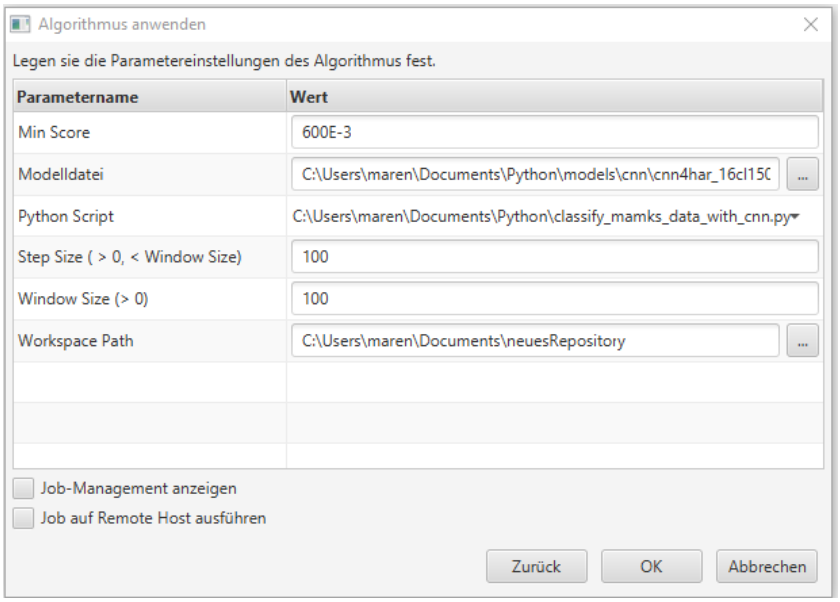


Abbildung 7.12.: Parameterwahlmenü für die Ausführung von Python Skripten

Dieser entstandene String wird mit Hilfe von der Java Klasse Process ausgeführt. Process ist eine abstrakte Klasse, die ein ausführendes Programm enthält. Dieses ausführende Programm wird von der Klasse Runtime gestartet. Runtime ist die Schnittstelle einer Java Anwendung zu ihrer Umgebung. Mit dem Befehl Runtime.getRuntime().exec() können Konsolenbefehle ausgeführt werden und somit auch der oben dargestellte Pythonbefehl.
Damit die Pythonskripte ausgeführt werden können, müssen auf dem Computer Python und verschiedene Pythonbibliotheken installiert sein. Da dies auf jedem Betriebssystem anders installiert wird, bietet die Skriptausführung kein automatisiertes Installieren aller notwendiger Pakete an. Allerdings prüft sie, ob ein möglicher Fehler in der Ausführung auf fehlende Installationen zurückzuführen ist und macht den Nutzer in der angezeigten Fehlermeldung darauf aufmerksam.

Für Matlabskripte werden in der Algorithm Klasse die Parameter Matlabskript-pfad und Workspacepfad festgelegt. Auch hier werden die zur Verfügung stehenden Skripte aus einem Ordner ausgelesen. In diesem Fall heißt der Ordner MATLAB.

Die ausführbare jar-Datei muss also im gleichen Ordner liegen, wie der Ordner MATLAB. Innerhalb des im MATLAB-Ordner liegenden Ordner mamks_interface werden alle Dateien zur Auswahl angezeigt. Als Workspace Pfad wird wie schon bei den Pythonskripten als Standard der aktuelle Workspace der MAMKS Anwendung angezeigt. In Abbildung 7.13 ist das Parameterauswahlmenü für die Ausführung von Matlabskripten zu sehen. Aus den eingegebenen Parametern wird der Matlabbefehl

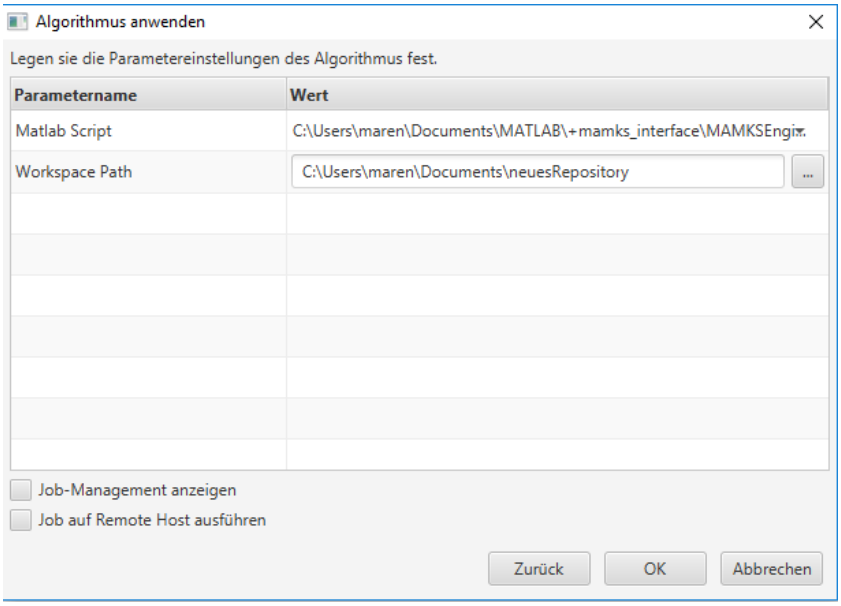


Abbildung 7.13.: Parameterwahlmenü für die Ausführung von Matlab Skripten

als String zusammengesetzt. Dieser ist in Listing 7.4 zu sehen. Bei der Ausführung des Befehls wird Matlab gestartet, das Skript ausgeführt und dann wird Matlab wieder geschlossen. Der Ausdruck '-nodesktop' sorgt dafür, dass für die Ausführung des Befehls nicht das Standardprogramm Matlab gestartet wird. Es wird lediglich eine Matlabkonsole geöffnet. Der Ausdruck '-r' bedeutet, dass ein festgelegtes Skript ausgeführt werden soll. Nach Ende der Ausführung soll die Matlab Konsole wieder geschlossen werden. Das wird mit 'quit' erreicht.

Listing 7.4: Matlabbefehl

```
matlab -nodesktop -r
        "$(Matlabskriptpfad)('$(Workspacepfad)')"
```

```
;quit
```

Der Matlabbefehl wird analog zum Pythonbefehl mit der Java Klasse Runtime ausgeführt. Damit das funktioniert muss Matlab auf dem genutzten Computer installiert sein. Anderenfalls wird der Nutzer auf das Fehlen der Installation in einer Fehlermeldung hingewiesen. Leider funktioniert diese Art der Ausführung von Matlabskripten nur auf Computern mit dem Betriebssystem Windows, da es auf anderen Systemen Probleme mit dem Starten von Matlab aus einem Java Programm gibt. Auf anderen Systemen sollte also auf die Matlab Engine zurückgegriffen werden, die weiter oben beschrieben wurde.
Ist in der MAMKS Software zur Zeit der Ausführung der Algorithmen bereits ein Datensatz angezeigt, sollen nach dessen Ende die neu klassifizierten Label ausgewählt und angezeigt werden können, ohne dass der Nutzer den Datensatz neu laden muss. Die Pythonskripte und Matlabskripte legen neue Labeldateien in den entsprechenden Ordnern der klassifizierten Datensätze an. Die MAMKS Software liest Dateien aus Ordnern jedoch nur beim Neuladen eines Datensatzes aus. Deshalb wurde die Software so geändert, dass das Laden von Daten aus den Ordnern nach Beendigung eines Algorithmus ebenfalls durchgeführt wird.

7.3. MATLAB

In diesem Kapitel sollen die Implementierungsdetails und Toolboxen erläutert werden, die für die Umsetzung der Klassifizierungsalgorithmen mit Matlab verwendet wurden.

7.3.1. Einlesen der MAMKS-Daten

Damit Matlab mit den binären Gürteldaten arbeiten kann, müssen diese erst in das richtige Dateiformat gewandelt werden. Auch der strukturelle Aufbau sollte dem vorab

definierten Konzept (siehe Kapitel 4) entsprechen. Um dies zu erreichen wurde in Matlab die Funktion load_mamks_data erstellt. Diese bekommt den Verzeichnispfad zu den SHIDXXXXX-Ordnern übergeben sowie Label und Channel (z.B. die Spalte Luftdruck), welche nicht Teil des Exports sein sollen und somit weggelassen werden. Außerdem wird ein Postfix übergeben, welcher die Labelexporte dementsprechend benennt. Mit Hilfe der Funktion wurden mehrere Trainingsdatensätze erstellt mit unterschiedlichen Labeln (siehe 8.15). Die jeweiligen Ergebnisse des Exports sind in mat-Dateien gespeichert. Inhalt ist eine Tabelle mit der Bezeichnung X. Um schnelle Information über diese Tabelle zu erhalten wurden zusätzlich die Anzahl der Zeilen und Spalten mit dem Befehl size(X) und die enthaltenen Labels mit dem Befehl unique(X.Label) im gleichen Verzeichnis in Form einer txt-Datei gespeichert.

7.3.2. Classification Learner App

Mit Hilfe der Classification Learner App ist es möglich, schnell eine breite Auswahl an Modellen zu starten. Dies ist insbesondere praktisch, um einen groben Überblick über die ungefähre Genauigkeit der Algorithmen zu bekommen. Auch kann eine begrenzte Anzahl an Parametern je Modell verändert werden. Vor dem Beginn der Pilot-Studie wurde dadurch getestet, inwiefern die verschiedenen Modelle auf die IMU-Datensätze reagieren.

Für die Durchführung haben wir zunächst den Datensatz eingelesen, welcher bereits durch die load_mamks_data.m-Funktion in die Benötigte Formatierung gebracht wurde. Dies ist mit dem Befehl load('Filename.mat') möglich. Anschließend kann die App über die App-Auswahl in Matlab gestartet werden. Nach dem Start kann über den Button New Session –> From Workspace ein neue Sitzung gestartet werden (siehe Abbildung 7.14).

Danach müssen in drei Schritten die erforderlichen Einstellungen für die Sitzung getätigt werden (siehe Abbildung 7.15). Auf der linken Seite sollte der richtige Datensatz ausge-

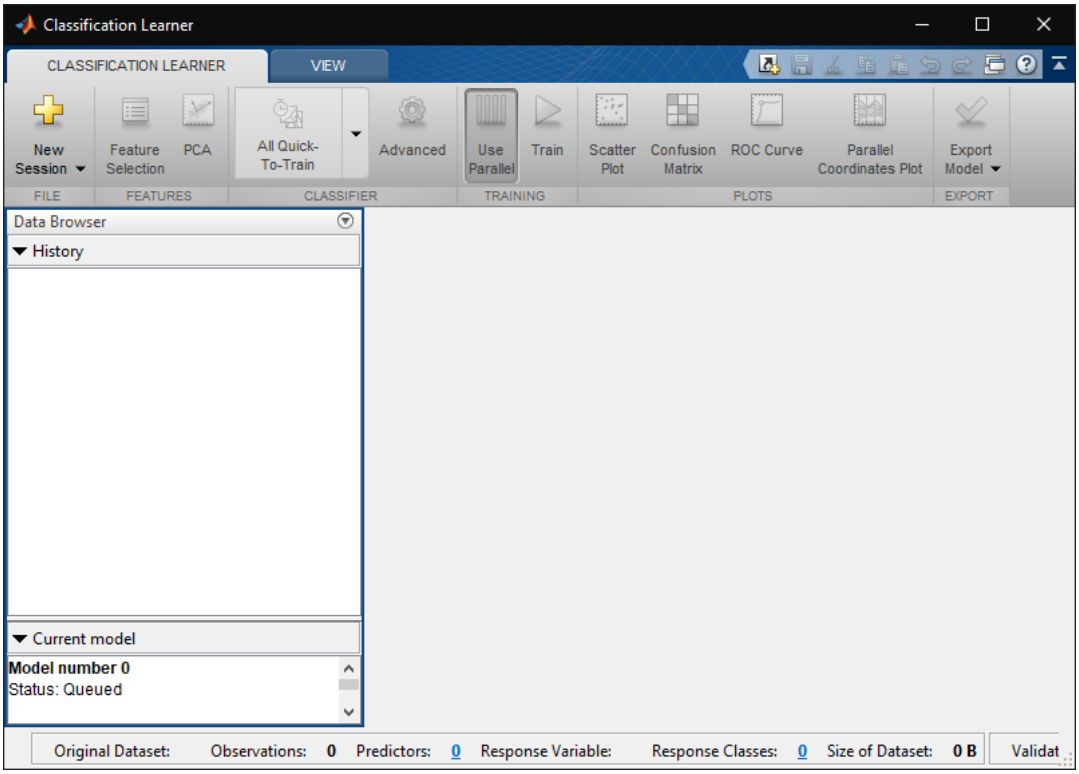


Abbildung 7.14.: Start einer neuen Sitzung

wählt werden. In der Mitte muss bestimmt werden, welche Spalten als Predictor gelten und welche die Response Spalte ist. Die Response Spalte war in unserem Fall Label. Auf der Rechten Seite kann schlussendlich die Form der Validierung angepasst werden. Dabei kann eine Kreuz-Validierung mit Festlegung der Folds gewählt werden oder einer Holdout-Validation mit Auswahl von fünf bis maximal 50 Prozent. Auch das weglassen einer Validierung ist möglich. Damit wir einen möglichst gleichen Validierungsstandard haben, wurde entschieden die 5-Folds Validierung zu nutzen.

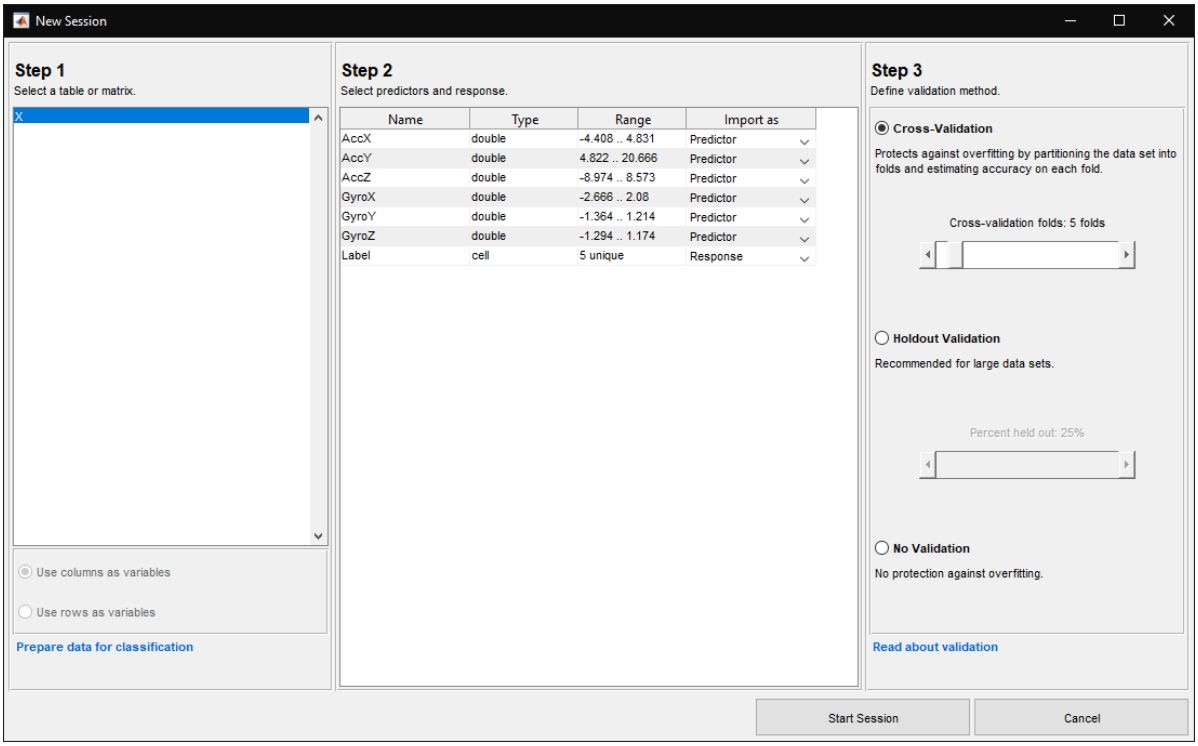


Abbildung 7.15.: Festlegung der Sitzungsparameter

Nach dem Klick auf Start Session wird das eigentliche Konfigurationsfenster geöffnet. Im oberen Bereich ist die Auswahl der zu trainierenden Modelle auswählbar (siehe Abbildung 7.16). Dabei sind auch mehrere Algorithmen einstellbar, welche im weiteren Schritt parallel antrainiert werden. Unter Advanced kann zudem bei fast jedem Modell eine Auswahl an Parametern gesetzt werden.

Eine Voranalyse bietet der dargestellte Scatter Plot, welcher die Verteilung der einzelnen Klassen je Attribut anzeigt. Dadurch lassen sich beispielsweise Ausreißer identifizieren. Mit einem Klick auf Train startet das Anlernen der Modelle. Dazu wird im Standardfall ein Lokaler Pool gestartet, welcher die Berechnung möglichst auf die zur Verfügung stehenden Rechenkerne verteilt.

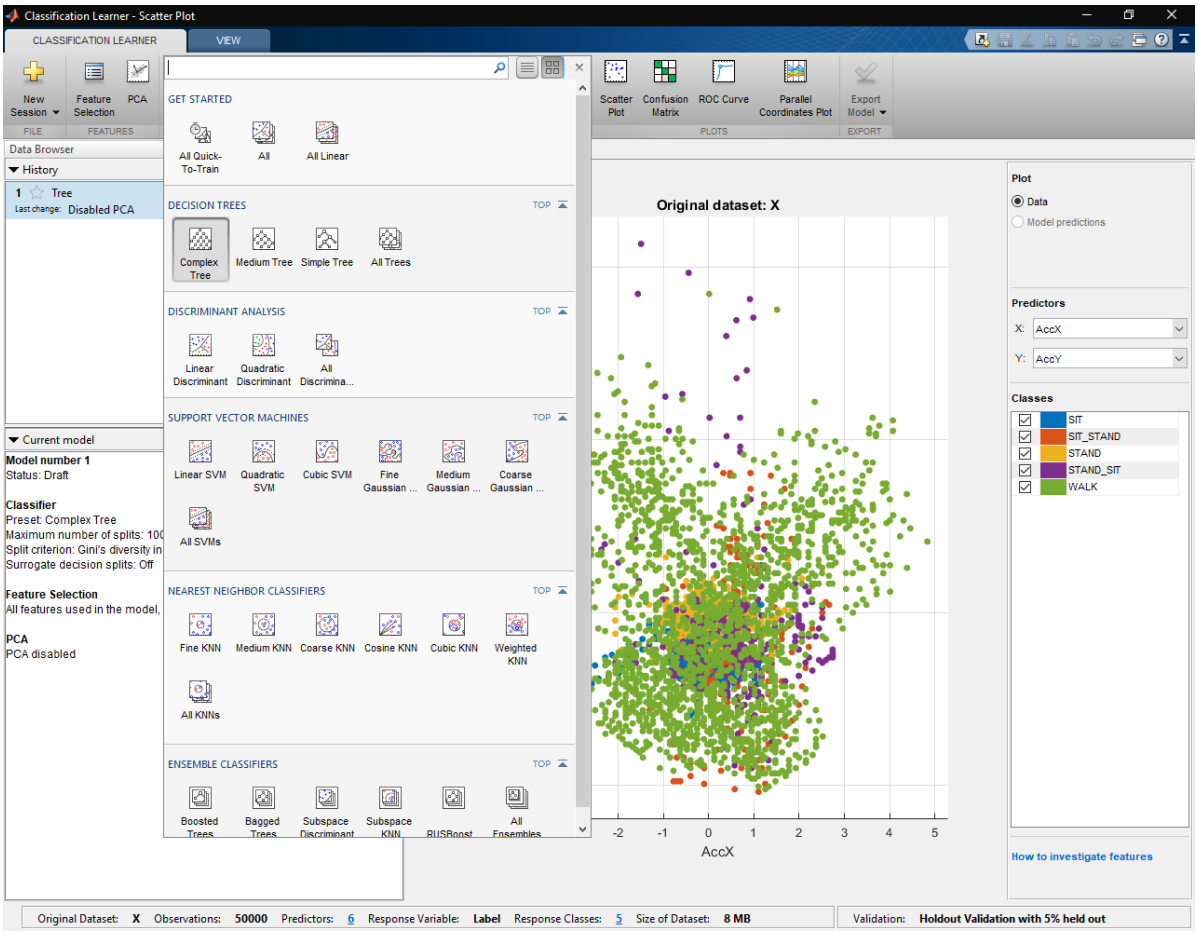


Abbildung 7.16.: Auswahl der Modelle

Nach dem Training wird die Genauigkeit des Jeweiligen Modelles in prozentualer Darstellung ausgegeben. Nützlich sind auch die weiteren Informationen zum Beispiel über die Laufzeit welche unten links in der Box aufgelistet sind. Zur weiteren Analyse des Modelles bietet die App vier Grafiken. Direkt angezeigt wird der Scatter Plot (siehe

Abbildung 7.17).

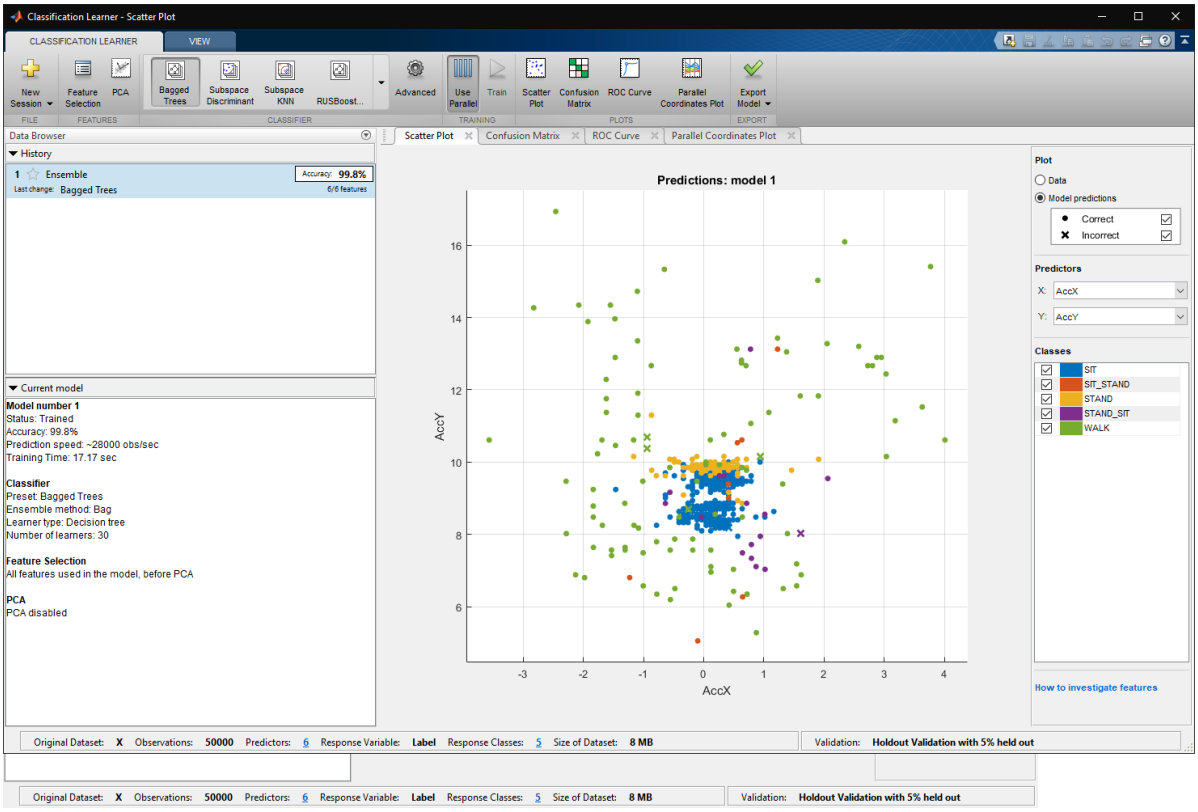


Abbildung 7.17.: Scatter Plot

Bei dem Scatter Plot werden zu den jeweiligen Prädiktoren die Prognosen dargestellt. Ein Punkt zeigt dabei einen richtig klassifizierten Eintrag an. Ein Kreuz zeigt wiederum die falsch klassifizierten Ergebnisse. Dadurch kann eine Analyse beispielsweise über die Erkrankungsrate von Ausreißern getätigt werden.

Eine andere Analyse bietet die Darstellung der ROC (Receiver Operating Characteristic)-Kurve wie in Abbildung 7.18 zu erkennen. Hierbei wird die True-positiv-rate der False-positv-rate gegenübergestellt. Bei einer 100 Prozentigen Erkennung liegt der Punkt dabei auf 0, 1.

Zur Darstellung hochdimensionaler Strukturen bieten sich die parallelen Koordinaten an. Auch diese können von Matlab visualisiert werden (siehe Abbildung 7.19). Die senk-

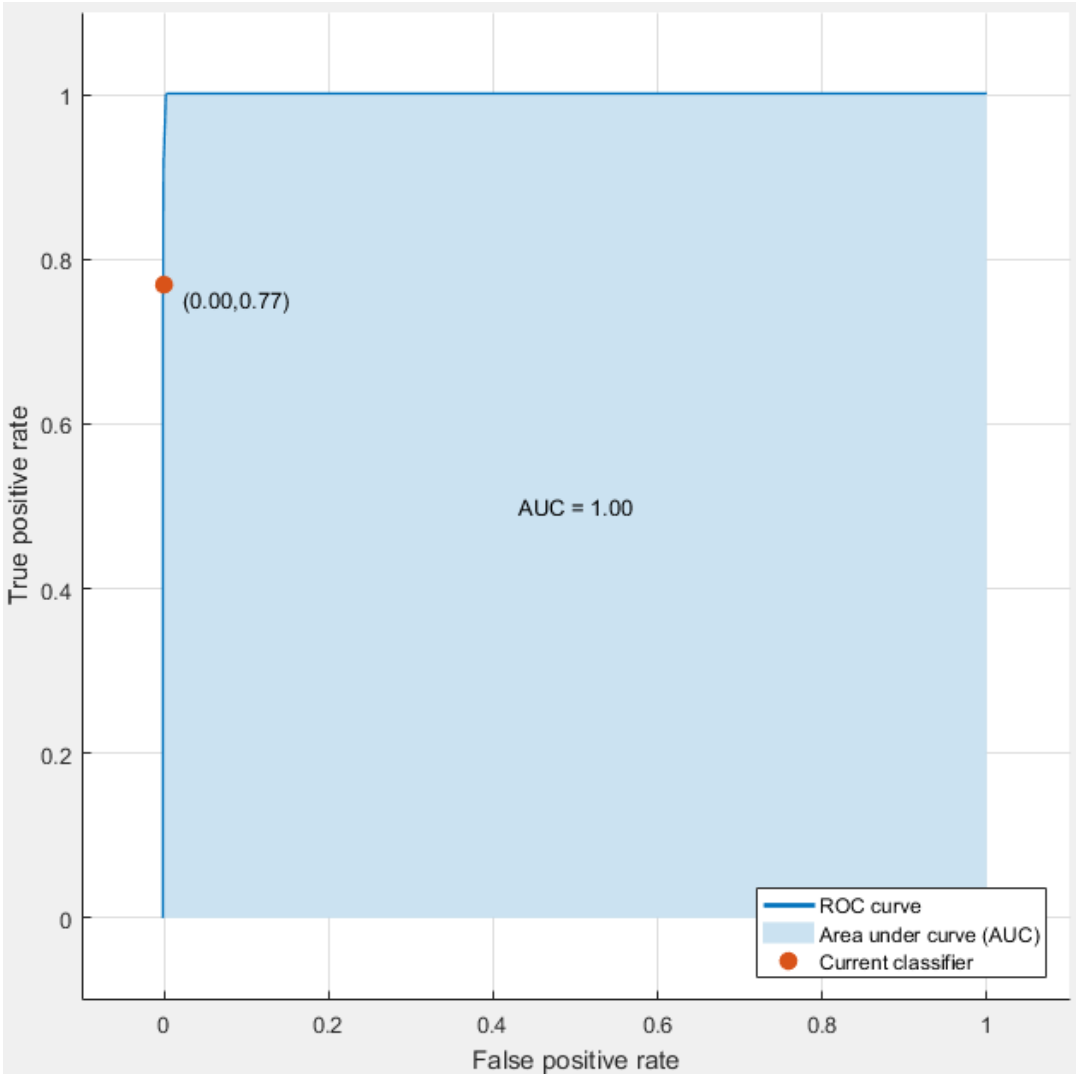


Abbildung 7.18.: ROC Kurve

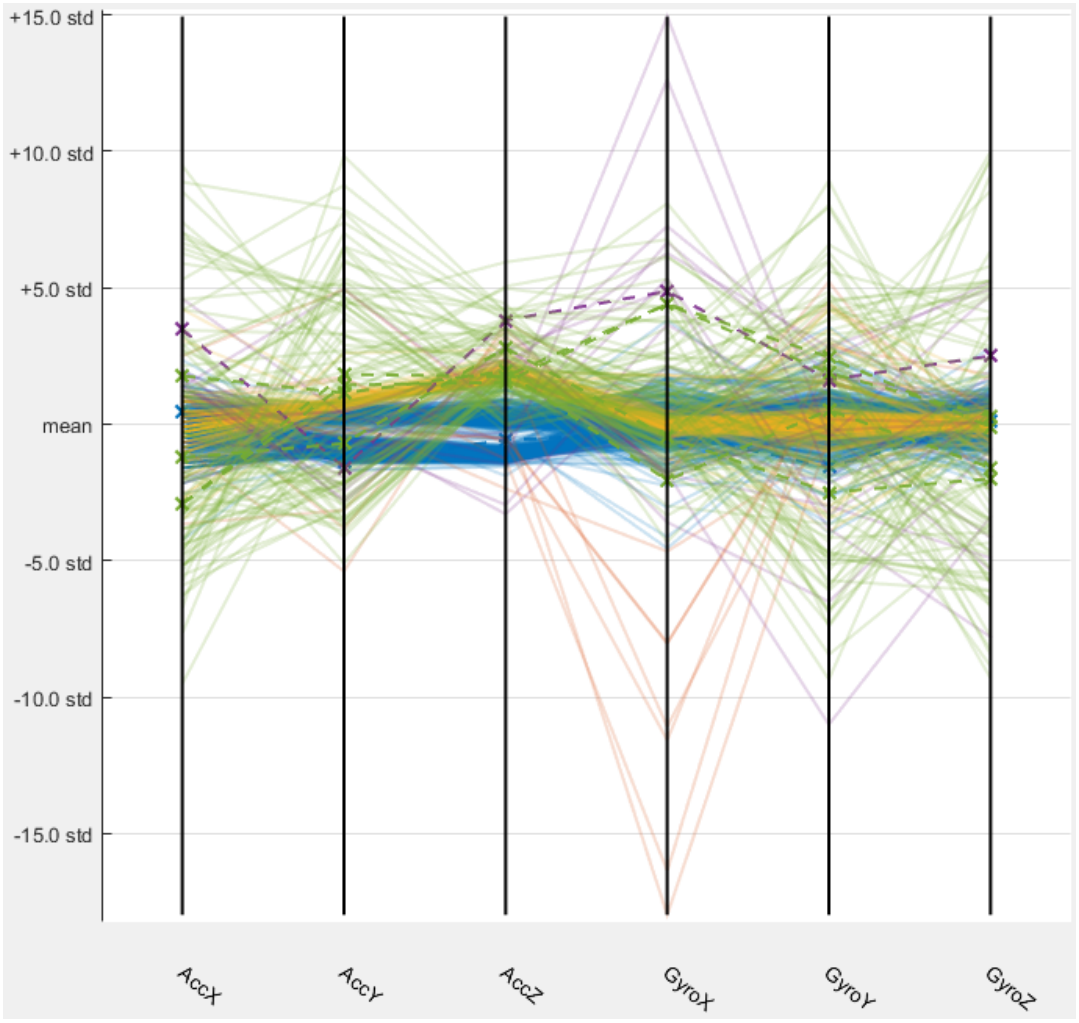


Abbildung 7.19.: Coordinates Plot

rechten, parallelen Linien stellen dabei jeweils die einzelnen Achsen dar.

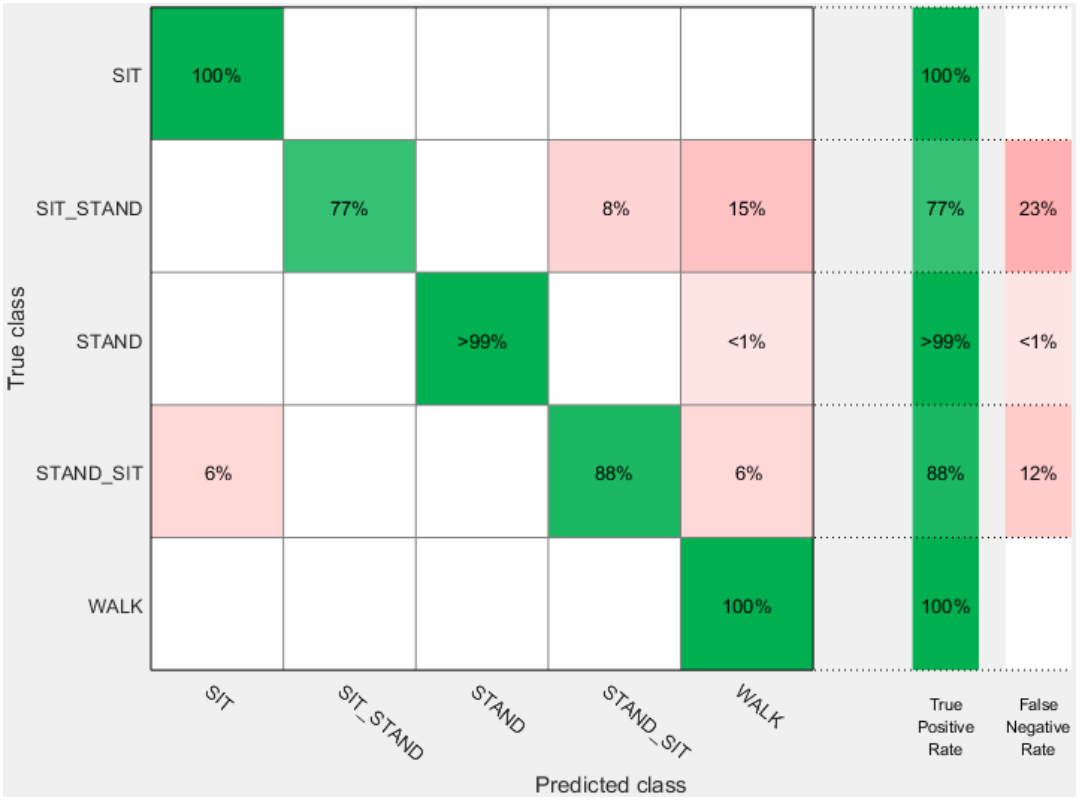


Abbildung 7.20.: Konfusions-Matrix

Schlussendlich ist aus der App heraus auch die Möglichkeit gegeben, eine Konfusionsmatrix zu erstellen. Hierbei werden beispielsweise die true class Daten der predicted class Daten gegenüber gestellt (siehe Abbildung 7.20). In dem Beispiel wurde SIT zu 100 Prozent richtig erkannt. Die Transition SIT_STAND hingegen wurde nur in 77 Prozent korrekt erkannt. In acht Prozent der Fälle wurde falscher weise ein SIT_STAND als STAND_SIT gelabelt, in 15 Prozent der Fälle wurde es als WALK klassifiziert. Diese Darstellungsform bietet eine recht genaue Aufschlüsselung, bei welchen Labeln eine verstärkte Verwechslungsgefahr besteht wie die Erkennungsrate der einzelnen Merkmale generell ausfällt. Aus diesen Gründen wurde entschieden, wenn möglich eine Analyse der unterschiedlichen Modelle mit Hilfe der Konfusionsmatrix durchzuführen.

Nach dem Training können die einzelnen Modelle über Export Modell exportiert werden.

Dabei ist es auch möglich den genutzten Code für das antrainieren des Klassifikators generieren zu lassen. Beides wird zunächst in den Workspace geladen. Erst mit dem Befehl save Filename.mat bzw. save Filename.m für den Code werden die Daten in derzeit ausgewählten Verzeichnis gespeichert. Das Speichern von größeren Modellen führte zu Problemen, so dass diese nicht gespeichert wurden. Dieses Problem konnte mit der zusätzlichen Übergabe der Dateiformat-Versionsnummer (save('Filename.mat','-v7.3');) behoben werden, wodurch auch das Speichern größerer Modelle problemlos gelang.

Da das lokale Antrainieren oft, gegeben durch geringe Ressourcen, sehr viel Zeit beansprucht sollen die Berechnungen bevorzugt auf dem HPC der Universität durchgeführt werden. (Vgl. Kapitel 8.1). Hierfür musste Matlab zunächst korrekt konfiguriert werden. Dafür wurde eine SLURM-Integrationsdatei heruntergeladen und im Verzeichnis MATLABROOTDIR/toolbox/local/ entpackt. Anschließend müssen benutzerspezifische Einstellungen vorgenommen werden. Dafür wurde ein neues Cluster Profil über den Cluster-Manager hinzugefügt. In diesem Profil wurden daraufhin verschiedene Parameter, wie zum Beispiel die maximale Anzahl an reservierbaren Nodes, eingestellt. Eine genaue Beschreibung hierzu befindet sich im HPC-Wiki unter:

`https://wiki.hpcuser.uni-oldenburg.de/index.php?title=Configuration_MDCS_2016`

Um einen neuen Job auf dem HPC durchzuführen wird zunächst mit dem Befehl sched = parcluster('CARL'); das Clusterprofil ausgewählt. Anschließend können mit dem set-Befehl Anforderungen, wie zum Beispiel die maximale Laufzeit des Jobs, bestimmt werden:

```
set(sched, 'CommunicatingSubmitFcn', cat(2,
sched.CommunicatingSubmitFcn, {'runtime','288:0:0'}));
```

Mit dem nächsten Befehl werden alle dort aufgelisteten Dateien auf den Cluster geladen und der Job wird daraufhin gequeued bzw. gestartet. Übergeben wird auch die Anzahl

an Nodes, die das System zur Berechnung bereit stellen soll:

```
job_20_30 = batch(sched, 'jobscript_20_30',
'Pool', 15, 'AttachedFiles', {'trainClassifier_1b.m',
'trainClassifier_2.m','trainClassifier_3.m'})
```

7.3.3. Exportieren zu MAMKS

Mit Hilfe der erstellten Modelle ist es möglich, die Datensätze zu klassifizieren. Damit die Ergebnisse auch in MAMKS geladen und analysiert werden können wurde für Matlab eine Export-Funktion erstellt. Die export_label_to_mamks-Funktion erstellt dabei im Zusammenspiel mit der Funktion run_calculate_label und calculate_rotations XML-Dateien, welche die für MAMKS notwendige Struktur besitzen. Diese können in die SHID-Ordner kopiert und anschließend eingelesen werden.

7.4. Python Module

In diesem Kapitel werden die Bibliotheken kurz vorgestellt, die für die Umsetzung des konzepierten Klassendiagramms 4.3 benutzt worden sind. Die nachfolgenden Bibliotheken besitzen außerdem Abhängigkeiten untereinander. Um vor der Verwendung die Bibliotheken zu installieren wurde ein entsprechendes Setupscript angelegt. Durch die Ausführung werden alle Abhängigkeiten, die zur Ausführung nötig sind, bereitgestellt. Nachfolgend eine kurze Beschreibung der verwendeten Bibliotheken und Frameworks.

Pandas

Pandas ist eine open-source-Library um komplexe Datenstrukturen möglichst einfach anzulegen. Des Weiteren dient sie zum Anlaysieren von Daten und bietet dabei eine

hohe Performanz.

numpy

Numpy ist eine Bibliothek zur einfachen Verarbeitung von Vektoren oder Matrizen. Mit ihr wird außerdem der Umgang mit großen mehrdimensionalen Arrays vereinfacht. Des Weiteren bietet sie Funktionen zum Verarbeiten von linearer Algebra.

SciPy

SciPy (scientific computing) stellt eine Großzahl an Methoden zum wissenschaftlichen Rechnen zur Verfügung. Hauptkomponenten sind unter anderem effiziente Tools zum Datenmanagement so wie high-performance computing.

matplotlib

Eine sehr umfangreiche Bibliothek zur graphischen Darstellung von Daten. Sie ermöglicht ein besonders intuitives Plotten von Graphen und Diagrammen. Matplotlib ermöglicht eine Darstellung im zwei- und dreidimensionalen Format.

scikit-learn

Bei scikit-learn handelt es sich um eine ausgereifte Machine-Learning-Library. Mit ihr ist es möglich sehr effizient Machine-Learning-Modelle anzulernen und zu testen. Des Weiteren bietet scikit-learn unzählige Möglichkeiten zum Visualisieren, Testen und Validieren. Unter anderem stehen Methoden wie die Klassifikation, Regression, Clustering, Dimensionsreduktion und Vorverarbeitung zur Verfügung.

Tensorflow

TensorFlow ist ein Open-Source Framework, das im Google Brain Project entwickelt worden ist. Das Hauptziel der Entwickler ist es, eine hoch skalierbare Lösung zur Ausführung von Berechnungen auf heterogenen Systemen zu entwickeln. Ursprünglich sollte Tensorflow auf Google-Produkte, wie Google- Search oder Google-Maps beschränkt sein. Im Jahr 2015 beschloßen die Entwickler das Framework unter der Apache 2.0 Lizenz zu veröffentlichen. Das quelloffene System ist unter www.tensorflow.org verfügbar. Die Berechnungen in Tensorflow werden, basiert auf einem Datenfluss-ähnlichen Modell auf verschiedene Systeme verteilt [5]. So kann eine Berechnung definiert werden und im Hintergrund werden einzelne Teilschritte auf berechnende Teile des/der Systems/Systeme, wie CPUs oder GPUs verteilt.
Jede Berechnung wird als ein gerichteter Graph aufgefasst. Dabei repräsentiert ein Graph Datenflüsse von einem Knoten zum nächsten.

TFLearn

TFLearn ist eine auf Tensorflow aufbauende deep learning library. Es handelt sich hierbei um eine High-level-API die den Umgang mit Tensorflow vereinfacht und Befehle zusammenfasst. Sie ist vor allem dazu ausgelegt neuronale Netze zu implementieren und zu testen.

7.4.1. Implementierung der Datenstruktur

Nachfolgend werden die Implementierungsdetails in Python erläutert. Sie umfassen das Importieren, Verarbeiten und Exportieren der konformen Daten.

- Um mit den Bewegungsdaten von MAMKS zu arbeiten, müssen diese vorerst importiert werden. Aus diesem Grunde wurde in dem Script ‘mamks_utils’

eine Importfunktion implementiert. Sie nimmt den Workspace-Pfad entgegen und importiert anschließend die .bin-Dateien. Derzeit werden ausschließlich die Beschleunigungs- und Gyroskopdaten importiert. Eine Erweiterung ist im Code ohne Probleme möglich. Die .bin-Dateien werden mit Hilfe von Float32 in ihre ursprüngliche numerische Form gebracht, damit mit ihnen komfortabel weitergearbeitet werden kann. Nach dem Import und der Konvertierung werden die Daten in einem Array abgespeichert.

- Zur Klassifikation der eingelesenen Daten bestand die Voraussetzung unterschiedliche Modelle anwenden zu können. Hierzu wurde das Script 'classification_models' erstellt. Es implementiert eine abstrakte Oberklasse und nimmt den entsprechenden Modelpfad zum Importieren entgegen. Des Weiteren gibt es die predict-Funktion vor, die anschließend von den Modellklassifikatoren implementiert werden muss. Bisher wurden die Klassen für die Ausführung von CNN-Modellen, scikit-learn-Modellen und deren Nachbearbeitung implementiert. Es können je nach Anforderung weitere Modelle in diesem Script ergänzt werden.

- Nachdem die eingelesenen Daten fertig klassifiziert und nachbearbeitet worden sind, müssen anschließend noch die zugehörigen Label erzeugt werden, damit sie in MAMKS angezeigt werden können. Hierzu wurde in dem Script 'mamks_utils' die Klasse 'MamksLabelGenerator' erstellt. Um die Label zu exportieren benötigt sie beim Aufruf die Parameter des derzeitigen Dateipfads (damit der Export im richtigen Probandenordner durchgeführt wird), die klassifizierten Label in Array-Form (numerisch) und eine Label-Map, damit die Namenszuweisung erfolgen kann. Anschließend werden für jeden Datensatz die Label in XML-Form generiert und im dafür von MAMKS vorgesehenen Label-Ordner abgelegt.

7.4.2. Mamks-Schnittstelle

Um eine Schnittstelle zu MAMKS zu gewährleisten wurden für die zuvor beschriebenen Klassen entsprechende Scripte erstellt welche als Einstiegspunkt des CLI Aufrufs dienen. Hierbei handelt es sich um die Scripte 'classify_mamks_data_scikit_models' und 'classify_mamks_data_with_cnn'. Die beiden Scripte können per CLI aufgerufen werden und erwarten folgende Parameter:

1. **ws_path** - Der Workspace Pfad, in dem die Probandendaten im MAMKS Format liegen
2. **model_path** - Der Pfad zum Modell, welches die Klassifikation vornehmen soll
3. **window_size** - Die Fensterbreite die bei der Nachbearbeitung angewendet werden soll
4. **step_size** - Die Schrittweite die bei der Nachbearbeitung angewendet werden soll
5. **min_score** - Der Wert, wie sicher sich das Modell beim Labeln sein soll, bevor das Label erstellt wird. Falls das Modell beim Klassifizieren nicht über diesen Wert kommt, wird ein 'UNKNOWN'-Label erzeugt. Dieser Wert wird nur von CNN unterstützt. Die anderen Modelle ignorieren diesen Wert, auch wenn er angegeben wird.

Ein Beispielhafter Aufruf könnte also wie folgt aussehen:

```
classify_mamks_data_scikit_models.py ws_path=C:\Desktop\MAMKS
model_path=C:\Desktop\Modelle\wknn.sav
window_size=100 step_size=100 min_score=0.6
```

Eine detailierte Beschreibung der Aufrufsparameter und wie die Scriptausführung durchgeführt wird, ist in dem Handbuch für die Python-Scripte zu finden. Des Weiteren wird dort erläutert, wie die Durchführung mit Hilfe der MAMKS-GUI gestartet werden kann.

Falls weitere Modelle über die MAMKS-Schnittstelle aufgerufen werden möchten, können analog zu den bisher vorhandenen Scripten weitere hinzugefügt werden. Hierbei sollte allerdings auf die Namenskonvention geachtet werden (‘classify_mamks_data..’), damit sie von MAMKS erkannt werden.

8. Optimierungen

Bei der Optimierung hat sich die Projektgruppe auf fünf Algorithmen konzentriert. Das Ziel sollte es sein, jedes Modell bezogen auf seine jeweiligen Parameter auf Verbesserungsmöglichkeiten zu überprüfen. Diese sind im folgenden näher beschrieben.

8.1. HPC

Da viele der genutzten Algorithmen sehr rechenintensiv sind, wurde versucht auf eine lokale Durchführung zu verzichten und dafür die Berechnungen auf dem HPC durchzuführen, welcher durch die Universität für diese Zwecke zur Verfügung gestellt wird.

Der HPC besteht aus den beiden Clustern CARL und EDDY. CARL besitzt eine theoretische Performanz von maximal 271 TFlop/s. Er besteht aus:

- 327 Nodes (9 mit einer GPU)
- 7.640 CPU Kerne
- 77 TB RAM
- 360 TB lokaler Speicher

EDDY besitzt eine theoretische Performanz von maximal 201 TFlop/s und beinhaltet:

- 244 Nodes (3 mit einer GPU)
- 5.856 CPU Kerne
- 21 TB RAM

Dabei ist die Ressourcenverteilung auf den einzelnen Clustern nicht gleich verteilt. CARL besitzt beispielsweise fünf verschiedene Gruppen. Diese unterscheiden sich unter anderem in ihrer Arbeitsspeicheranbindung. Die Gruppe High-Memory besitzt beispielsweise pro Node 512 GB RAM und besteht aus insgesamt 30 Nodes. Die Gruppe Low-Memory besitzt pro Node 128 GB RAM und besitzt 128 der insgesamt 327 Nodes. Durch die unterschiedlichen Ressourcen in den Gruppen sind diese unterschiedlich relevant. So ist die Gruppe welche die neun NVIDIA Tesla P100 16GB PCI-Edition besitzt besonders für Algorithmen wie CNN interessant. Die High-Memory-Gruppe ist wiederum für speicherhungrige Algorithmen wie Bagging-Trees interessant.

Das Job-Management auf dem HPC übernimmt der SLURM (Simple Linux Utility for Resource Management) Workload Manager. Dieser ist Opensource und besitzt eine weiter Verbreitung unter allen Supercomputern. Mit Hilfe von SLURM können somit beispielsweise von Matlab Jobs hochgeladen und ausgeführt bzw. eingereiht werden. Auch die Verwaltung des einzelnen Jobs ist möglich wie etwa der Abbruch oder ähnliches. Mit einem Job können auch die erforderlichen Ressourcen definiert werden. Diese werden dann, insofern vorhanden, automatisch zugewiesen.

Damit die Projektgruppe Zugriff auf den HPC bekommt mussten einige Schritte durchgeführt werden. Zunächst musste jeder Benutzer einen Account beantragen. Zur Nutzung muss dann, je nach Standort, eine VPN-Verbindung zur Universität aufgebaut werden. Für diverse Themen steht ein umfangreiches Wiki zur Verfügung, in dem unter anderem die Einrichtung für einzelne Programme genauer beschrieben wird. Dennoch gab es in der Projektgruppe Fragen, die in einem Interview mit dem HPC-Beauftragtem versucht wurden zu klären. Das Interview erbrachte das folgende Ergebnis:

- Ist die Matlab Version 2017b in Verbindung mit dem Cluster nutzbar?
  - Die Version 2017b ist in Planung. Anleitung zur Nutzung erfolgt im Wiki.
- Ist die Nutzung vom „Parallel Pool“ möglich? (Symbol unten Links in Matlab)

  - Nein. Wird zunächst auch nicht folgen.

- Wie viel RAM kann pro Node für die Berechnung genutzt werden?

  - Ist abhängig von dem genutzten Cluster (z.B. Carl). Verfügbare Ressourcen stehen im Wiki. Die Nodes erstellen keinen Swap, wodurch mehr HDD-Speicher keinen Mehrwert bringt.

- Können mehr als 36 Nodes genutzt werden?

  - Nein, da maximal 72 Lizenzen zur Verfügung stehen, welche pro Node angesetzt sind.

- Dürfen Berechnungen länger als 21 Tage dauern?

  - Ja, allerdings nur nach Absprache.

- Wann wird der Cluster viel genutzt?

  - Primär im Sommer, wenig gegen Winter/ Weihnachten.

- Wie viel Speicher steht im „/home“-Verzeichnis zur Verfügung?

  - 1TB. Im „work“ und „data“-Verzeichnis 10TB - zudem mit einer deutlich schnelleren Anbindung.

Als Resultat aus dem Interview wurden entschieden, Matlab in der Version 2016b einzusetzen, damit keine Probleme in Verbindung mit dem HPC entstehen. Auch war geplant, größerer Berechnungen über Winter/ Weihnachten durchzuführen.

Um die Ergebnisse der Berechnungen einheitlich, und an einem zentralen Ort speichern zu können wurde im Laufe des Projektes entschieden, lediglich einen Benutzer zu verwenden. Daraufhin wurde dieser Account für die Projektgruppe erstellt und eingesetzt.

8.2. Convolutional Neural Network

Ein künstliches neuronales Netz ist in der Lage beliebige Annäherungen zu approximieren.Wie in der Seminararbeit zu faltenden neuronalen Netzen bereits beschrieben sind diese eine Erweiterung klassischer vorwärtsgerichteter neuronaler Netze um die Möglichkeit, Merkmale automatisiert zu generieren. Die Motivation dahinter ist die negative Erfahrung der Projektgruppe hinsichtlich der manuelle Auswahl von Merkmalen.

8.2.1. Zu optimierende Parameter

Ein faltendes neuronales Netz kann prinzipiell auf viele Weisen angepasst werden. Zum einem kann die Struktur des Netzes und die Anzahl der versteckten Schichten angepasst werden. Durch die Steigerung der Schichtenanzahl wächst die Modellvarianz und damit auch die Fähigkeit des Netzes komplexere Daten zu klassifizieren. Auf der anderen Seite wächst auch die Gefahr des Overfittings.
Eine zweite Möglichkeit das Modell zu optimieren ist die Steigerung der Anzahl von Feature-Maps pro versteckte Schicht. Dies kann ebenfalls zur Steigerung der Modellvarianz, mit ähnlichen Auswirkungen und Folgen führen.
Eine dritte Möglichkeit besteht darin, komplexere Aktivierungsschichten einzusetzen. Beispielsweise kann die Tanh-Funktion im Gegensatz zur linearen ReLu-Funktion auch nicht-lineare Merkmalsräume separieren. Der Einsatz von komplexeren Funktionen führt jedoch zu stark ansteigenden Ausführungszeiten, da sich die Anzahl der Multiplikationsoperationen pro Schicht, im Vergleich zu linearen Funktionen, oder Funktionen aus der ReLu-Familie verhundertfachen.
Wie in der Seminarausarbeitung zu Faltenden Neuronalen Netzen vorgeschlagen, besteht unsere Struktur des Netzes aus einem Inception-Modul. Die Originalstruktur enthält maximal 64 Feature-Maps pro Schicht. Mehrere Trainingsversuche haben gezeigt, dass der Einsatz komplexerer Aktivierungsschichten eine deutlich höhere Ausführungsdauer, sowohl im Training als auch in der Klassifizierung zur Folge hatte. Das

Training konnte aufgrund sehr langer Dauer nicht abgeschlossen werden. Aus diesem Grund wurde entschieden lediglich die Anzahl der Feature-Maps pro Schicht zu erhöhen.

Das Training des Modells erfolgte auf zuvor vorverarbeiteten Daten. Der Vorverarbeitungsschritt bestand aus der Segmentierung der Daten auf ein Fenster mit der Länge von 150 Signalen des Accelerometers und des Gyroskops jeweils in den x,y und z-Achsen. Die Länge der Zeitdauer wurde ebenfalls durch mehrfache Versuche mit unterschiedlichen Zeitfenstern unter Beurteilung des Trainingsverlaufes anhand des Verlaufes der Loss-Funktion bestimmt. Die Daten wurden mit einer Schrittweite von 25 Signalen segmentiert. Die Annotation der Segmente erfolgte nach meist vorkommenden Auftreten eines Labels. Nach dem Segmentierungsschritt wurden stark unterrepräsentierten Segmente mit dem Upsampling-Verfahren auf 25% der maximal vorkommenden Klasse aufgefüllt.

Der Trainingsverlauf erfolgte mit Techniken zur Verhinderung des Overfittings. Die erste Technik ist Drop-Out. Dabei wurden in der Klassifizierungsschicht Neuronen mit einer Wahrscheinlichkeit von 20% mit 0 Werten belegt. Diese Maßnahme ist eine strukturelle Regulierung des Netzes, um zu verhindern dass Teile des Netzes die Datenbeispiele auswendig lernen. Die zweite Maßnahme ist die L2-Regularisierung. In jeder faltende Schicht wurden die Gewichtsvektoren bestraft, wenn ihre Komplexität zu hoch war. Das heißt wenn die Werte der Gewichtsvektoren stark variieren. Der Regularisierungswert R(W) wurde mit einem Gewichtsfaktor von 0.01 belegt.

8.2.2. Optimierung in MATLAB

MATLAB bietet mit seiner Neural-Network Pattern Recognition Toolbox eine einfache Möglichkeit, ein Feed Forward Netz mit eigenen Daten anzutrainieren. Diese Möglichkeit wurde zu Beginn des Projekts mit Assessmentsdaten der PG-MAMKS untersucht. Es

wurden insgesamt 18 Aktivitäten ausgewählt und nicht bekannte Bereiche aus dem Trainingsdatensatz ausgeschnitten.

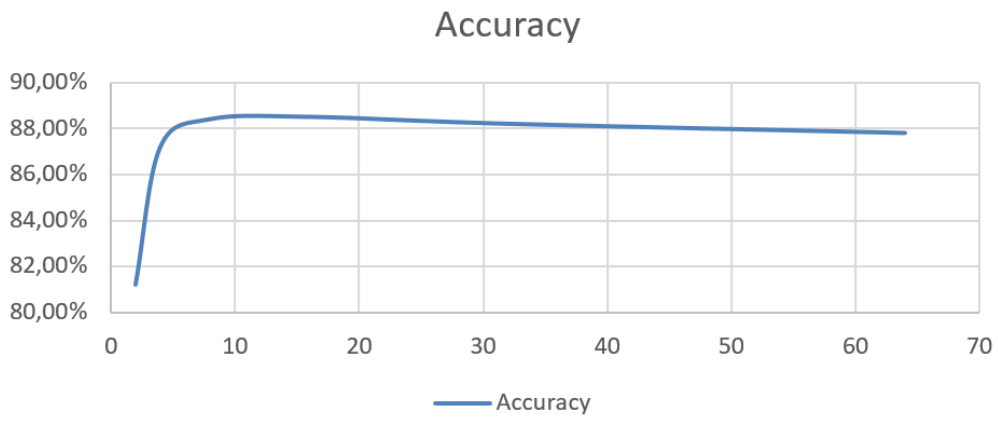


Abbildung 8.1.: Trainingsverlauf mit steigenden Anzahl an Neuronen

Die Übersicht in 8.1 zeigt Ergebnisse der Parameteränderung. Zunächst wurde nur die Anzahl der Neuronen in der versteckten Schicht des Netzes betrachtet. In der X-Achse ist die Anzahl der Neuronen in der versteckten Schicht angezeigt. Die Y-Achse zeigt die Accuracy in Prozent an. Mit der Zunahme der Neuronen verschlechtert sich das Ergebnis zunehmend. Dies liegt daran, dass die Anzahl der Iterationen pro Training auf 1000 Iterationen gesetzt war.

Die Darstellungen des Cross-Entropy Verlaufes in 8.2 zeigen, dass der Trainingsalgorithmus (in diesem Fall das Verfahren der konjugierten Gradienten) noch nicht das globale Optimum erreicht hat. Dargestellt ist darin der Verlauf des Trainings mit 64 Neuronen. Wie in der Seminararbeit zu faltenden neuronalen Netzen bereits erläutert, ist das Training eines Netzes, im Grunde ein Optimierungsverfahren, wobei je nach Ziel, die gemachten Fehler, aber auch die Bestrafungsterme zur Berechnung der Zielfunktion (od. auch Verlustfunktion) herangezogen werden. In MATLAB gibt es keine direkte Möglichkeit, im Training Regularisierung oder Drop-Out zu nutzen. Der Grafik in 8.2 ist zu entnehmen, dass die Anzahl der Iterationen der limitierende Faktor war. Im nächsten

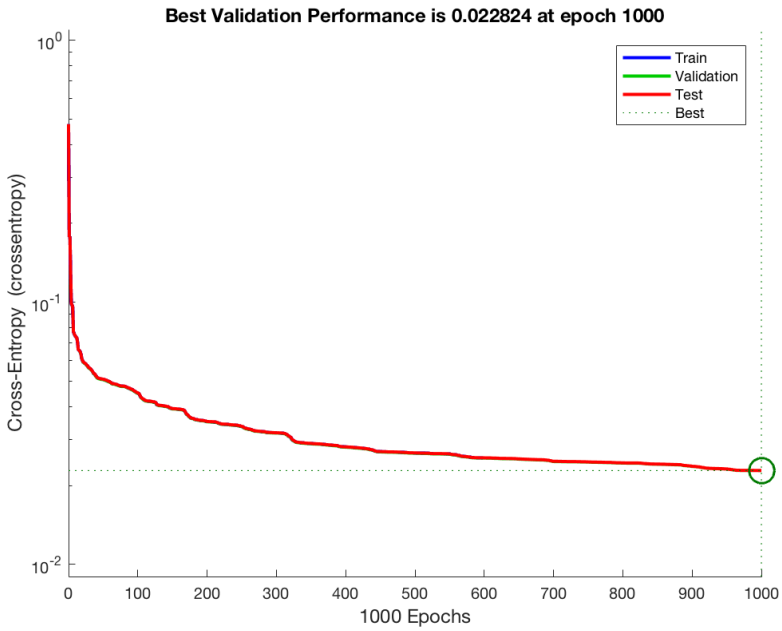


Abbildung 8.2.: Trainingsverlauf mit steigenden Anzahl an Neuronen

Schritt, kann die Anzahl der Iterationen variierend mit einem Netz, das 8 versteckte Neuronen hat (bestes Ergebnis), betrachtet werden.

Die Abbildung 8.3 zeigt die Konfusionsmatrix nach dem Training eines Netzes mit 64 Neuronen in der versteckten Schicht. Die Y-Achse stellt dabei die Ground-Truth Aussagen zu den Aktivitäten in den Daten dar. Die X-Achse zeigt die, vom Neuronalen Netz gemachten Voraussagen an. Ein weiteres Problem stellt die Klassen-Imbalance in den Trainingsdaten dar. Es ist zu sehen, dass eine starke Ungleichverteilung der Klassen in den Daten vorhanden ist. Die Aktivitäten 1 und 3 sind mit 19 % beziehungsweise 39 % die am meisten vorkommenden Aktivitäten. Die besten Klassifikationsergebnisse wurden in den Aktivitäten mit den meisten Samples erzielt.
Im nächsten Schritt sollte das Netz auf Optimierungsmöglichkeiten untersucht werden. Prinzipiell wurden die Erweiterungen des Merkmalsraumes um weitere Merkmale diskutiert. Eine weitere Optimierungsmöglichkeit wäre eine Erweiterung des Netzes um

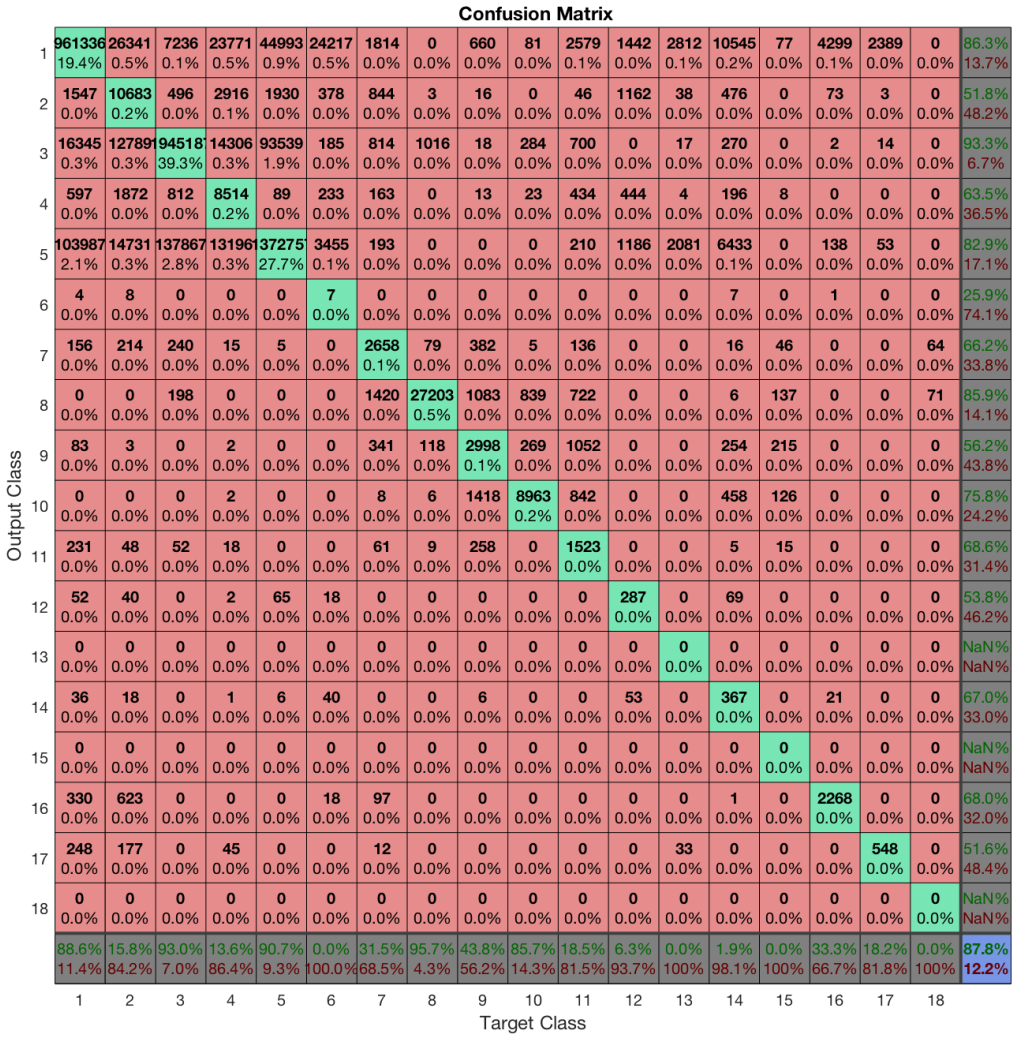


Abbildung 8.3.: Trainingsverlauf mit steigenden Anzahl an Neuronen

fensterbasierte Klassifizierung, da sie im ersten Ansatz nur punktbasiert erfolgte. Punktbasiert heißt hier, dass jedes Signal in den Daten klassifiziert wurde. Sowohl die Merkmalsextraktion, als auch die fensterbasierte Klassifizierung würden, wie in der Seminararbeit zu faltenden neuronalen Netzen beschrieben, zu stark ansteigenden Ausführungszeiten, sowie zu nicht zu bewältigenden Trainingszeiten führen, da die Anzahl der vollvernetzten Neuronen mit jedem neuen Signal, beziehungsweise mit steigenden Fensterbreite proportional steigen würde. Aus diesem Grund wurde entschieden, weitere Optimierungen mit faltenden neuronalen Netzen fortzuführen. Da es keine offiziellen Implementierung von CNNs in MATLAB (R217a) gibt, erfolgt die weitere Optimierung in Python.

8.2.3. Optimierung in Python

Mit dem Umstieg der Optimierung von klassischen, vorwärts gerichteten neuronalen Netzen zu faltenden neuronalen Netzen, wird die Optimierung hinsichtlich der Merkmalsauswahl strukturbedingt automatisiert. Für die Erzeugung des Merkmalsraumes ist jetzt lediglich die Struktur des Netzes, sowie die Anzahl der Feature-Maps pro Schicht relevant.
Wie bereits erwähnt, wird das Netz, das in der Seminararbeit zu faltenden neuronalen Netzen entwickelt wurde eingesetzt. Dieses Netz generiert Merkmalskarten auf unterschiedlichen Skalierungsebenen. Die Generierung erfolgt mit Hilfe des sogenannten Inception-Moduls, das ebenfalls bereits in der Seminararbeit behandelt wurde. Anschließend folgt eine voll vernetzte Klassifikationsschicht, die mit der SoftMax-Aktivierungsschicht verknüpft ist. Diese Struktur wird im Laufe der Optimierung beibehalten.

Da die steigende Komplexität bei der Auswahl von Aktivierungsfunktionen unverhältnismäßig ist, wird sich die Optimierung auf die Steigerung der Feature-Maps fokussieren. Es werden daher zwei Konfigurationen untersucht. Das erste Netz (CNN4Har2) be-

inhaltet in seinen versteckten Schichten 16 Feature-Maps in den Komponenten des Inception-Moduls, sowie 64 Feature-Maps in der nachfolgenden faltenden-Schicht. Die anschließende voll vernetzte Schicht besteht aus 512 Neuronen. Die zweite Variante (CNN4Har3) beinhaltet in ihren versteckten Schichten 32 Feature-Maps in den Komponenten des Inception-Moduls, sowie 128 Feature-Maps in der nachfolgenden faltenden-Schicht. Die anschließende voll vernetzte Schicht besteht aus 1024 Neuronen.
Die Datenvorbereitung bestand aus insgesamt zwei Schritten. Das Einlesen der Daten und Segmentierung der Daten mit einer festen Fensterbreite mit einer Schrittweite, die kleiner oder gleich der Fensterbreite sein kann. Diese Vorgehensweise wurde breites in der Seminararbeit zu faltenden neuronalen Netzen diskutiert. Der fensterbasierte Segmentierungsansatz ist für faltende neuronale Netze besonderes gut geeignet, da diese die inhärente Eigenschaft besitzen, lokale Zusammenhänge als Merkmale zu extrahieren. Eine Schrittweite die kleiner als die Fensterbreite gewählt wurde, sorgt zusätzlich dafür, dass die Samples, durch Verschiebung (bzw. Translation) aus quasi unterschiedlichen Perspektiven präsentiert werden können. Diese Segmentierungsmethode könnte man daher auch als implizites Upsampling bezeichnen. Schließlich wurden die Fenster mit der am meisten vorkommenden Aktivität beschriftet. Waren beispielsweise 50 Signale von 150 als Gehen beschriftet, so wurde das gesamte Fenster als Gehen beschriftet. Dies hat den Vorteil, dass Ungenauigkeiten nach manuellen Beschriftungen in den Zwischenbreichen weniger negative Auswirkungen haben dürften. Die Granularität der Klassifikation entspricht dabei der Schrittweite, so dass der Klassifikationsalgorithmus eine Entscheidung für die Breite des Schrittes mittels Beobachtung einer größeren Fensterbreite trifft.
Durch Probeläufe und Beobachtung des Trainingsverlaufs mittels des Kurvenverlaufes der Loss-Funktion wurden einige Parameter vorab festgelegt. Um zu entscheiden, ob ein Parameter, wie Fensterbreite oder Schrittweite während der Segmentierung optimal gewählt sind, reicht es aus die ersten zwei bis drei Durchläufe zu beobachten. Verläuft die Kurve einer Loss-Funktion steiler und regelmäßiger nach unten als ihre Vorgänger-Probeläufe, so kann angenommen, dass die gewählte Parameterkonstellation optimaler

ist als die, der Vorgänger-Probeläufe. So wurde festgelegt, dass die Fensterbreite von 150 Signalen pro Fenster, sowie die Schrittweite von 75, die optimaleren Ergebnisse liefern, als ihre Vorgänger mit kleineren Fensterbreiten. Eine größere Fensterbreite wurde nicht betrachtet, da die Befürchtung bestand, dass während des Segmentierungsschrittes kurze Aktivitäten mit einer Länge von 1-1,5 Sekunden verschwinden würden.
Im Rahmen der Studiendurchführunng wurden Datensätze mit Alltagsaktivitäten älterer Probanden aufgezeichnet und beschriftet. Durch Trainingsergebnisse in Probeläufen wurde festgestellt, dass sehr viele Daten sehr stark unterrepräsentiert sind. Es besteht daher die Befürchtung, dass die Daten nicht ausreichen, um ein gutes Klassifikationsergebnis zu erreichen. Um diese Befürchtung zu untersuchen, wurden mehrere Datensatzgruppen, mit unterschiedlicher Anzahl an Beschriftungen angelernt und die Trainingsergebnisse gegenübergestellt.

Die Ergebnisse der Klassifikation haben diese Befürchtung widerlegt. Es hat sich gezeigt, dass unterschiedliche Kombinationen von Beschriftungen keine sichtbaren Unterschiede gebracht haben. Die Klassifikationsergebnisse lagen für die erste Netzvariante bei etwa 92 %. Aus diesem Grund wurde entschieden, die Optmierung auf zwei Labelgruppen zu reduzieren. Die verwendeten Gruppen sind Labelgruppe 1b (Datensatz 1) und die Labelgruppe full (Datensatz 2). Die enthaltenen Aktivitäten der jeweiligen Gruppen sind im Anhang zu finden.

Eine weitere Befürchtung ist, dass aufgrund schwieriger Bedingungen währen der Datenaufzeichnung und schwer zu definierender Grenzen zwischen den Aktivitäten, das Rauschen in den Daten sehr hoch sein könnte. Es wird angenommen, dass das Rauschen in den Daten bei etwa 10 % liegt. Dem entsprechend müssen die Klassifikationsergebnisse interpretiert werden.

Abbildung 8.4 zeigt eine Konfusionsmatrix, die nach dem Training der ersten Variante des Netzes, mit dem Datensatz 1 erzeugt wurde. Es fällt negativ auf, dass die

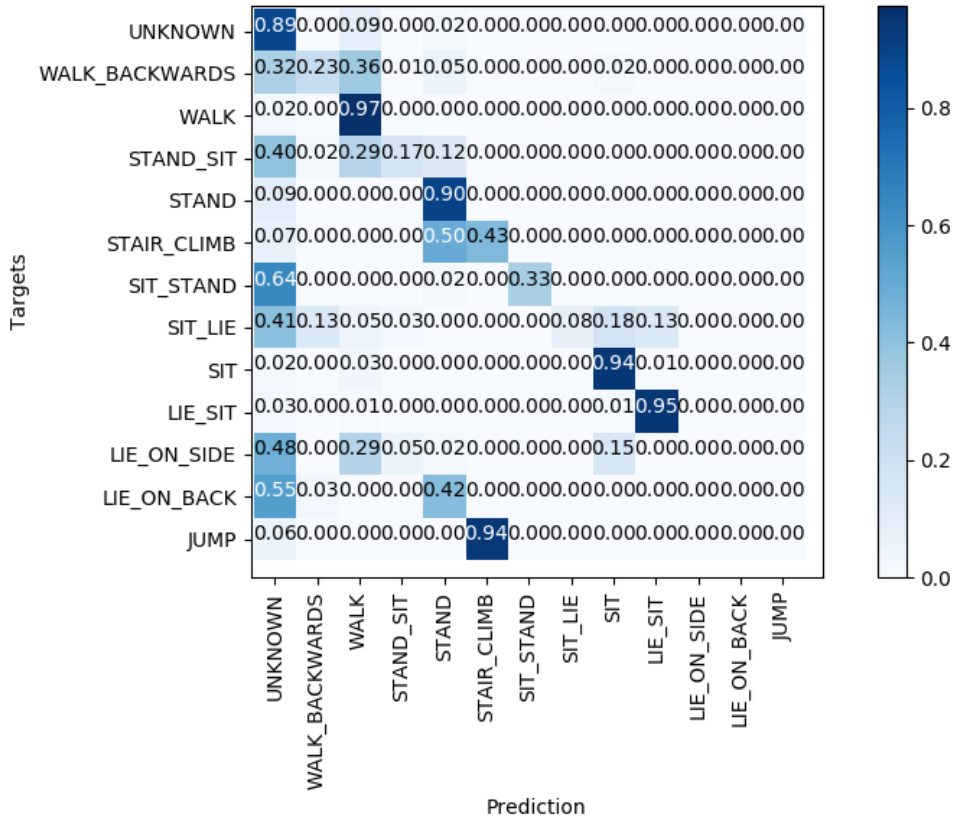


Abbildung 8.4.: CNN4Har2 Labelgruppe 1b.

falsch-positiv Rate für das Springen sehr hoch ist. Das Springen wurde in den meisten Fällen, vermutlich aufgrund der starken Signalstärke, als Treppensteigen erkannt. Viele Aktivitäten wurden als unbekannt markiert. Auch hier zeigte sich, dass die am meisten vorkommenden Aktivitäten am besten erkannt wurden. Hier besteht der Verdacht, dass das Modell nicht in der Lage ist der Komplexität der Daten gerecht zu werden. Wie in der Konfusionsmatrix in 8.4 ersichtlich ist, ist die Klassifikationsgenauigkeit auch von der Mindeskonfidenz des Modells abhängig. Mit Mindestkonfidenz ist hier die Prozentzahl gemeint, die notwendig ist, um eine Aktivität zu markieren. Ist die Konfidenz niedriger als nötig, wird das Sample als unbekannt markiert. Die Abbildung 8.5 zeigt die untersuchten Mindestkonfidenzen im Verhältnis zu den Evaluationsmetriken.

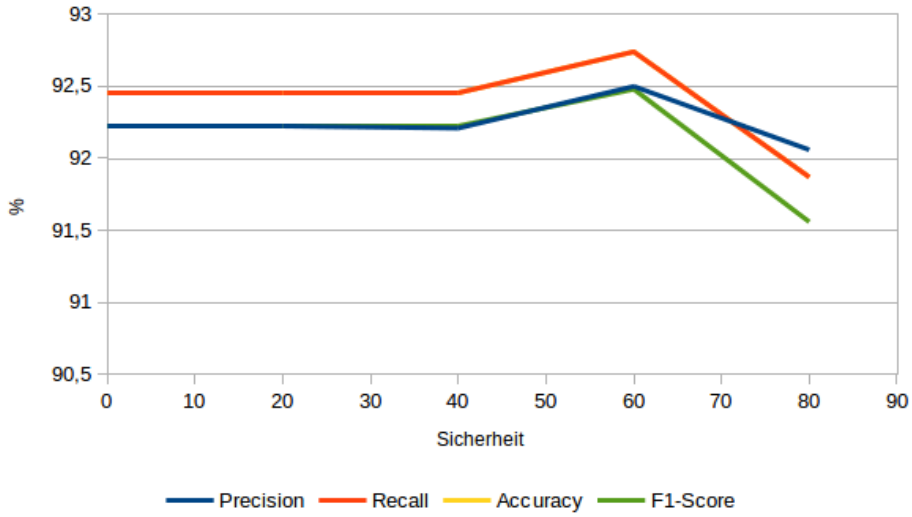


Abbildung 8.5.: CNN4Har2 Mindestkonfidenzen im Verhältnis zu Metriken, Precision, Recall, Accuracy und F1.

Dargestellt sind Angaben in Prozent für die festgelegte Mindestkonfidenz für das Klassifikationsergebnis auf der X-Achse. Auf der Y-Achse sind Prozentangaben für die jeweilige Metrik angegeben. Es hat sich gezeigt, dass die besten Ergebnisse mit einer Mindestkonfidenz von 60 % erzielt werden konnten. Die Ergebnisse für die Mindestkonfidenz von 100 wurden nicht aufgenommen, weil sie bei 0 liegen.

Die Erkenntnisse aus den Trainingsergebnissen der ersten Netzvariante werden für die Optimierung der zweiten Variante verwendet. Eine Erhöhung der Feature-Map-Anzahl hat ein weiteres Ziel, nämlich zu untersuchen, wie stark anpassungsfähig das Netz ist. Da nun davon ausgegangen werden muss, dass die Daten stark verrauscht sein müssen und etwa 10 Prozent der Daten falsch beschriftet sind, müsste ein Modell, um eine Klassifikationsgenauigkeit von über 93 % die Daten auswendig lernen. Um dem entgegen zu wirken wurden die Gegenmaßnahmen, wie L2-Regularisierung und Drop-Out in beiden Netzen verwendet.

Die Abbildung 8.6 zeigt das Ergebnis des Trainings mit dem optimierten Modell auf dem voll beschrifteten Trainingsdatensatz.

Insgesamt konnte eine Genauigkeit von 92,66% erreicht werden. Das optimierte Modell zeigt eine geringe falsch-positiv Rate im Vergleich zur ersten Variante. Das Ergebnis von 92,66% zeigt aber auch, dass das Modell die verrauschten Daten nicht auswendig gelernt zu haben scheint.

Auch für dieses Modell wurde der Einfluss der Mindestkonfidenz auf die Ergebnisgenauigkeit untersucht. Die Abbildung 8.7 zeigt die untersuchten Mindestkonfidenzen im Verhältnis zu den Evaluationsmetriken.

Die Untersuchungen haben gezeigt, dass die Mindestkonfidenz von 60% und höher die Präzision auf bis zu 96% verbessern. Gleichzeitig jedoch, wird das Recall, also die Aussage darüber, wie viele richtige Aktivitäten erkannt wurden, schlechter. Als Ergebnis verschlechtert sich durch das sinkende Recall auch das F1-Score.
Die Ergebnisse sind durch mehrere Trainingsdurchläufe entstanden. Aus Gründen der

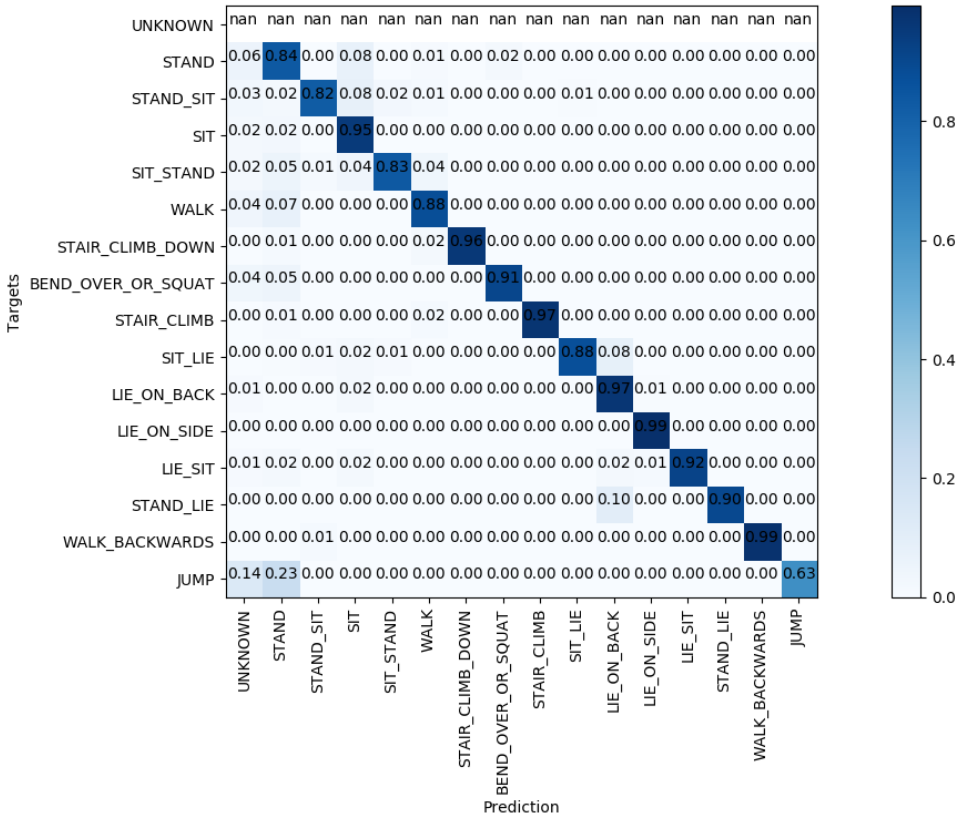


Abbildung 8.6.: CNN4Har3 Klassifikationsergebniss mit Labelgruppe full

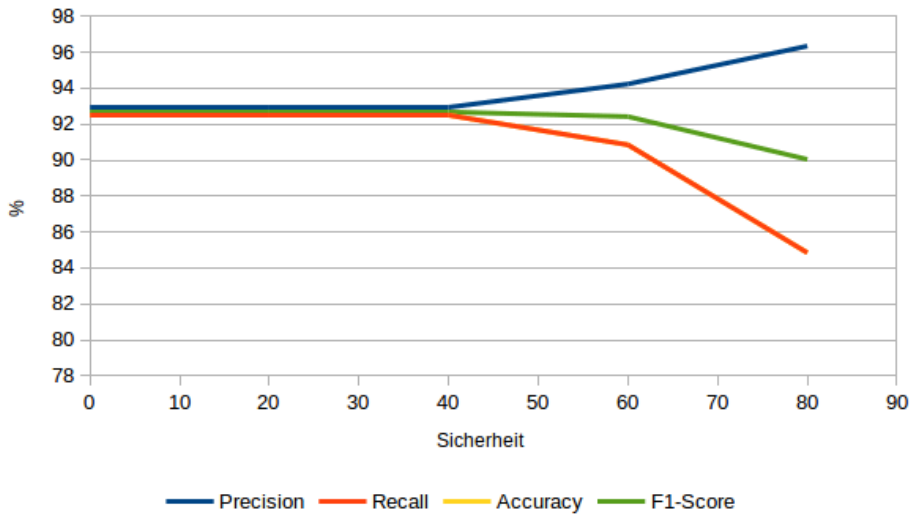


Abbildung 8.7.: CNN4Har3 Mindestkonfidenzen im Verhältnis zu Metriken, Precision, Recall, Accuracy und F1.

Übersichtlichkeit werden alle vollständigen Trainings-Protokolle dem Anhang beigefügt.

8.2.4. Interpretation der Ergebnisse

Das entwickelte Modell ist sehr anpassungsfähig und kann auch nach dem Training an sich ändernde Rahmenbedingungen angepasst werden. Es ist möglich, das Modell grob- oder feingranular zu nutzen. Dabei muss lediglich die Schrittweite pro Klassifikationsschritt angepasst werden. Das Modell klassifiziert dann mit der Granularität der Schrittweite unter Berücksichtigung des Fensters von 150 Signalen.
Das Modell kann, wie in der Seminararbeit zu faltenden neuronalen Netzen bereits gezeigt auch für den Einsatz in mobilen Systemen, wie RaspberryPI eingesetzt werden und ist in der Lage quasi realzeitfähige Ergebnisse zu liefern.

Je nach Einsatzbedingungen kann das finale Modell in seiner Sensitivität beziehungsweise Spezifität variabel eingesetzt werden. Um dies zu erreichen ist es lediglich notwendig, die Mindestkonfidenz zu erhöhen, beziehungsweise zu verringern.
Da die Datenbasis sehr unausgeglichen ist und einige Aktivitäten viel seltener vorkommen als andere, ist zu befürchten, dass das Ergebnis der Evaluation nicht aussagekräftig ist. Ein weiteres Problem, das im Rahmen dieser Optimierung nicht verbessert werden kann ist die Tatsache, dass Aktivitäten wie Stehen/Sitzen, sowie Gehen/Treppesteigen in den Daten des Accelerometers und des Gyroskops quasi identisch aussehen. Ein Möglichkeit das Stehen vom Sitzen zu unterscheiden, wäre es die entsprechenden Ausschnitte statistisch zu untersuchen, um beispielsweise des Schwanken im Stehen festzustellen. Dies hätte zur Folge, dass so ein Verfahren naturgemäß nicht mehr echtzeitfähig wäre.

8.3. Quadratische Diskriminanzanalyse

Die Diskriminanzanalyse ist eine Methode der multivariaten Verfahren in der Statistik und dient der Unterscheidung von zwei oder mehreren Gruppen, die mit mehreren Merkmalen beschrieben werden. Bei der, wie in dieser Projektgruppe angewendet, quadratischen Diskriminanzanalyse (QDA) wird nicht davon ausgegangen, dass die Gruppen gleiche Kovarianzmatrizen aufweisen. Wie bei der linearen Diskriminanzanalyse werden Beobachtungen jeweils in die Gruppe klassifiziert, die die kleinste quadrierte Distanz aufweist. Die quadrierte Distanz wird jedoch nicht auf eine lineare Funktion reduziert, daher der Name „quadratische Diskriminanzanalyse“.

Anders als die lineare Distanz ist die quadratische Distanz nicht symmetrisch. Mit anderen Worten, die quadratische Diskriminanzfunktion von Gruppe i, bewertet mit dem Mittelwert von Gruppe j ist nicht gleich der quadratischen Diskriminanzfunktion von Gruppe j, die mit dem Mittelwert von Gruppe i bewertet wird. In den Ergebnissen wird die quadratische Distanz als verallgemeinerte quadrierte Distanz bezeichnet. Wenn die Determinante der Kovarianzmatrix der Stichprobengruppe kleiner als 1 ist, kann die verallgemeinerte quadrierte Distanz negativ sein.

Die Durchführung einer Diskriminanzanalyse lässt sich in sechs Teilschritte zerlegen, wie das folgende Ablaufdiagramm in Abbildung 8.8 darstellt:

1. Definition der Gruppen
Die Diskriminanzanalyse beginnt mit der Definition der Gruppen. Diese kann sich unmittelbar aus dem Anwendungsproblem ergeben (z.B. Gruppierung von Käufern nach Produktmarken). Sie kann aber auch das Resultat einer vorangegangenen Analyse sein. Mit der Definition der Gruppen ist auch die Festlegung der Anzahl der Gruppen, die die Diskriminanzanalyse berücksichtigen soll, verbunden.

2. Formulierung der Diskriminanzfunktion
Im nächsten Schritt ist eine Diskriminanzfunktion oder auch Trennfunktion zu formulie-

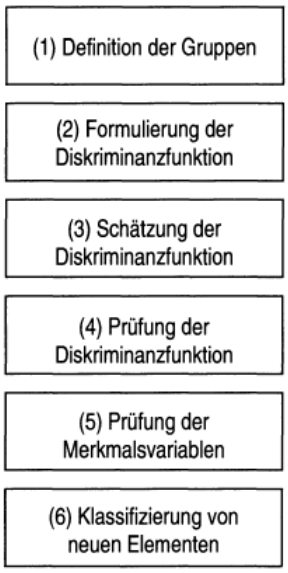


Abbildung 8.8.: Ablauf Diskriminanzanalyse

ren und zu schätzen, die dann eine optimale Trennung zwischen den Gruppen und eine Prüfung der diskriminatorischen Bedeutung der Merkmalsvariablen ermöglicht, welches für den späteren Verlauf von Bedeutung ist.

3. Schätzung der Diskriminanzfunktion

Die Schätzung der Diskriminanzfunktion bzw. der unbekannten Koeffizienten b_j in der Diskriminanzfunktion erfolgt so, dass sie im möglichen Fall optimal zwischen den gegebenen Gruppen trennt.

4. Prüfung der Diskriminanzfunktion

Die Trennkraft einer Diskriminanzfunktion lässt die Unterschiedlichkeit der Gruppen messen. Eine Möglichkeit zur Prüfung der Diskriminanzfunktion ist, die Untersuchungsobjekte mit deren tatsächlicher Gruppenzugehörigkeit zu vergleichen

Um die Klassifikationsfähigkeit einer Diskriminanzfunktion richtig beurteilen zu können, muss deren Trefferquote mit derjenigen Trefferquote verglichen werden, die man bei einer rein zufälligen Zuordnung der Elemente erreichen würde.

5. Prüfung der Merkmalsvariablen

Wichtig könnte zudem sein, die Merkmalsvariablen in der Diskriminanzfunktion beurteilen zu können. Zunächst, um die Unterschiedlichkeit der Gruppen zu erklären, und außerdem, um unwichtige Variablen aus der Diskriminanzfunktion zu entfernen. Hier kann der allgemein übliche F-Test verwendet werden. Das Ergebnis entspricht dann einer einfachen Varianzanalyse zwischen Gruppierungs- und Merkmalsvariable.

6. Klassifizierung von neuen Elementen

Für die Klassifikation von neuen Elementen sind grundsätzlich drei verschiedene Konzepte vorhanden: Klassifizierung mittels Distanzen, mittels Klassizierungsfunktionen und über Wahrscheinlichkeitsberechnungen.

Ausgewählt wurde die Quadratische Diskriminanzanalyse nach einem Test mit mehreren Algorithmen auf den Daten der vorherigen MAMKS-Projektgruppe, wobei die Quadratische Diskriminanzanalyse eine Gesamtgenauigkeit von 89% aufwies. Durch die ohne Optimierung bereits hohe Genauigkeit, sollte dieser Ansatz weiter verfolgt und verbessert werden. Dies wurde zuerst mit Matlab und später mit Python realisiert.

8.3.1. Optimierung in Matlab

Zuerst wurden die Assessment-Daten der MAMKS Projektgruppe herangezogen um die ersten Ergebnisse zu erhalten, da wir zu diesem Zeitpunkt noch keine anderen Daten zur Verfügung hatten. Die Ergebnisse, sind in den nachfolgenden Abbildungen einzusehen. Die Daten basieren auf den Erhebungen der MAMKS-Projektgruppe und sollten weiter optimiert werden. Hierzu wurde zum Anlernen die Matlab-App Classification Learner genutzt. Weiterhin wurde eine Cross-Validation von 5 benutzt, um Overfitting zu vermeiden.

Matlab bietet für die Optimierung der QDA nur wenige Parameter, das liegt jedoch auch daran, dass die QDA allein wenig zu optimierende Funktionen besitzt. Die wenigen

Parameter die in der Matlab-App eingestellt werden können sind:

1. Covariance Structure Full
2. Covariance Structure Diagonal

Aus der „Classification Learner App“ konnte als nächster Schritt ein Skript generiert werden, welches zusätzlich zu den bisherigen anzupassenden Parametern noch zusätzliche Optimierungen der QDA zulässt. Auf dieser Ebene kann man z.B. den Delta und Gamma Wert anpassen, sowie die „Hyperparameter Optimization Option “ (HOO), also ob Bayes, GridSearch oder RandomSearch verwendet werden soll. Anschließend wurden die verschiedenen Einstellungen untereinander kombiniert, um die besten Ergebnisse zu finden. Die besten Ergebnisse wurden jedoch ausschließlich mit der Standardeinstellung erzielt. Diese Standardeinstellungen sind die Parameter und der gesetzte Wert:

1. Delta = 0
2. Gamma = 0
3. HOO = Bayes
4. PCA = off

Die Ergebnisse dieser Daten sind in Abbildung 8.9 und Abbildung 8.10 einzusehen. Um die gezeigten Ergebnisse besser verstehen zu können, sind einige Grundbegriffe nötig. Nachfolgend eine kurze Erklärung der verwendeten Begriffe und eine kurze Erklärung:

1. TP = true positives, richtig als richtig erkannt
2. FN = false negatives, falsch als falsch erkannt
3. FP = false positives, falsch als richtig erkannt
4. TN = true negatives, richtig als falsch erkannt

Aus diesen Grundwerten lassen sich folgende Berechnungen für die Einzelwerte in der Konfusionsmatrix vornehmen.

$$precision = \frac{TP}{(TP + FP)} \tag{8.1}$$

$$recall = \frac{TP}{(TP + FN)} \tag{8.2}$$

$$f1score = \frac{2TP}{(2TP + FP + FN)} \tag{8.3}$$

in Abbildung 8.9 wird eine False-Positive Matrix gezeigt, die die verschiedenen Aktivitäten gegenüberstellt und anzeigt wie gut diese vom Algorithmus erkannt worden sind. In dieser Matrix kann man z.B. sehr gut sehen, dass SIT mit 95% und STAND mit 89% von der QDA erkannt wurden. Jedoch wurden Aktivitäten wie STAIR_CLIMB oder LIE_SIT nur mit etwa 20% erkannt.

Die Abbildung 8.10 zeigt, im Gegensatz zu Abbildung 8.9, eine Falsch-Negativ Matrix an, die ebenfalls die verschiedenen Aktivitäten gegenüberstellt und anzeigt wie gut diese vom Algorithmus erkannt worden sind. Besonders gut erkannt wurden hier z.B. LIE_ON_BACK mit 97% und SIT mit 92%. Was jedoch auch heraussticht ist, dass STAIR_CLIMB zu 89% als WALK missinterpretiert wird. Sodass in einer Anwendung des Algorithmus, diese Art der Bewegung extrem schlecht erkannt werden würde.

Allgemein ist erkennbar, dass besonders die Aktivitäten Gehen, Stehen und Sitzen gut erkannt worden sind, mit einer Genauigkeit von ≥ 95. Die anderen Aktivitäten sind teilweise sehr schlecht bis gar nicht prognostiziert worden, weshalb hier angesetzt werden kann zu optimieren. Z.B. könnte hier Down- oder Upsampling angewandt werden, um die großen Verteilungsunterschiede der verschiedenen Aktivitäten ein wenig auszugleichen. Dies wurde erst im späteren Verlauf der Projektgruppe getan, da zu jetzigem Zeitpunkt

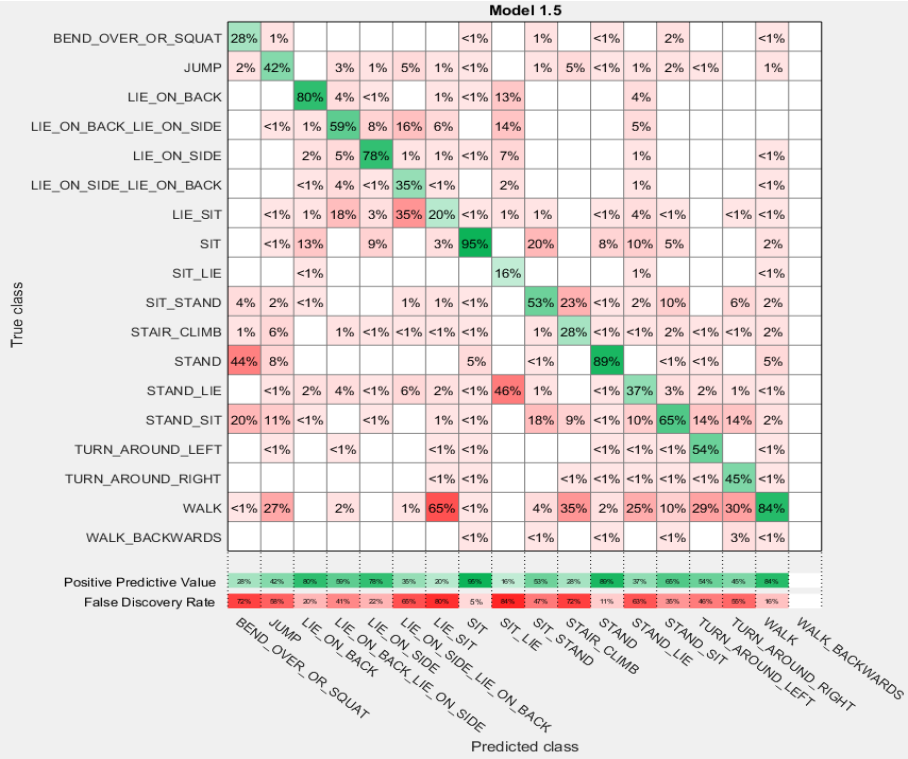


Abbildung 8.9.: Konfusionsmatrix QDA False-Positive

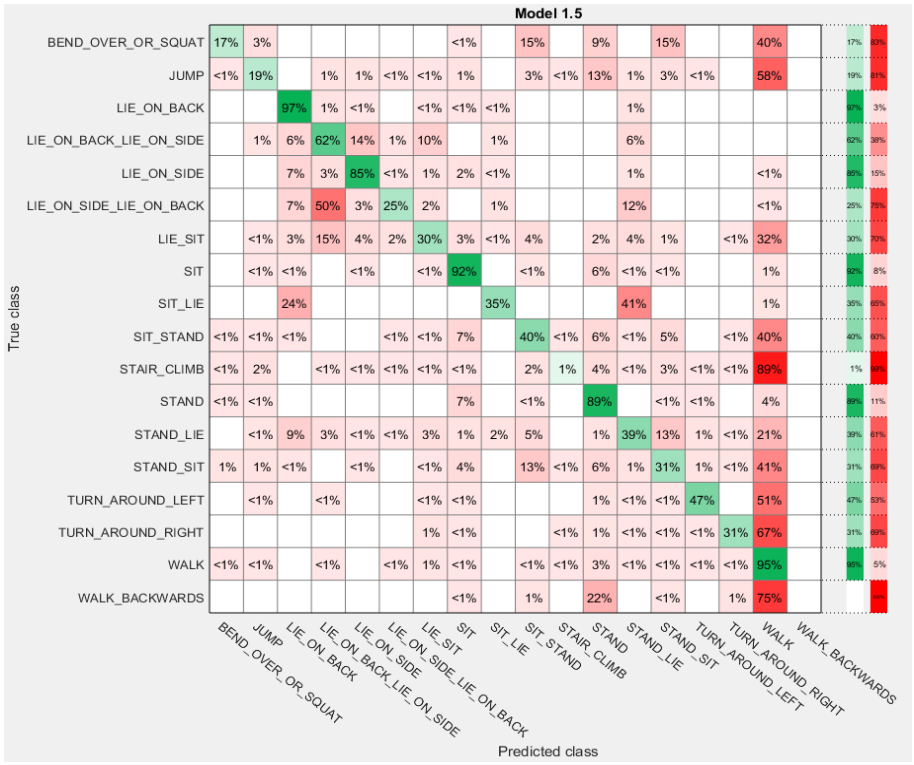


Abbildung 8.10.: Konfusionsmatrix QDA False-Negative

noch zu wenig Daten vorhanden waren.

In der nachstehenden Tabelle 8.1 sind die besten Ergebnisse noch einmal tabellarisch dargestellt. Große Unterschiede sind trotz der verschieden genutzten Parameter kaum ersichtlich.

Covariance Structure	Gamma	Delta	HOO	PCA	Accuracy in %
Full	1	0	Bayes	off	88,97%
Full	1	0	GridSearch	off	89,03%
Full	0	0	Bayes	off	89,04%
Diagonal	1	0	Bayes	off	88,97%
Diagonal	0	0	Bayes	off	89,03%

Tabelle 8.1.: Ergebnisse der MAMKS Daten mit QDA

Leider ist die Matlab Entwicklungsumgebung sehr langsam und aus unserer Sicht nicht optimal, um z.B. die QDA anzulernen und weiter zu nutzen. Es ist zu voll mit verschiedensten Funktionen und Apps, die zwar alle Funktionen bieten, die man sich für Maschinelles Lernen wünscht, jedoch wirkt es alles zu unübersichtlich. Außerdem benötigt das Anlernen in Matlab sehr viel länger als in z.B. Python. Dies kann man gut an den Zeiten erkennen die z.B. der KNN zum Anlernen benötigt (siehe dafür Optimierung KNN).

Die Probleme die in Matlab während dieser ersten Testphasen aufgetreten sind, haben uns dazu bewegt eine „leichtere Umgebung“ nutzen zu wollen. Da ebenfalls schon Erfahrung in Python vorhanden ist und die Python Bibliotheken ebenfalls wunderbar für Maschinelles Lernen geeignet sind, haben wir und entschieden einen Wechsel von der trägen Plattform Matlab hin zu der leichtgewichtigen Python Umgebung zu machen. So können wir ebenfalls direkte Vergleiche zwischen den beiden Entwicklungsumgebungen machen. Den Wechsel zu Python haben die Quadratische Diskriminanzanalyse, K-Nearest-Neighbour und Convolutional Neural Networks Optimierer bewusst durchgeführt, nicht aber die Bagged Trees und Und Boosted Dicision Trees Optimierer. Diese

wollten sich weiterhin mit den Möglichkeiten, die Matlab bietet, befassen.

8.3.2. Optimierung in Python

Beim Wechsel zu Python, wurde zuallererst versucht die selben Datensätze wie bei Matlab zu verwenden, um die selbigen Ergebnisse herauszubekommen. Nach dem dies bestätigt werden konnte, wurde im weiteren Verlauf der Projektgruppe für alle Optimierer festgelegt, welche Datensätze für die Algorithmen verwendet werden sollen. Dabei wurden die Datensätze in Tabelle 8.2 verwendet.

Die Datensätze unterscheiden sich in der Anzahl der betrachteten Merkmale. In der Tabelle ist zu erkennen, welche Merkmale in welchem Datensatz vorkommen und welche ausgelassen worden sind. Die Umsetzung der QDA in Python wurde analog zur Umsetzung des KNN vorgenommen, welches in Kapitel 8.6.4 zu finden ist, weshalb z.B. der Mehrheitsentscheid der QDA in diesem Kapitel weggelassen worden ist. Dieser findet sich bei der Modelloptimierung KNN Labelgenerierung.

Datensatz 0	Datensatz 1	Datensatz 2	Datensatz 3
Stehen	Stehen	Stehen	Stehen
Sitzen	Sitzen	Sitzen	Sitzen
Gehen	Gehen	Gehen	Gehen
Treppensteigen (hoch)	Treppensteigen (ohne Richtung)	Treppensteigen (ohne Richtung)	Treppensteigen (ohne Richtung)
Treppensteigen (runter)	Hinsetzen	Hinsetzen	Hinsetzen
Hinsetzen	Aufstehen	Aufstehen	Aufstehen
Aufstehen	Liegen (Rücken)	Liegen (Rücken)	Liegen (Rücken)
Liegen (Rücken)	Liegen (Seite)	Liegen (Seite)	Liegen (Seite)
Liegen (Seite)	Hocken		Hocken
Hocken	Springen	Springen	
Springen	Liegen -> Sitzen		
Liegen -> Sitzen	Sitzen -> Liegen		
Sitzen -> Liegen	Gehen Rückwärts		
Gehen Rückwärts			
Stehen -> Liegen			

Tabelle 8.2.: Genutzte Datensätze

Für Python sind ebenfalls zusätzliche Pakete notwendig, die nachträglich installiert werden müssen. Diese sind wie folgt aufgelistet:

1. pandas
2. matplotlib
3. sklearn
4. datetime
5. pickle

Mit Hilfe der Daten aus Tabelle 8.2 ist die QDA mehrere Male angelernt und untereinander verglichen worden. Die „Accuracy“ und die Zeit die der Algorithmus zum Anlernen benötigt hat wird in der nachstehenden Tabelle 8.3 gezeigt. Besonders auffällig war, dass das Anlernen der QDA und das klassifizieren, noch schneller waren als in Matlab. Zum Trainieren wurde das Paket „QuadraticDiscriminantAnalysis“ von sklearn verwendet.

Datensatz	Accuracy in %	Zeit in s
0	79,911%	28,99s
1	79,926%	28,83s
2	80,274%	28,79s
3	79,991%	28,88s

Tabelle 8.3.: Ergebnisse des Anlernens mit QDA

Da „Datensatz 0“ alle von uns aufgezeichneten Merkmale nutzt und selbige Genauigkeiten wie die anderen Datensätze ausgibt, wird dieser Datensatz weiterverfolgt. Nun galt es, diese Genauigkeit noch weiter zu erhöhen. Eine Idee war es, die Daten ein wenig mittels Downsampling oder Upsampling anzupassen. Mit Hilfe der Python Bibliotheken „Imbalanced-learn“ ist es möglich vorgefertigte Downsampling Algorithmen verwenden zu können. Es wurden drei Algorithmen ausgewählt, die schnell sind aber dennoch zu-

verlässig arbeiten.

Die drei getesteten Downsampling Algorithmen sind Edited Nearest Neighbours (ENN), Repeated Edited Nearest Neighbours (RENN) und All-K Nearest Neighbours (ALL-KNN). Alle Verfahren zeigten eine höhere Genauigkeit beim Anlernen der QDA. Jedoch sind diese Genauigkeiten mit Vorsicht zu genießen, da nicht nur die Daten entfernt wurden, die besonders oft vorkommen, wie Sitzen, Gehen oder Stehen, sondern auch teilweise Daten die nur selten vorkommen, wie Treppe steigen oder die Transition Stehen-Sitzen. Alle drei Downsampling Algorithmen wurden auf „Datensatz 0“ angewandt und die Ergebnisse in der Tabelle 8.4 zusammengefasst.

Algorithmus	Accuracy in %	Zeit in s
ENN	88,353%	23,49s
RENN	89,466%	22,33s
ALLKNN	88,751%	23,21s

Tabelle 8.4.: Ergebnisse nach Downsampling von Datensatz 0

Das Klassifizieren und die Labelgenerierung wurden, wie am Anfang dieses Abschnittes bereits erwähnt, gemeinsam mit dem KNN entwickelt. Um doppelte Einträge zu vermeiden, wird hier auf Abschnitt 8.6.4 verwiesen, wo genau beschrieben wird, wie der weitere Verlauf der Optimierung in Python ablief.

8.3.3. Interpretation der Ergebnisse

Anhand der vorliegenden Ergebnisse, die mit der QDA nicht weiter verbessert werden konnten, lässt sich feststellen, dass mit diesem Algorithmus leider nicht die Ergebnisse erzielt werden konnten, die wir uns vorgestellt hatten. Die Ergebnisse sind mit Hilfe des Mehrheitsentscheides, jedoch fast gleichwertig, wie die des KNN. Leider treten dennoch zu oft Fehlklassifikationen auf, sodass die QDA nicht im Produktiven Bereich genutzt werden könnte. Jedoch gab es Beobachtungen, dass die QDA bestimmte Transitionen

wie hinsetzen besser erkennen konnte, als z.B. KNN oder CNN. Jedoch trat dies nur sporadisch auf und war nicht übertragbar auf alle klassifizierten Daten.

Des Weiteren kann beobachtet werden, dass Aktivitäten, von denen mehr Daten vorhanden waren, zuverlässiger erkannt wurden. Aus diesem Grund ist besonders die Erkennung von Gehen, Stehen und Sitzen mit sehr hohen Wahrscheinlichkeiten vertreten. Probleme treten allerdings bei besonders kurzen Aktivitäten auf, oder bei Aktivitäten die beim Anlernen nur wenig vorhanden waren. Beispielsweise die Transitionen wie das Hinsetzen oder Aufstehen können nicht immer zuverlässig klassifiziert werden.

8.3.4. Ausblick

Das Modell kann gut als Grundlage für die Klassifizierung von Bewegungsdaten verwendet werden, ist jedoch nicht gut genug, um wirklich genaue Aussagen treffen zu können. Man könnte allerdings durch eine Nachbereitung die Ergebnisse noch verbessern. Jedoch bietet es sich an, dafür die anderen genutzten Algorithmen zu verwenden, da diese eine höhere Grunderkennungsgenauigkeit haben. Besonders gut geeignet ist der Algorithmus jedoch für schnelle und Speicherkritische Bereiche. Wie in Tabelle 8.8 und Tabelle 8.9 zu erkennen ist, benötigen die angelernten QDA Dateien nur minimalen Speicherplatz und wären somit auch für mobile Laufzeitumgebungen geeignet.

Die geringe Genauigkeit, die in dieser Untersuchung auftrat, lässt sich entweder auf schlecht gelabelte Daten zurückführen oder auf die zu wenig vorhandenen Aktivitäten, wie „Treppe steigen“ oder „Hinsetzen“. Leider ist ein großes Ungleichgewicht in den Daten vorhanden, was sich wahrscheinlich in den Ergebnissen des Klassifikators kenntlich macht. Zur näheren Betrachtung der QDA, sei auf die Seminararbeit Diskriminanzanalyse verwiesen. Auch durch Upsampling oder Downsampling konnten die Daten nicht so aufbereitet werden, dass sie optimal von der QDA genutzt werden konnten.

Zusammenfassend lässt sich sagen, dass die Quadratische Diskriminanzanalyse ein span-

nender Algorithmus ist, der für viele verschiedene Sachverhalte nutzbar ist, jedoch für den weiteren Gebrauch der Projektgruppe MAMKSFZ nicht geeignet ist. Die bisher erzeugten Ergebnisse sind nicht relevant genug, um diesen Algorithmus produktiv nutzen zu können, sodass auf die anderen Algorithmen verwiesen werden muss, mit denen wir uns näher beschäftigt haben.

8.4. Bagging Trees

Im Rahmen der Optimierung wurde auch der Bagging-Trees-Algorithmus herangezogen. Hierzu wurde vorab in einer Seminararbeit die Funktionsweise und die potenzielle Eignung für diese Projektgruppe genauer untersucht. Diese Ausarbeitung befindet sich im Anhang B. Folgend findet somit nur eine gröbere Beschreibung des Algorithmus statt. Auch wird folgend auf die Parameter und die Anwendung in Matlab eingegangen.

8.4.1. Einleitung

Bagging Trees bauen auf Entscheidungsbäumen auf, welche zum Supervised Machine Learning gehören und graphenbasiert sind. Entscheidungsbäume treffen mit Hilfe von Regeln Entscheidungen. Die graphische Darstellung ermöglicht eine gute Interpretierbarkeit. Ein Beispiel eines Entscheidungsbäumes ist in Abbildung 8.11 ersichtlich. Entscheidungsbäume können sowohl zur Klassifikation als auch im Regressionsbereich angewandt werden.

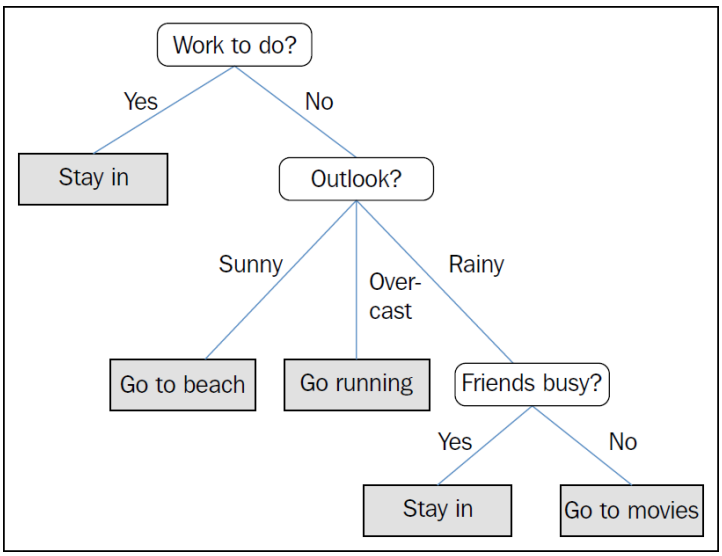


Abbildung 8.11.: Beispielentscheidungsbaum [13]

Um einen Entscheidungbaum möglichst optimal aufzubauen exisitieren mehrere Var-

fahren, wie z.B. CaRT oder ID3. Hierbei wird beispielweise mit Hilfe des Gini-Index versucht, eine Gewichtung der Attribute zu erhalten und somit den Entscheidungbaum zu gestalten. Beispielberechnungen hierzu befinden sich in der Seminararbeit.
Ein Problem der Entscheidungbäume ist das Overfitting. Dieser Effekt ist in Abbildung 8.12 dargestellt.

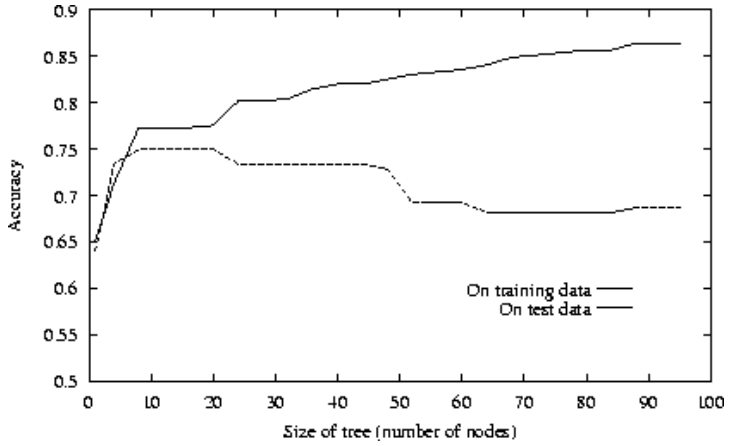


Abbildung 8.12.: Overfitting in Entscheidungbäumen[6]

Zu erkennen ist hierbei, dass mit steigender Anzahl der Knoten eines Baumes zunächst die Erkennungsrate, sowohl beim Klassizieren von Trainingsdaten als auch von Test-Daten steigt. Ab einer gewissen Größe stagniert die Erkennungsrate auf den Test-Daten jedoch. Bei einem weiter wachsenden Baum fällt die Rate sogar. Um dieser Problematik entgegenzuwirken, kann das Pruning-Verfahren genutzt werden. Dabei wird ein Entscheidungsbaum zunächst komplett erstellt. Anschließend werden jedoch eher unwichtige Knoten wieder herausgenommen, um eine zu starke Anpassung an die Trainingsdaten zu verhindern. Dies hat ebenfalls den Vorteil, dass weniger Speicherplatzbedarf benötigt wird. Um der Problematik z.B. des Overfittings noch weiter entgegenzuwirken, ist der Einsatz von Bagging denkbar.

Der Begriff „Bagging“ ist eine Zusammensetzung der beiden Wörter „**B**ootstrap **agg**regat**ing**“ und wurde durch Leo Breiman geprägt[9]. Es bezeichnet eine Methode in der mehrere verschiedene Klassifikationsmodelle angelernt werden. Diese verschiedenen

Modelle bilden anschließend einen zusammengesetzten Klassifikator der im Idealfall eine gesteigerte Vorhersagegenauigkeit besitzt. Werden für das Bagging lediglich Entscheidungsbäume angelernt, so kann dies als „Bagging Trees“ bezeichnet werden. Eingesetzt werden kann dieses Verfahren sowohl bei der Regression, als auch bei der Klassifikation welche in dieser Ausarbeitung genauer betrachtet wird. Es folgt eine Beschreibung des Ablaufes der Methode.
Zu Beginn können die zur Verfügung stehenden Daten in Test- und Trainingsdatensätze aufgeteilt werden. Anschließend werden die Trainingsdaten durch Ziehen von Bootstrap-Stichproben in einzelne, unabhängige Datensätze aufgeteilt. Dabei wird pro anzulernendem Modell eine Stichprobe benötigt. Die Bootstrap-Stichproben werden dabei nach einem definierten Verfahren gezogen. Stehen beispielsweise insgesamt 30 Trainingssätze zur Verfügung, wird eine definierte Anzahl davon gezogen. Beispielsweise vier Sätze, womit jede Stichprobe immer vier Datensätze enthält. Beim Ziehen der Sätze kann es auch vorkommen, dass ein bereits gezogener Datensatz erneut gezogen wird. Nach dem Erstellen der Stichproben, oder auch „bags“ genannt, wird daraufhin zu jeder einzelnen Stichprobe ein Klassifikationsmodell angelernt. Das gesamte Verfahren wird in 8.13 skizziert.

Die erlernten Modelle bilden nun im Zusammenschluss das fertige Klassifikationsmodell. Zur Klassifikation wird der zu klassifizierende Datensatz an jedes einzelne Teilmodel gesendet. Diese geben jeweils eine Klasse zurück, welche als „Vote“ bezeichnet werden kann. Per Mehrheitsentscheid wird somit final entschieden um welche Klasse es sich handelt. Skizziert ist dieser Vorgang in 8.14.

8.4.2. Zu optimierende Parameter

Generell gibt es diverse Parameter, die bei dem Bagging-Trees Algorithmus verändert werden Können. Bei den Entscheidungsbäumen kann beispielsweise festgelegt werden, ob dieser nach dem CaRT-verfahren oder nach dem ID3-Verfahren aufgebaut wird. Auch

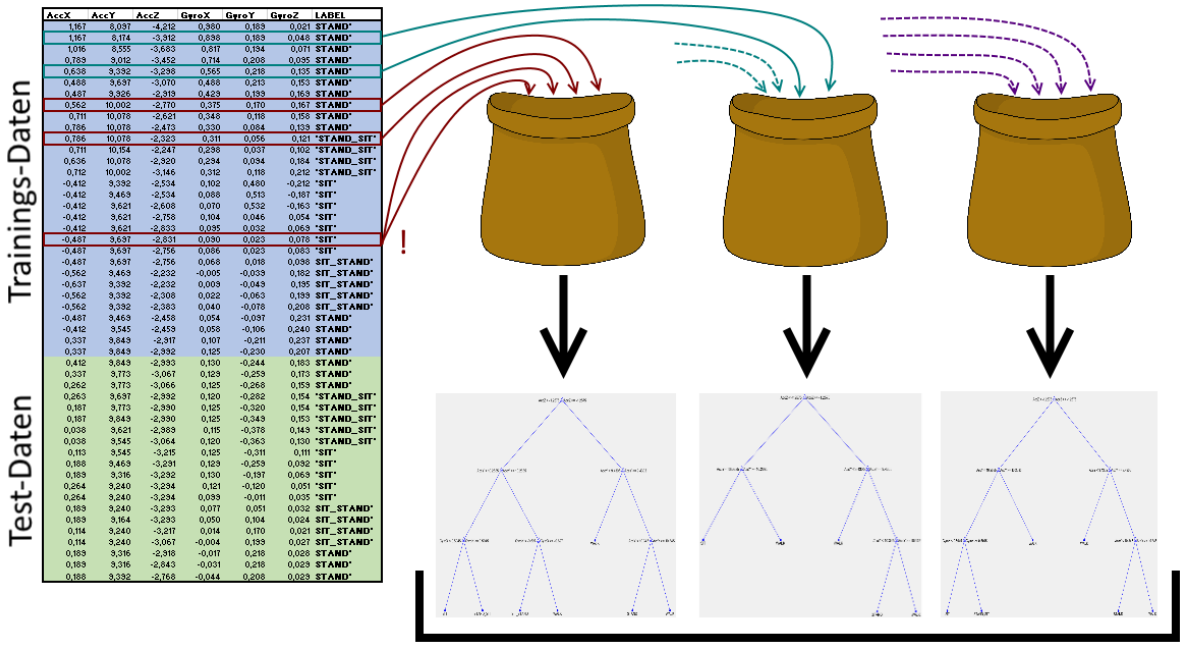


Abbildung 8.13.: „bagging“-Methode: Erstellung

ob die Bäume auf eine bestimmte Anzahl an Splits begrenzt wird kann festgelegt werden. Denkbar ist weiter die Festlegung der Bootstrap-Stichproben oder die Vorgabe der Menge an Elementen, welche je Bag enthalten sein soll. Es kann ebenfalls angepasst werden, ob je Bag eine Mehrfachziehung der gleichen Elemente möglich ist oder durch eine Regel verhindert wird. Zu nennen ist auch, ob die Entscheidungsbäume gestutzt werden oder komplett Auswachsen.

Nach der Definition von Leo Breimann[9] gibt es jedoch Vorgaben, welche Parametereinstellung für Bagging-Trees genutzt werden sollen. In der Seminararbeit**??** wurde erkannt, dass der relevanteste Parameter lediglich die Anzahl der genutzten Bags/ Bootstrap-Stichproben ist. Die anderen Parametereinstellung wurden somit weitestgehend von der Definition übernommen. Aus der Seminararbeit ging auch hervor, dass das Stutzen der Entscheidungsbäume zwar zu einem deutlich geringeren Speicherplatzbedarf führte. Dadurch wurde jedoch die Genauigkeit bei der Klassifikation enorm gesenkt. Aus diesem Grund wurde auf das Stutzen verzichtet.

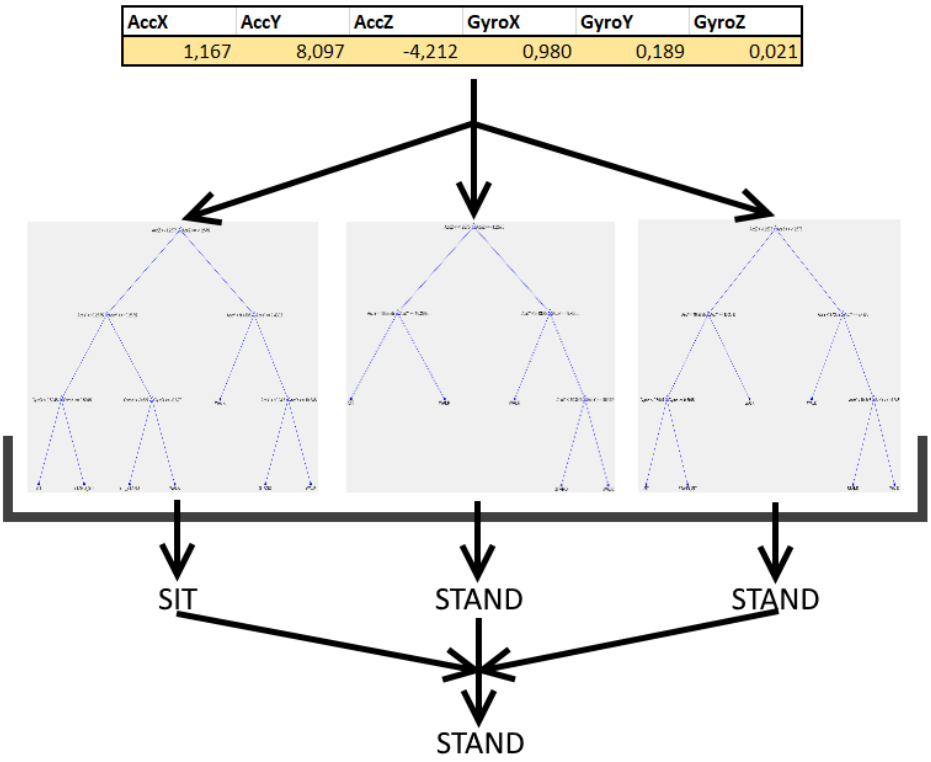


Abbildung 8.14.: „bagging“-Methode: Klassifizierung

8.4.3. Optimierungen in Matlab

Zum antrainieren der Klassifikatoren wurde Matlab in der Version R2016b genutzt. Die Berechnungen wurden hauptsächlich auf dem HPC Cluster (Kapitel 8.1) durchgeführt. Da dort noch keine neuere Version zur Verfügung stand, wurde das Training mit der Version R2016b durchgeführt.

Das Training wurde auf vier verschiedenen Datensätzen durchgeführt. Diese sind folgend in Abbildung 8.15 einzusehen.

Datensatz 0	Datensatz 1	Datensatz 2	Datensatz 3
Stehen	Stehen	Stehen	Stehen
Sitzen	Sitzen	Sitzen	Sitzen
Gehen	Gehen	Gehen	Gehen
Treppensteigen (hoch)	Treppensteigen (ohne Richtung)	Treppensteigen (ohne Richtung)	Treppensteigen (ohne Richtung)
Treppensteigen (runter)	Hinsetzen	Hinsetzen	Hinsetzen
Hinsetzen	Aufstehen	Aufstehen	Aufstehen
Aufstehen	Liegen (Rücken)	Liegen (Rücken)	Liegen (Rücken)
Liegen (Rücken)	Liegen (Seite)	Liegen (Seite)	Liegen (Seite)
Liegen (Seite)	Hocken		Hocken
Hocken	Springen	Springen	
Springen	Liegen -> Sitzen		
Liegen -> Sitzen	Sitzen -> Liegen		
Sitzen -> Liegen	Gehen Rückwärts		
Gehen Rückwärts			
Stehen -> Liegen			

Abbildung 8.15.: Genutzte Datensätze

Dabei enthält Datensatz 0 alle Label, welche nach dem Herausfiltern von z.B. UNKNOWN überbleiben. Datensatz 1 verzichtet auf eine Unterscheidung von Treppehinauf- und Treppeheruntergehen sowie die Transition „Stehen -> Liegen“. Diese wurde entfernt, da sie in der Praxis nahezu nie vorkommt. Datensatz 2 verzichtete auf weitere Transitionen und Datensatz 3 ähnelt in seinen Labeln stark den Datensätzen der vorherigen Projektgruppe.

Bei jedem antrainieren wurden je 30 Bags bzw. Bootstrap-Stichproben eingesetzt. Ei-

ne höhere Anzahl gewinnt nur marginal an Genauigkeit, benötigt jedoch deutlich mehr Speicherplatz. Diese Erkenntnisse gehen aus der Seminararbeit hervor. Auch wurde wieder eine 5-Folds-Validierung genutzt, welche zwar rechenintensiv ist, dafür ein relativ genaues und vergleichbares Ergebnis erzielt.

Das Ergebnis des Trainings ist in Abbildung 8.16 dargestellt.

Datensatz	Speicherplatzbedarf	Ergebnis
0	118 GByte	91,73%
1	100 GByte	92,17%
2	97 GByte	92,28%
3	99 GByte	92,19%

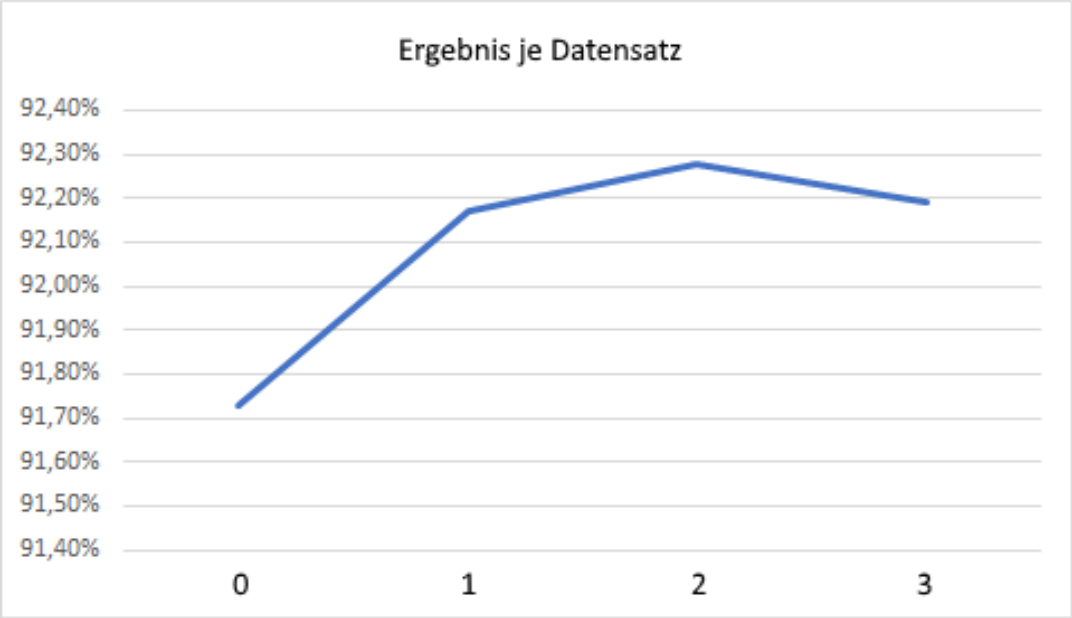


Abbildung 8.16.: Ergebnisse des Trainings

Zu Erkennen ist dabei eine Verbesserung der Ergebnisse durch das weglassen bestimmter Label. Zwischen Datensatz 0 bis 2 ist dieser Trend im dargestellten Liniendiagramm klar erkennbar. Das Tauschen der Aktivitäten Springen und Hocken führt wiederum zu einer kleinen Verschlechterung.
Zu beachten ist jedoch auch der sehr hohe Speicherplatzbedarf der Klassifikatoren. Dieser schwankt zwischen 97 bis 118 Gigabyte. Der genutzte Datensatz benötigte ca. 6,7

Gigabyte Speicherplatz. Interessant ist, dass das beste Ergebnis mit dem Klassifikator erzielt wurde, welcher am wenigsten Speicherplatzbedarf hat. Die Unterschiede sind jedoch gering und verantwortlich hierfür könnte auch eine schlechte Qualität der weggelassenen Label sein.

Bei der Durchführung der Berechnungen auf dem HPC kam es mehrfach zu Abbrüchen. In fast allen Fällen lag dies an unzureichendem Arbeitsspeicher. Erfolgreich liefen die Berechnung erst bei der folgenden Job-Konfiguration:

```
set(sched, 'CommunicatingSubmitFcn',
cat(2, sched.CommunicatingSubmitFcn, {'diskspace','100000M'}));

set(sched, 'CommunicatingSubmitFcn',
cat(2, sched.CommunicatingSubmitFcn, {'memory','250G'}));

set(sched, 'CommunicatingSubmitFcn',
cat(2, sched.CommunicatingSubmitFcn, {'runtime','288:0:0'}));
```

Dabei wurden insgesamt 16 Nodes genutzt. Dementsprechend mussten im Produkt vier Terabyte Arbeitsspeicher reserviert werden. Der Job lief dennoch über einige Tage. Dies bestätigt die Beobachtung aus der Seminararbeit, dass dieser Algorithmus sehr rechenintensiv und speicherhungrig ist. Zu beachten ist jedoch, dass ein Teil der benötigten Rechenzeit dem Validierungsverfahren geschuldet ist.

Der Code zum Anlernen der Klassifikatoren konnte teilweise durch die in Matlab integrierte *Classification Learner App* nach der Durchführung auf Testdatensätze exportiert werden. Der Code zum Anlernen des ersten Datensatzes sieht wie folgt aus:

```
function [trainedClassifier, validationAccuracy] =
trainClassifier_1b(trainingData)
inputTable = trainingData;
predictorNames = {'AccX', 'AccY', 'AccZ', 'GyroX', 'GyroY', 'GyroZ'};
```

```
predictors = inputTable(:, predictorNames);
response = inputTable.Label;
isCategoricalPredictor = [false, false, false, false, false, false];

classificationEnsemble = fitensemble(predictors, response,'Bag', ...
    30, 'Tree', 'Type', 'Classification', 'ClassNames',
    {'BEND_OVER_OR_SQUAT'; 'JUMP'; 'LIE_ON_BACK'; 'LIE_ON_SIDE';
    'LIE_SIT'; 'SIT'; 'SIT_LIE'; 'SIT_STAND'; 'STAIR_CLIMB'; 'STAND';
    'STAND_SIT'; 'WALK'; 'WALK_BACKWARDS'});

predictorExtractionFcn = @(t) t(:, predictorNames);
ensemblePredictFcn = @(x) predict(classificationEnsemble, x);
trainedClassifier.predictFcn = @(x)
ensemblePredictFcn(predictorExtractionFcn(x));
trainedClassifier.RequiredVariables = {'AccX', 'AccY', 'AccZ',
'GyroX', 'GyroY', 'GyroZ'};
trainedClassifier.ClassificationEnsemble = classificationEnsemble;

inputTable = trainingData;
predictorNames = {'AccX', 'AccY', 'AccZ', 'GyroX', 'GyroY', 'GyroZ'};
predictors = inputTable(:, predictorNames);
response = inputTable.Label;
isCategoricalPredictor = [false, false, false, false, false, false];

partitionedModel = crossval(trainedClassifier.ClassificationEnsemble,
'KFold', 5);
validationAccuracy = 1 - kfoldLoss(partitionedModel,
'LossFun', 'ClassifError');
[validationPredictions, validationScores] =
kfoldPredict(partitionedModel);
```

An die Funktion *trainClassifier_1b* muss der Datensatz mit benötigten Spalten übergeben werden. Die Funktion *fitensemble* trainiert den Klassifikator an. Dafür müssen die enthaltenen Label sowie Parameter wie z.B. die Anzahl an Bags übergeben werden. Das Ergebnis wird anschließend wiederum in eine Funktion gefasst und durchläuft die Validierung. Zurückgegeben wird am Ende die erreichte Genauigkeit sowie das trainierte Model.

Um auch in Matlab direkt die gelabelten Daten zu analysieren wurde eine Plot-Funktion geschrieben. Diese ermöglicht es bis zu vier Labeldatensätze und die dazugehörigen Daten auf für Beschleunigung, Gyroskop und Magnetometerdaten anzuzeigen. Dabei kann beliebig gezoomt werden. Die Farben für die Labelaktivitäten sind dabei Kräftiger dargestellt (siehe Abbildung 8.17). Die Performanz stellte sich auch bei größeren Datensätzen als gut heraus.

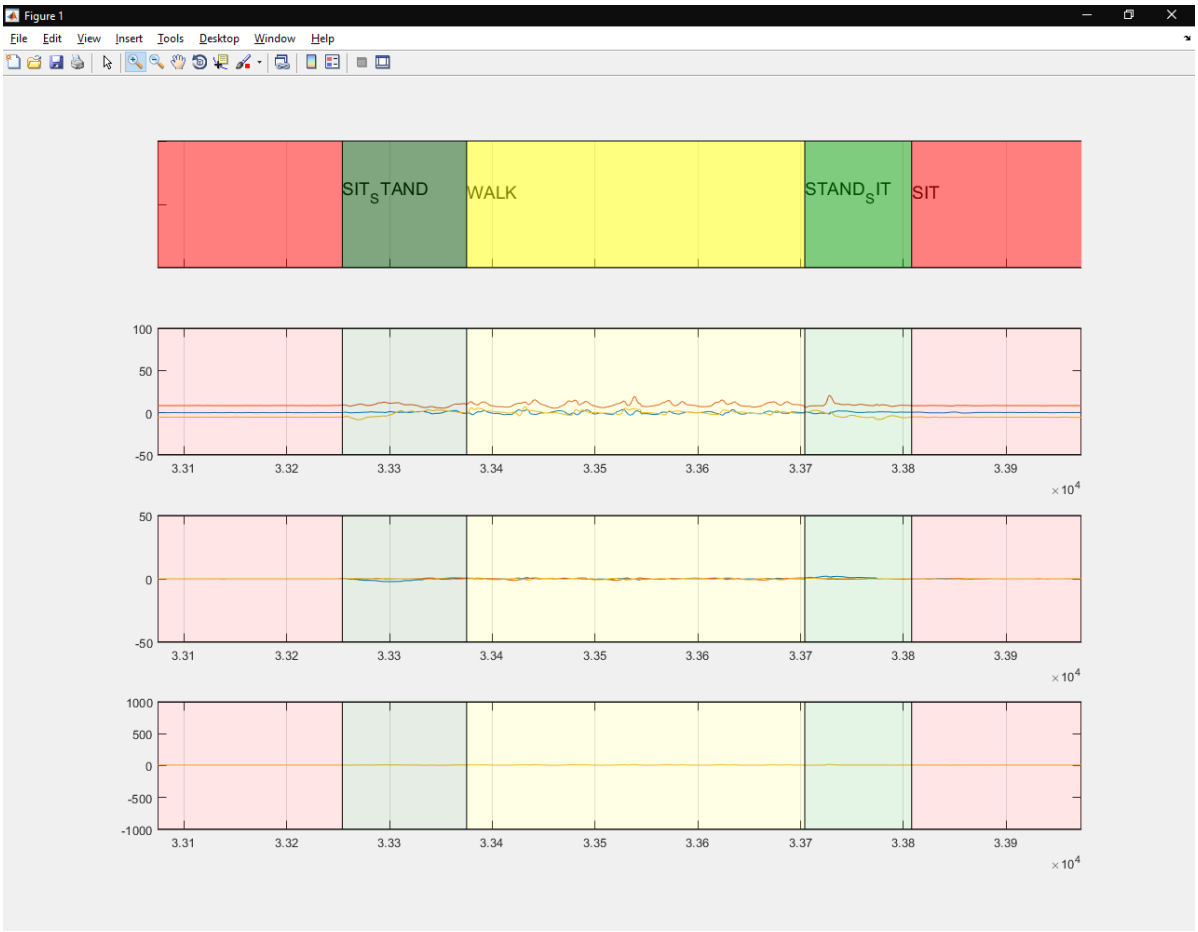


Abbildung 8.17.: Plot-Funktion in Matlab

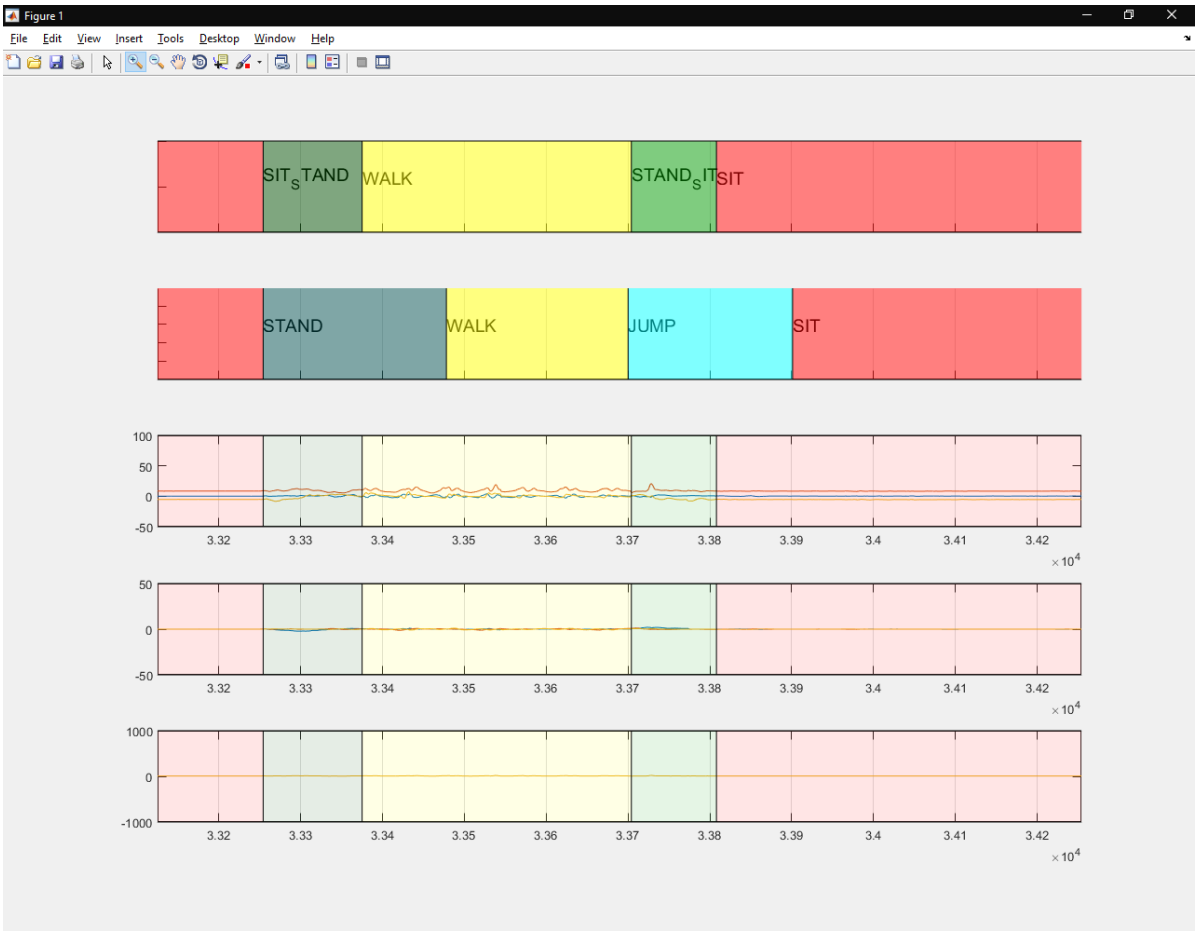


Abbildung 8.18.: Plot-Funktion in Matlab mit zwei Labelgruppen

8.5. Boosted Decision Tree

In diesem Abschnitt wird der Prozess der Optimierung für den Algorithmus beschrieben. Da es keine Seminararbeit zu dem Thema Boosted Decision Tree gab, wird der Algorithmus im Vorfeld kurz beschrieben. Diese Einleitung soll dem Leser helfen die nachfolgenden Unterkapitel besser zu verstehen. Im Anschluss werden die einstellbaren Parameter vorgestellt und auf deren Auswirkung eingegangen. Daran anknüpfend wird aufgezeigt, wie diese Parameter in die Optimierung eingeflossen sind. Dabei werden Zwischenergebnisse gezeigt und diese interpretiert.

8.5.1. Einführung

Ein Entscheidungsbaum (Decision Tree) besteht aus beliebig vielen Knotenpunkten, die eingehende Elemente dem zugehörigen nächsten Knotenpunkt zuweisen, bis diese an den Blättern angekommen sind (siehe Abbildung 8.19). An einem Knotenpunkt wird das aktuelle Element also anhand seiner Werte abgefragt. Dabei besitzt jeder Knotenpunkt mit Ausnahme der Blätter, genau zwei ihm untergeordnete Knotenpunkte. Solch ein Konstrukt kann für eine kleine Knotenanzahl noch manuell erstellt werden, mit steigender Knotenanzahl ist es jedoch zwingend notwendig die Erstellung zu automatisieren. In der Regel werden dafür zuerst Knotenpunkte gewählt, die das Eingabefeld möglich in zwei gleichgroße Hälften teilt. Auch mit den nächsten Punkten wird so verfahren, bis schlussendlich keine sinnvolle Teilung mehr machbar ist oder die maximal vorgegebene Anzahl der Teilung erreicht worden ist.

Eine andere Möglichkeit solche Entscheidungsbäume zu erstellen, ist die Boosted Decision Tree Methode. In dieser Methode werden viele Bäume erstellt. Diese werden in mehreren Iterationen erstellt und basieren somit immer auf den Ergebnissen der vorher erstellten Bäume. Nachdem der erste Baum erstellt wurde, werden alle falsch klassifizierten Elemente höher gewichtet. Durch diese Gewichtung wird der nächste Baum diese

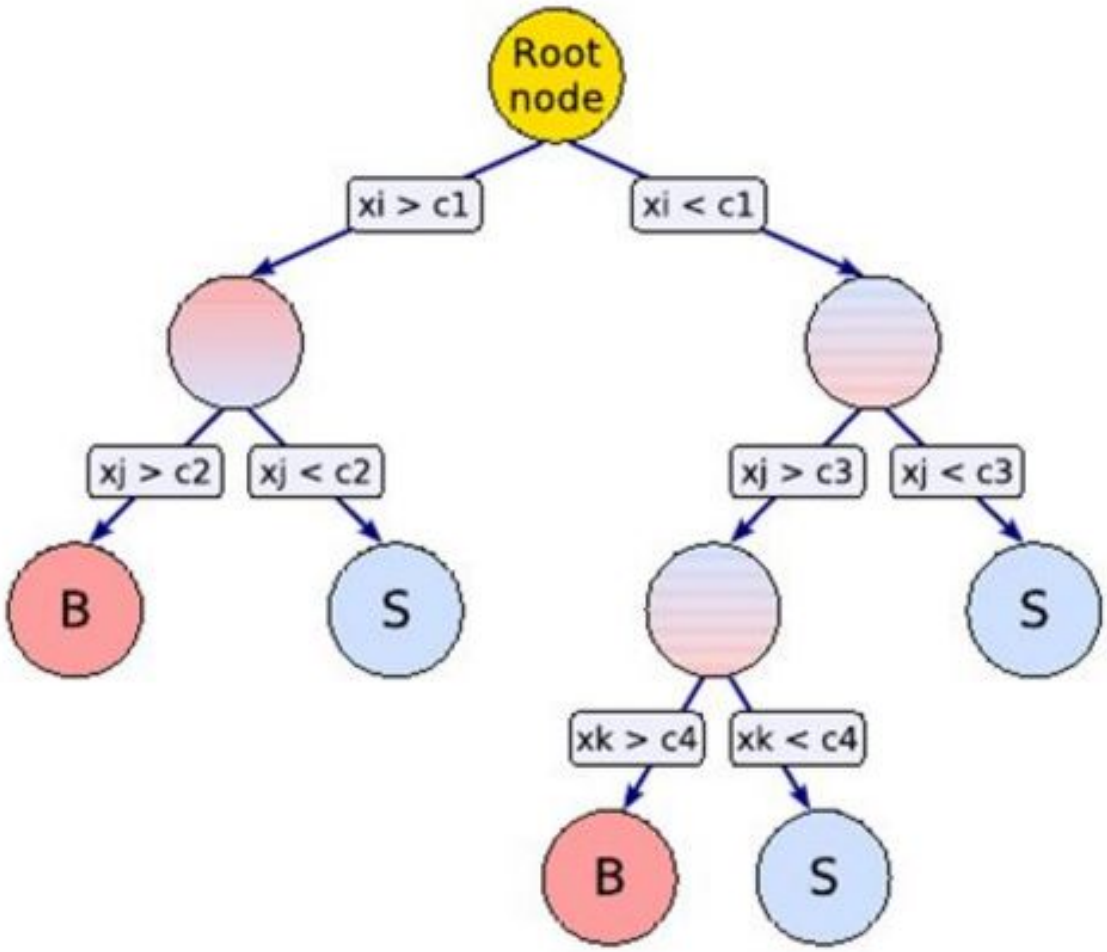


Abbildung 8.19.: Aufbau eines Entscheidungsbaumes [7]

Elemente priorisieren, damit sie in diesem Durchlauf richtig eingeordnet werden. Dieser Vorgang wird nach gegebener Anzahl der Iterationen beendet. Nach Beendigung des Trainings erhält man somit eine Vielzahl an Bäumen. Nimmt man nun diesen Klassifizierer zum Klassifizieren neuer Elemente, durchläuft jedes Element jeden Baum. Über einen Mehrheitsentscheid wird dann entschieden welcher Klasse das aktuelle Element angehört.

8.5.2. Zu Optimierende Parameter

Das Ergebnis des Boosted Decision Tree Algorithmus hängt von drei Parametern ab, die frei einstellbar sind. In diesem Kapitel werden die drei Parameter und ihre Auswirkung genauer beschrieben.

Number of Splits - Maximale Anzahl an Splits pro Baum

Gibt die Anzahl der Teilungen an die in einem Baum maximal vorkommen dürfen. In vielen Beispielen wird hier eine maximale Teilung von 1 gewählt. Dies macht jedoch nur bei Aufgabenstellungen mit zwei zu unterscheidenden Klassen Sinn. Bei mehreren Klassen, wie in unserem Beispiel, werden für ein genaueres Ergebnis mehr Splits benötigt.

Number of Learners - Anzahl generierter Bäume

Number of Learners gibt an, wie viele Bäume erstellt werden, bevor der Algorithmus beendet wird. Dabei erzielt ein höherer Wert in der Regel auch ein genaueres Ergebnis. Jedoch wird für jeden erstellten Baum Speicherplatz benötigt. Auch das spätere Klassifizieren wird durch eine höhere Anzahl an Bäumen länger benötigen.

Learning Rate - Gewichtung falsch klassifizierter Objekte

Die Learning Rate beschreibt die Veränderung der Gewichtung nachdem ein Element falsch klassifiziert wurde. Oftmals wird hier ein Wert von 0,1 verwendet. Dies verleiht einem Element eine Gewichtung von zusätzlichen 10%.

8.5.3. Optimierung in Matlab

Zur Erstellung der Klassifikatoren wurde Matlab in der Version 2016b genutzt. Die Berechnung fanden aufgrund der hohen Rechenintensität auf dem HPC der Carl-von-Ossietzky Universität Oldenburg statt. Zum Zeitpunkt dieser Projektgruppe befand sich keine aktuellere Version auf dem HPC. In der Projektgruppe wurde sich für das Antrainieren der Klassifikatoren auf vier Labelgruppen geeinigt (siehe Tabelle 8.15). Im genannten Kapitel befinden sich auch die Beschreibung und Begründung der Labelgruppen.

Zur Ermittlung der passendsten Parametereinstellungen wurden verschiedene Einstellungen antrainiert und verglichen. Der Speicherbedarf variierte bei allen Durchgängen zwischen ca. 520 MiB (Number of Splits: 50, Number of Learners: 50, Learning Rate: 0,1) und 830 MiB(Number of Splits: 100, Number of Learners: 100, Learning Rate: 0,3). Die genauen Werte des Speicherbedarfs können in der Tabelle 8.5 nachgelesen werden. Auffällig ist außerdem der relativ große Anstieg der Klassifikatorgröße mit steigender *number of learners*, also steigender Anzahl generierten Bäume. In Tabelle 8.6 werden die Genauigkeitswerte des Klassifikators für jede Parametereinstellung dargestellt. Dieser Genauigkeitswert ergibt sich durch eine 5-fache-Kreuzvalidierung, die bei jeder Erstellung berechnet wurde.

Wie in Tabelle 8.6 zu beobachten, befindet sich die Genauigkeit des Klassifikators je nach Parametereinstellung zwischen 89% und 91% (gerundet). Anhand dieser Auswertung wurden folgende Parameter für die SeniorHome-Studie verwendet:

		Number of Learner					
		50			100		
Number of Splits	50	520 MiB	666 MiB	723 MiB	712 MiB	768 MiB	789 MiB
	100	540 MiB	675 MiB	746 MiB	707 MiB	814 MiB	830 MiB
		0,1	0,2	0,3	0,1	0,2	0,3
		Learning Rate					

Tabelle 8.5.: Parametertuning - Größe - Boosted Decision Tree

		Number of Learner					
		50			100		
Number of Splits	50	89,39%	89,74%	89,87%	89,72%	90,12%	90,20%
	100	90,21%	90,40%	90,58%	90,43%	90,64%	90,73%
		0,1	0,2	0,3	0,1	0,2	0,3
		Learning Rate					

Tabelle 8.6.: Parametertuning - Genauigkeit - Boosted Decision Tree

Labelgruppe	0	1	2	3
Genauigkeit	86,71%	77,34%	77,34%	77,36%

Tabelle 8.7.: SeniorHome - Genauigkeit - Boosted Decision Tree

- Number of Split: 100, Number of Learner 50, Learning Rate 0,1

Vorteile dieser Einstellung sind zum einen die vergleichsweise hohe Genauigkeit im Gegensatz zu den anderen gezeigten Einstellungen, sowie die Dateigröße des erstellten Klassifikators. Für diese Einstellung beträgt sie ca. 540 MiB.

Die in der Tabelle 8.15 beschriebenen Labelgruppen wurden mit der gezeigten Parametereinstellung antrainiert. Die Ergebnisse sind in der Tabelle 8.7 dargestellt.

8.5.4. Interpretation der Ergebnisse

Zur Interpretation der Daten betrachten wir hier die Konfusionsmatrix (Abbildung 8.20 des Datensatzes SHID11437. Auf der linken Seite stehen die Label, die die Projektgruppe nach der Datenaufnahme überarbeitet hat und somit als Standard gelten. Im oberen Bereich sind die jeweils vom Boosted Decision Tree annotierten Label zu sehen. Besonders auffällig ist zunächst der Bereich STAND. Circa 56% des gesamten Datensatzes wurde als Stehen annotiert.

Besonders schwer fällt dem Boosted Decision Tree Algorithmus die Unterscheidung zwischen Sitzen und Stehen. In den Daten ist erkennbar, dass der Klassifikator an kleinen Änderungen der Accelerometer Z-Achse bestimmt, ob es sich um Stehen oder Sitzen handelt (siehe Abbildung 8.21).

Außerdem kann man aus der Tabelle ablesen, dass Transitionen vom Boosted Decision Tree Klassifikator nicht annotiert werden. Im Übergang zwischen Sitzen und Stehen wird keine Transition gelabelt. Vielmehr wird nach überschreiten eines Schwellwertes

	SIT	STAND	SIT_STAND	STAND_SIT	BEND_OVER_...	STAIR_CLIMB	STAIR_CLIM...	WALK
SIT	82054	146218	0	0	0	0	0	300
STAND	1500	84979	0	0	193	0	0	100
SIT_STAND	271	263	0	0	0	0	0	300
STAND_SIT	328	1080	0	0	50	0	0	186
BEND_OVER_...	0	9500	0	0	1950	0	0	313
STAIR_CLIMB	0	0	0	0	0	0	0	615
STAIR_CLIMB_...	0	17	0	0	0	0	0	445
WALK	0	3013	0	0	0	0	0	12845

Abbildung 8.20.: Konfusionmatrix SHID11437 - Boosted Decision Tree

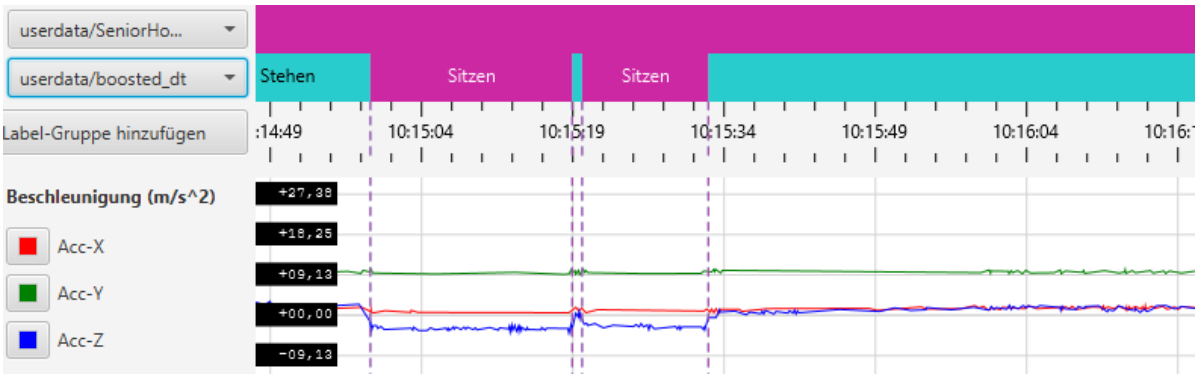


Abbildung 8.21.: MAMKS Screenshot - Sitzen / Stehen - SHID11437

zwischen Sitzen und Stehen gewechselt. Dieses Verhalten lässt sich durch die punktuelle Betrachtung eines Impulses begründen. Eine Fensterbetrachtung in gewisser Größe würde diese Abschnitte eventuell besser annotieren.

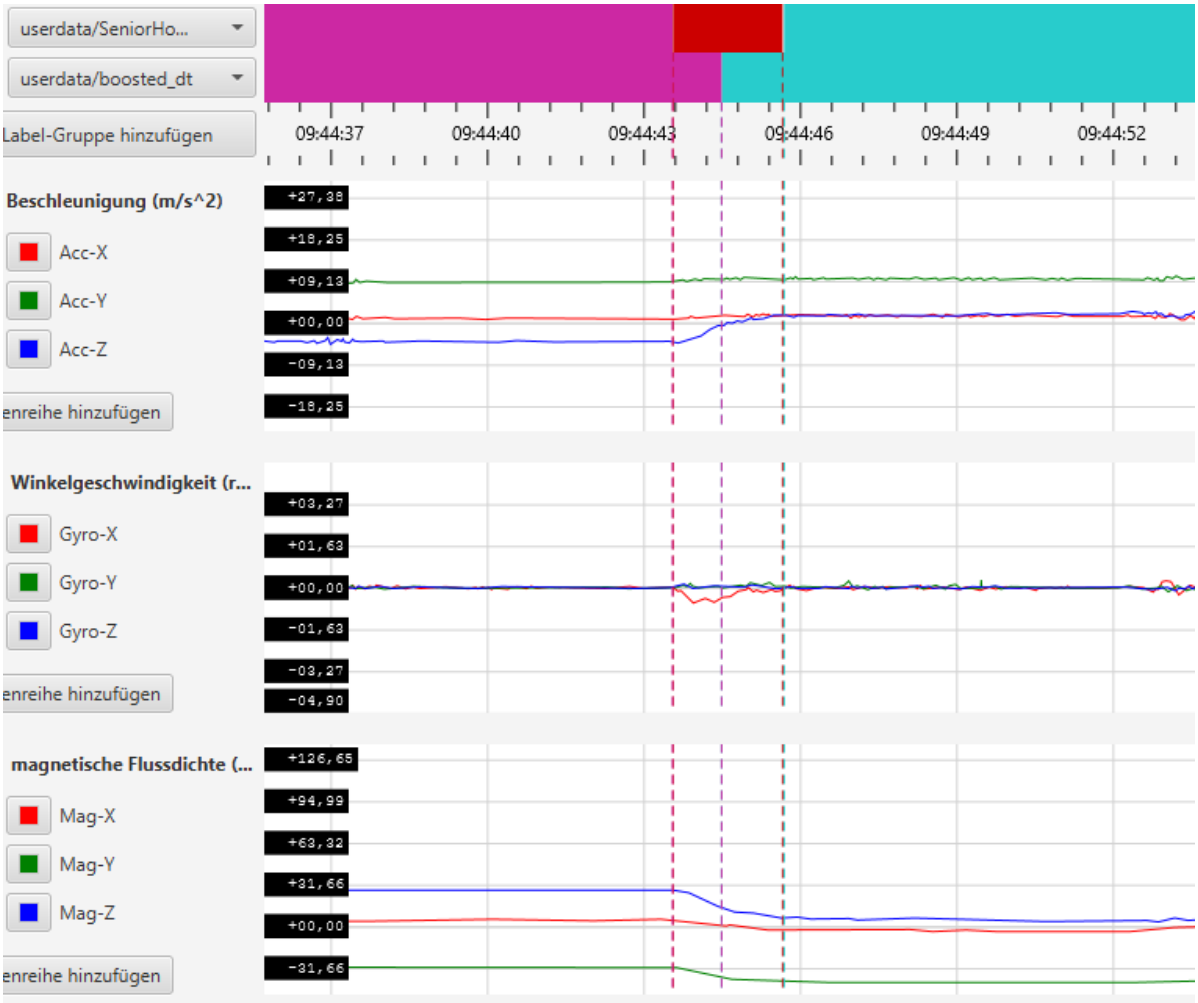


Abbildung 8.22.: MAMKS Screenshot - Aufstehen / Hinsetzen - SHID11437

Neben den Transitionen fällt in der Abbildung 8.20 auf, dass das Treppensteigen nie als Treppensteigen erkannt wurde. Die Richtung spielt hierbei keine Rolle. In nahezu allen Fällen wird Treppensteigen als Laufen erkannt (siehe Abbildung 8.23). Der leicht höhere Impact, der beim Treppensteigen entsteht, wird nicht als solcher erkannt. In den Trainingsdaten ist Treppensteigen im Verhältnis zu Gehen um ein vielfaches geringer.

Dadurch lässt sich die Fehlerkennung begründen.

8.5.5. Fazit

Der Boosted Decision Tree Klassifikator, wie er in der Projektgruppe trainiert wurde, eignet sich am besten für die Erkennung von Gehen. Im Schnitt werden ca. 76% der annotierten Gehen Daten auch als Gehen erkannt. Auch Stehen wird sehr gut erkannt. Bis zu 98% Erkennungsrate wird bei Stehen erreicht. Wie bereits im vorherigen Abschnitt erläutert, erkennt der Algorithmus besonders häufig Sitzen nicht als solches. Oftmals wird hierbei Stehen annotiert. Fasst man Stehen, Sitzen und andere statische Aktivitäten jedoch als statische Aktivitäten zusammen und vergleicht sie dann mit dynamischen Aktivitäten, kann man eine gute Trennung dieser beiden Kategorien feststellen. Möchte man also nur eine Aussage über das Verhältnis von Bewegungen zu Sitzen oder Stehen treffen, eignet sich die Klassifizierung des Boosted Decision Trees. Eine genau Klassifizierung aller Aktivitäten ist durch die geringe Genauigkeit nicht ratsam. In diesem Projektbericht werden dazu andere Klassifikatoren vorgestellt, welche einzelne Aktivitäten besser erkennen können.

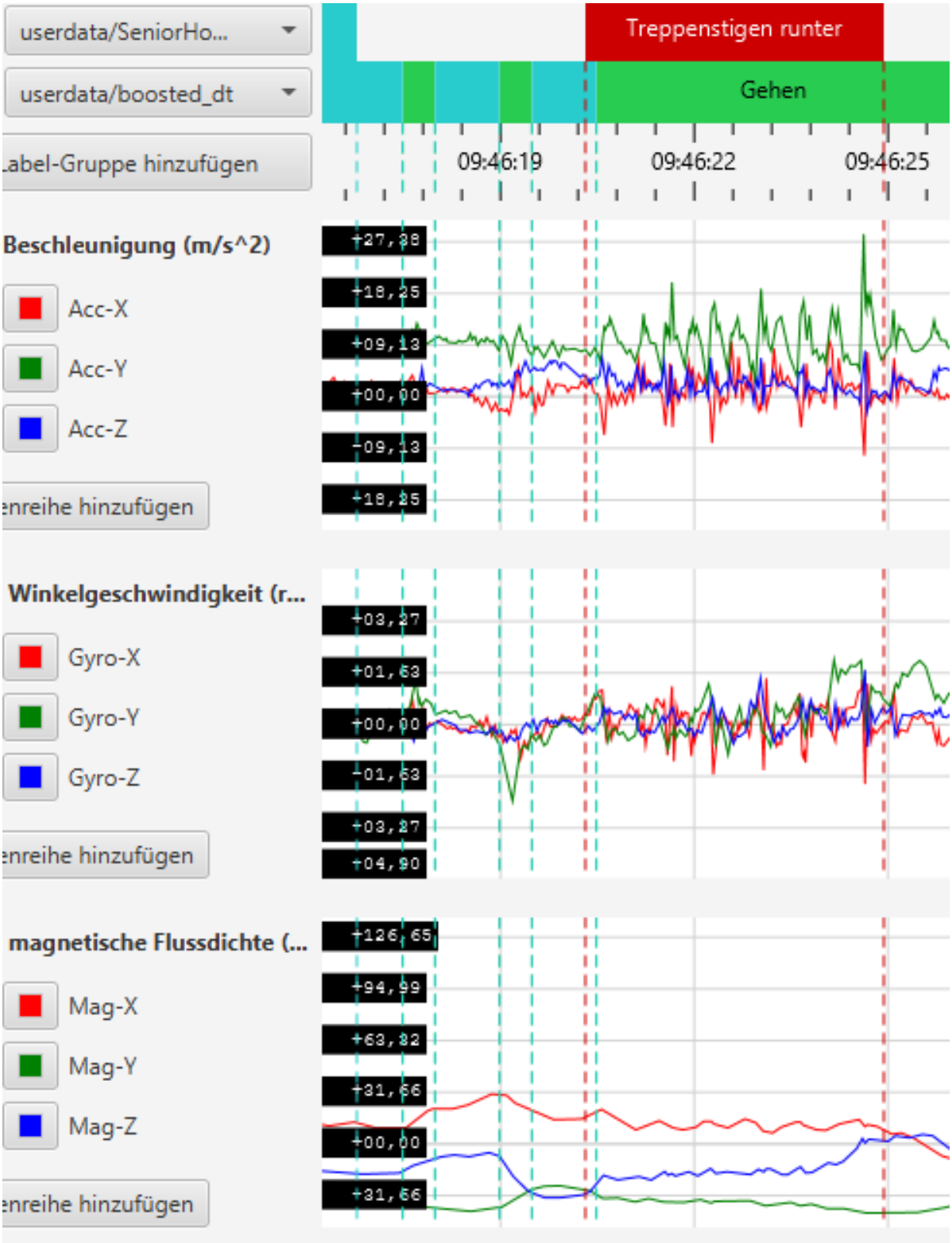


Abbildung 8.23.: MAMKS Screenshot - Treppe steigen - SHID11437

8.6. K-Nearest Neighbor

Um möglichst gute Klassifizierungsergebnisse zu erlangen wurden mehrere Algorithmen auf ihre Optimierung geprüft. Hierzu zählt unter anderem der nachfolgende beschriebene K-Nearest Neighbor-Algorithmus (kurz KNN). Bereits in der vorangegangenen Projektgruppe wurde die Funktionsweise kurz angeschnitten und der Algorithmus als sinnvoll zur weiteren Verwendung erachtet. Da dieser allerdings nicht umgesetzt wurde, wird er im Rahmen dieser Projektgruppe mit in die Ergebnisfindung eingebunden. Ein weiterer Grund, weswegen sich dafür entschieden wurde den KNN zu berücksichtigen, ist nachfolgendes Bild. Es wurde von den Erstellern der Open-Source-Bibliothek scikit-learn erstellt und dient als erste Auswahlmöglichkeit anhand der gegebenen Daten.

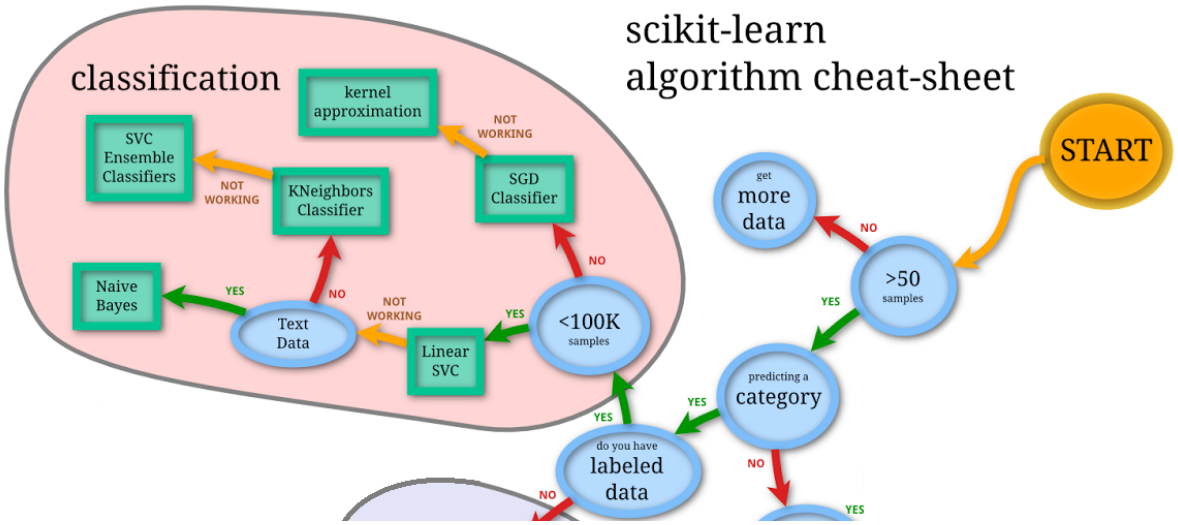


Abbildung 8.24.: Machine-Learning-Cheat-Sheet [2]

1. Beginnend vom Startpunkt besitzen wir mehr als 50 Einträge in unseren Daten.

2. Außerdem möchten wir die Daten in Kategorien einordnen (Sitzen, Stehen, Aufstehen etc.)

3. Da wir unsere Daten selber vorgelabelt haben, benutzen wir keine Clustering-Algorithmen, die selbst einzelne Gruppen deuten, sondern einen Klassifikationsalgorithmus.

4. Unsere Daten entsprechen auf jeden Fall mehr als 100000 Datenpunkte.

5. Entsprechend der Grafik ist ein linearer Support-Vector-Classifier zu wählen. Da die vorige Projektgruppe diesen bereits in Form einer Support Vector Maschine angewendet hat, springen wir einen Punkt weiter.

6. Bei unseren Daten handelt es sich nicht um Text-Daten, sondern um Zahlenpunkte - der Entscheidungsstrang bringt uns zu den K-Neighbors-Classifier-Algorithmen.

Eine weitere Erkenntniss die aus der Grafik entnommen werden kann ist, dass der KNN bereits ab 100000 Datenpunkten aussagekräftige Ergebnisse liefern kann. Entsprechend kann beim ersten Testen auf eine reduzierte Datenmenge zurückgegriffen werden, damit ein Anlernen nicht eine übermäßig hohe Zeit in Anspruch nimmt. Da der Sensorgürtel mit 100Hz die Sekunde Daten erfasst, werden die 100000 Datenpunkten bereits nach rund 17 Minuten generiert (100000 Samples / 100 Hz / 60 Sekunden = 16,666..). Des Weiteren werden von scikit-learn lediglich Algorithmen implementiert, die sich in der Vergangenheit bewährt haben und gut erforscht sind. Der Alogrithmus muss sich mindestens über die letzten drei Jahre seit seiner Publikation durchgesetzt haben und in mehr als 200 wissenschaftlichen Ausarbeitungen zitiert bzw. bearbeitet worden sein [16]. Somit wird sichergestellt dass der verwendete Algorithmus gut genug ergründet wurde, um produktiv eingesetzt zu werden.

8.6.1. Zu Optimierende Parameter

Wie bereits in der Seminararbeit zu KNN angesprochen, sind die anzupassenden Parameter des Algorithmus sehr begrenzt. Zum einen besteht die Möglichkeit die zu betrachtenen Nachbarn anzupassen. Sprich es wird ausgehend vom Datenpunkt nach

'N'-beliebigen Nachbarn gesucht. Durch einen Mehrheitsentscheid wird anschließend bestimmt, um was für einen Datenpunkt es sich bei dem Unbekannten Wert handelt.
Des Weiteren besteht die Möglichkeit die Distanzmetriken anzupassen. Das bedeutet, dass die zu bestimmenden Distanz zu den Nachbarn unterschiedlich berechnet werden kann. Dies ermöglicht es den Algorithmus möglichst ideal auf seine Daten zuzuschneiden. Somit können beispielsweise selbst Daten mit unterschiedlichen Typen berücksichtigt werden (beispielsweise können so Strings auf ihre Distanz verglichen werden). Eine genauere Betrachtung einzelner Distanzmetriken finden sich in der zugehörigen Seminararbeit.

Grundlagen Testdaten

Um einen ersten Überblick zu bekommen, wurde die nachfolgende Matrix erstellt. Sie soll helfen eine oberflächliche Einschätzung zu gewinnen. Hierbei sollen vor allem erste Annahmen getroffen werden, um sie im Nachhinein überprüfen zu können.

	precision	recall	f1-score	support
1	0.75	0.48	0.58	1580
2	0.98	0.99	0.98	104146
3	0.87	0.74	0.80	4414
4	0.76	0.25	0.37	1247
5	0.96	0.98	0.97	74871
6	0.87	0.79	0.83	4806
7	0.96	0.97	0.96	61408
avg / total	0.96	0.97	0.96	252472

Abbildung 8.25.: Beispiel F1 Score

Bei der Matrix handelt es sich um eine F1-score Tabelle. In der ersten Spalte befinden sich die verschiedenen Label (1-7).

1. Springen
2. Sitzen

3. Transition sitzen-stehen (aufstehen)
4. Treppensteigen
5. Stehen
6. Transition stehen-sitzen (hinsetzen)
7. Laufen

Des Weiteren sind einige Grundbegriffe nötig, um darauf aufbauende Begriffe zu verstehen. Nachfolgend eine kurze Erklärung der verwendeten Begriffe und eine kurze Erklärung:

1. TP = true positives, richtig als richtig erkannt
2. FN = false negatives, falsch als falsch erkannt
3. FP = false positives, falsch als richtig erkannt
4. TN = true negatives, richtig als falsch erkannt

Aus diesen Grundwerten lassen sich folgende Berechnungen für die Einzelwerte in der Konfusionsmatrix vornehmen.

$$precision = \frac{TP}{(TP + FP)} \tag{8.4}$$

$$recall = \frac{TP}{(TP + FN)} \tag{8.5}$$

$$f1score = \frac{2TP}{(2TP + FP + FN)} \tag{8.6}$$

Bei dem ‘support’ handelt es sich um die im Datensatz verfügbaren Sample der einzelnen

Aktivität. In der obigen Abbildung sind beispielsweise 1580 Datenpunkte mit dem Label springen vorhanden [12].

Zum Verständnis ein anschauliches Beispiel der zuvor definierten Begriffe.
In einem Bild sind 10 Pelikane und 6 Nilpferde zu sehen. Der Algorithmus erkennt 7 Pelikane. Davon sind 6 in Wahrheit wirklich Pelikane. Die 6/7 sind in diesem Fall die true positive Werte. Das fälschlicherweise als Pelikan erkannte Nilpferd (1/7) gehört zu den false positive Werten.
In diesem Beispiel ist also die precision 6/7. Sprich die vom Algorithmus erzielte precision liegt bei rund 85%.
Der recall ist in diesem Fall die richtig gedeuteten Klassifikationen im Verhältnis zu allen vorhandenen Pelikanen. Sprich 6/10 bzw. 60%.

Interpretation Testdaten

Bereits bei erster Betrachtung lässt sich feststellen, dass die Label mit wenig Support(vorhandene Daten) einen deutlich schlechteren f1-score liefern. Besonders die Aktivität Treppensteigen wird mit einem sehr schlechten Recall von 25% erkannt. Dies bedeutet, dass aus allen Treppensteigen Daten nur etwa ein viertel gefunden wurden. Von den gefundenen Daten wurde außerdem eine vergleichsweise niedrige precision erzielt. Eine Möglichkeit, weshalb die Aktivität Treppensteigen schlecht klassifiert wurde, könnte darauf hindeuten dass sie der Aktivität Laufen zu sehr ähnelt.
Des Weiteren besitzt die Aktivität Springen einen niedrigen f1-score von 58%. Es werden insgesamt nur knapp die hälfte aller Springen-Daten erkannt. Hiervon werden immerhin 75% richtig zugeordnet. Man würde annehmen das es sich beim Springen um eine sehr signifikante Bewegung handelt und einen hohen recall erwarten. Allerdings könnte es sein, dass die Bewegung mit der Transition aufstehen verwechselt wird, da es sich bei beiden Aktivitäten um einen großen Ansteig an der y-Achse handelt.
Bei allen Aktivitäten sieht man, umso mehr Support vorhanden sind, umso besser ist

auch der f1-score. Vor allem die drei am meisten vertretenen Aktivitäten Sitzen, Gehen und Stehen haben sehr gute Erkennungs und Klassifizierungsraten. Das Gesamtergebnis der Tabelle (avg / total) ist hinsichtlich der Häufigkeit gewichtet. Da die schlechten Ergebnisse wenig Support haben, entsteht trotzdem ein gutes Durchschnittsergebnis.

8.6.2. Optimierungen in Matlab

Zu Beginn der Projektgruppe wurde nach Möglichkeiten gesucht, entsprechende Modelle zum Klassifizieren anzulernen. Hierbei wurden Matlab und Python als mögliche Plattformen für eine Umsetzung identifiziert.
Anschließend wurde begonnen in Matlab die vorhandenen Bewegungsdaten mit Hilfe von KNN zu klassifizieren und die Parameter zu optimieren. Hierbei wurden diverse Abstandsmetriken und Nachbarn getestet. Der verwendete Datensatz besteht aus den Bewegungsdaten der vorherigen Projektgruppe. Er beinhaltet 18 verschiedene Aktivitäten die klassifiziert werden sollen.

Im ersten Optimierungsschritt wurde überprüft, welche Anzahl an Nachbarn die besten Ergebnisse liefert. Hierzu wurden verschiedene Nachbarn als Größe festgelegt und jeweils eine Konfusionsmatrix mit den Erkennungsraten generiert. Bei der Nachbarbetrachtung wurden Nachbargrößen von 1 - 100 betrachtet. Zusammenfassend wurde folgende Grafik erstellt.

Es zeigt sich eine deutlicher Abwärtstrend (gepunktet), umso höher die Anzahl der zu betrachtenden Nachbarn gewählt wird. Bei der Wahl K=1 und K=3 ist das Ergebnis identisch - 96.0%. Es kann allerdings vermutet werden das es sich bei der Betrachtung von einer so geringen Menge an Nachbarn um ein Overfitting handelt (siehe Seminararbeit). Auf Grund des kontinuierlichen Abwärtstrends wurden nicht mehr als 100 Nachbarn zum Vergleich in Betracht gezogen. Um einen repräsentativen Zwischenwert zu erhalten, wurde anschließend die Konfusionsmatrix von einer Nachbarbetrachtung von K=13 generiert.

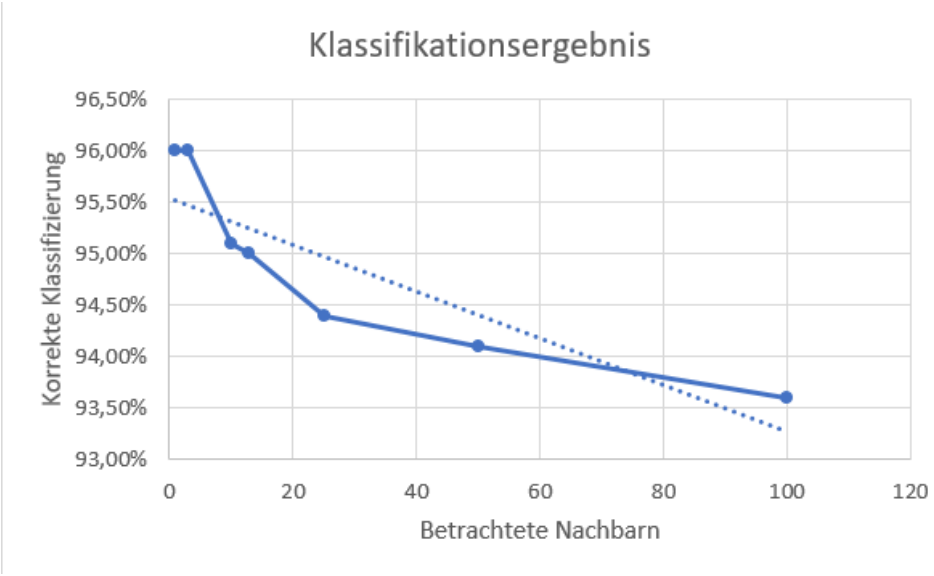


Abbildung 8.26.: KNN - Nachbarbetrachtung

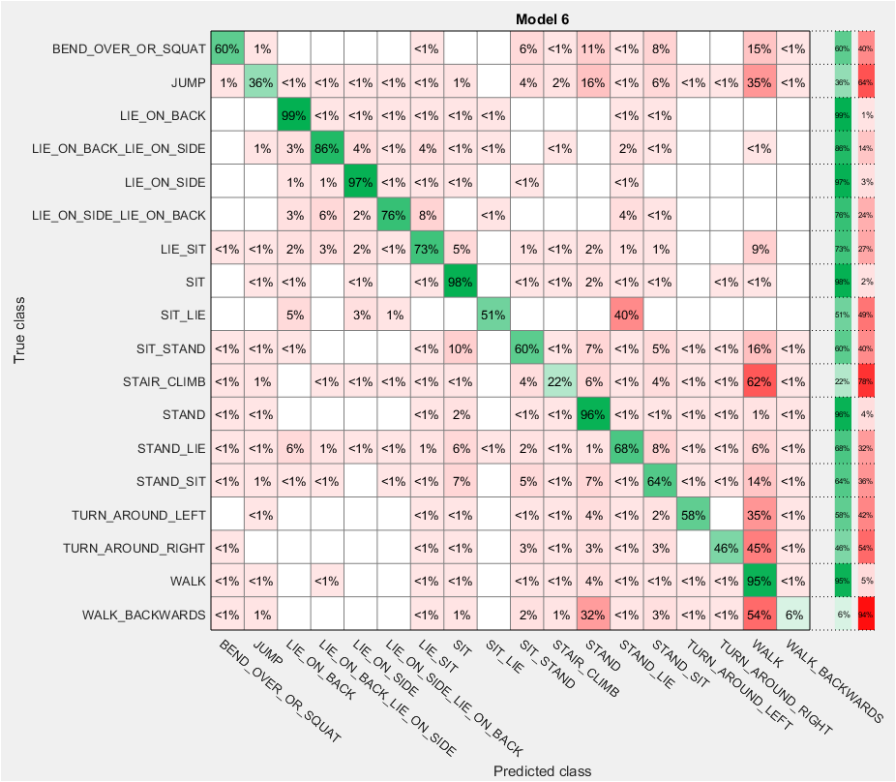


Abbildung 8.27.: 13NN - Konfusionsmatrix

Wie der Konfusionsmatrix zu entnehmen, wird, wie bereits angenommen, das Treppensteigen sehr oft mit dem normalen Gehen verwechselt. Außerdem wird die Aktivität Springen offenbar häufig mit Gehen und auch Stehen verwechselt. Eventuell, weil sich die Person beim Beginn des Springens in der Standposition befindet. Das gehen könnte interpretiert werden, wenn der Impact beim wieder landen auf dem Boden eventuell nicht sehr stark ist und somit einem normalen Gang-Auftritt entspricht. Besondere Schwierigkeiten sind beim Rückwärtsgehen zu erkennen. Das normale Gehen und Rückwärtsgehen ist zu identisch um gut voneinander differenziert werden zu können. Generell ist, wie bereits im ersten Testversuch festgestellt, zu sehen, dass Aktivitäten mit einem hohen Support eine sehr viel bessere Erkennungsrate besitzen. Man kann also darauf schließen das sich bei einem höheren Support gewisse Cluster bilden (visuell gesprochen). Diese hohe Verdichtung begünstigt eine gute Nachbarbetrachtung.

Außerdem wurde die Minkowski-Distanz als Distanzmetrik betrachtet. Da diese aber bereits bei einer Nachbarbetrachtung von 10NN eine Erkennungsrate von unter 95% besaß, wurde die Verwendung nicht weiter in Betracht gezogen. Im Rahmen der Seminararbeit wurden außerdem die Manhattan-Distanz und Hamming-Distanz untersucht. Resultierend aus den Untersuchungen ergab sich, dass die Distanzen nicht für unsere vorliegenden Daten geeignet sind.

Im nächsten Schritt wurde die inverse Distanzgewichtung untersucht. Mit den zuvor verwendeten Daten wurden entsprechend neue Modelle angelernt und die Ergebnisse miteinander verglichen.

Wie in dem Graphen zu erkennen, liefert die inverse Distanzgewichtung bessere Ergebnisse. Auch bei erhöhter Nachbarbetrachtung nimmt die Erkennungsgenauigkeit geringfügiger ab, als bei der Betrachtung ohne Distanzgewichtung. Bei einer Betrachtung von einem Nachbarn K=1 ist die Erkennungsgenauigkeit bei 96,78%. Bei einer Betrachtung von 50 Nachbarn ist diese noch bei 95,0%. Damit ein direkter Vergleich gezogen werden kann, wurde erneut eine Konfusionsmatrix mit 13 Nachbarn erzeugt.

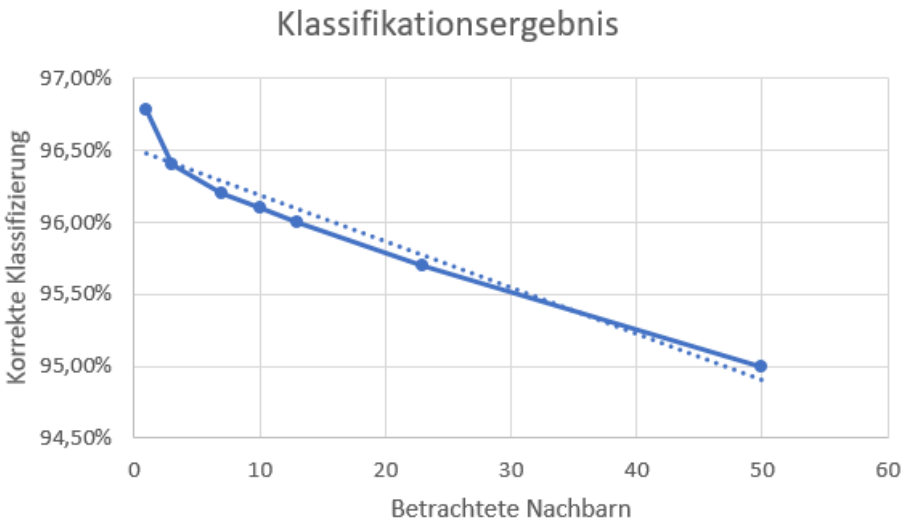


Abbildung 8.28.: WKNN - Nachbarbetrachtung

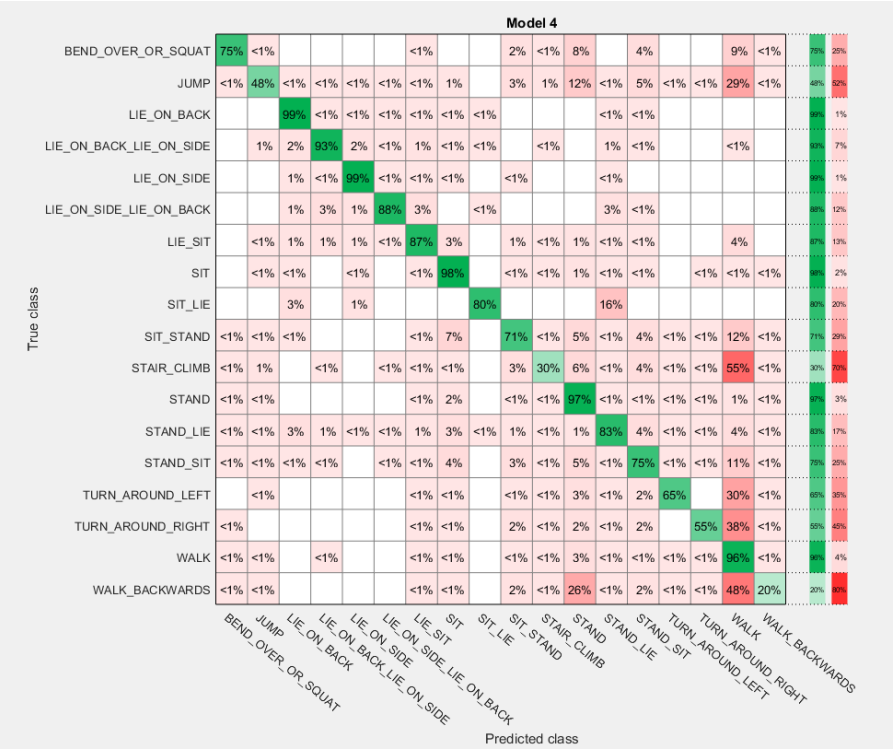


Abbildung 8.29.: W13NN - Nachbarbetrachtung

Im Vergleich zu der Konfusionsmatrix ist zu sehen, dass jedes der Ergebnisse besser erkannt wurde. Allerdings existieren dennoch die gleichen Probleme bei der Erkennung wie bereits vorher. Hierbei sind nach wie vor das Treppensteigen und das Rückwärtsgehen am Stärksten betroffen.

Auf Grund des hohen Zeitaufwandes, den Matlab bei der Klassifikation aufbringen muss, wurde sich im Laufe des Projekts dazu entschieden auf ein Anlernen des Algorithmus mit Python umzusteigen. Hierbei wurden die zuvor gesammelten Erkenntnisse adaptiert.

8.6.3. Umsetzung in Python

Zuerst wurden die zuvor erstellten Ergebnisse gegengetestet und konnten bestätigt werden. Der Weighted-KNN schneidet immer besser ab als ein KNN ohne Gewichtung. Des Weiteren fällt auf, dass die Modellgröße sich signifikant verkleinert hat und die Ausführungszeit, um ein Modell anzutrainieren, deutlich schneller ausfällt. Für das Training wird das Paket 'NearestNeighbors' von scikit-learn verwendet.

Labelgenerierung

Da der Algorithmus jeden einzelnen Punkt betrachtet und anschließend klassifiziert, würde er auch für jeden einzelnen Datenpunkt ein Label generieren. Bereits bei einer Erkennungswahrscheinlichkeit von 90 Prozent sind 10 Datenpunkte in einer Betrachtungsreihe von 100 Datenpunkten (1 Sekunde) mit unterschiedlichen Labeln versehen. Da dies schnell unübersichtlich werden kann, wurde sich dazu entschieden eine entsprechende Glättung der Label vorzunehmen. Hierbei wird auf einen Mehrheitsentscheid zurückgegriffen. Es wird ein Zeitraum festgelegt über den der Mehrheitsentscheid vorgenommen wird. Dies erleichtert eine bessere Ansicht in MAMKS und hat besonders Einfluss auf die Anzeigeperformance.

Um einen visuellen Vergleich der Daten zu haben und auch Ergebnisse zu erzielen, wurde

im Laufe der Optimierung eine Anwendung geschrieben, die für eine MAMKS-Umgebung automatisiert Label generieren kann. Hierzu muss in der Anwendung das entsprechende Scikit-Modell und der Pfad für die zu labelnen Daten angegeben werden. Der Vorteil ist ein einfaches Labeln von mehreren Daten, mit dynamischen Modellen. Dies ermöglicht ein Testen des angelernten Modells ohne großen Aufwand und anschließender Vergleich der Labeldaten in MAMKS.
Sobald die Anwendung gestartet wurde, ist es möglich das entsprechende scikit-learn Modell auszuwählen. Anschließend kann der Workspace gewählt werden. Hierbei ist darauf zu achten das der Workspace kompatibel ist. Dies bedeutet, dass die Unterordner mit den einzelnen Probandenordnern jeweils einen Ordner 'Sensordata' und 'Userdata' enthalten. In dem Ordner 'Sensordata' liegen die entsprechenden Bewegungsdaten in der MAMKS-Form (float32). Der Ordner 'Userdata' wird benötigt um die generierten Label abzuspeichern. Nach dem wählen des Workspace wird eine kurze Konsistenzprüfung durchgeführt, ob der gewählte Workspace kompatibel ist.

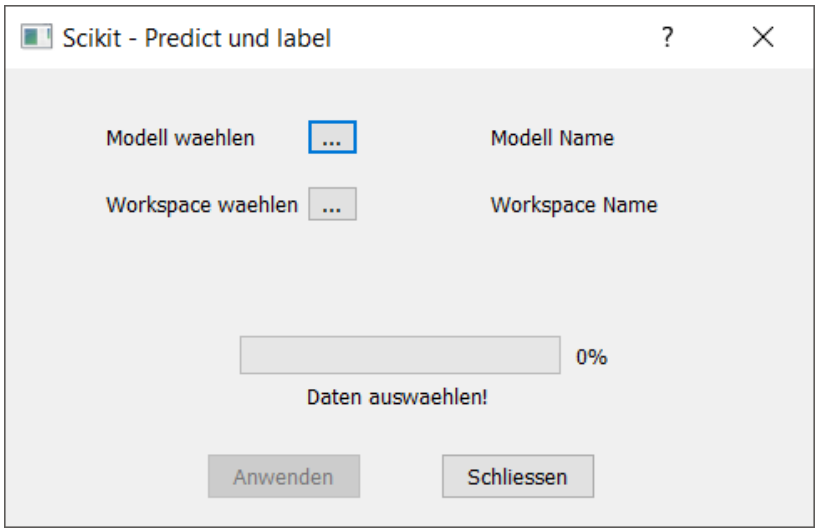


Abbildung 8.30.: Anwendung zum Label-Generieren

Falls das Modell gewählt und der Workspace kompatibel ist, lassen sich die Label generieren. Dies ermöglicht eine einfache Auswahl von mehreren Probandenordnern und eine bequeme Handhabung. Nach dem Starten wird endsprechend über einen Fortschritts-

balken angezeigt, wie weit der Status der Anwendung ist.

Für jeden Probandenordner wird entsprechend eine MAMKS-konforme Labeldatei erstellt und im dafür vorgesehenen Ordner gespeichert. Die Labeldatei orientiert sich an dem Namen des importierten Modells, damit eine leichte Zuordnung möglich ist. Falls das importierte Modell beispielsweise 'knn_7.sav' heißt, wird für jeden Probanden ein Label mit dem Namen 'label-knn_7.xml' generiert. In MAMKS können die Label anschließend in den Datensätzen angezeigt werden lassen. Beim Import der Modelle ist darauf zu achten, dass die Namenskonventionen eingehalten werden. Ein '=' im Namen ist beispielsweise nicht erlaubt, da Python dieses als Zuweisungsoperator deuten würde.

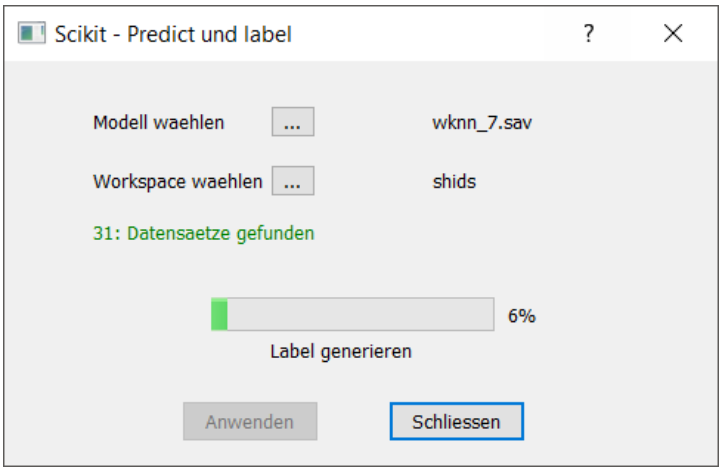


Abbildung 8.31.: Durchführung der Labelgenerierung 1

Die Anwendung ist derzeit so konfiguriert, dass sie eine Labelnachbereitung mit einer Labelbreite von 100Hz durchführt. Dies hat, wie bereits angesprochen, bisher eine hohe Genauigkeit erzielt. Die Labelbreite kann jederzeit durch die Änderung des Parameters 'window_size' im Code angepasst werden. Des Weiteren sind derzeit nur die Quadratische Diskriminanzanalyse und der K-Nearest-Neighbor -Algorithmus implementiert. Falls der Import von anderen Modellen von scikit-learn gewünscht ist, kann dies durch einen kurzen Import am Anfang des Codes gelöst werden. Beispielsweise könnte die Unterstützung von Support-Vector-Maschines mit der Zeile 'from sklearn import svm' hinzugefügt werden.

Auswahl Mehrheitsentscheid

Der Mehrheitsentscheid kann mit beliebigen Zeitwerten getetestet werden. Das nachfolgende Beispiel zeigt einen kurzen Vergleich der unterschiedlichen Intervalle.

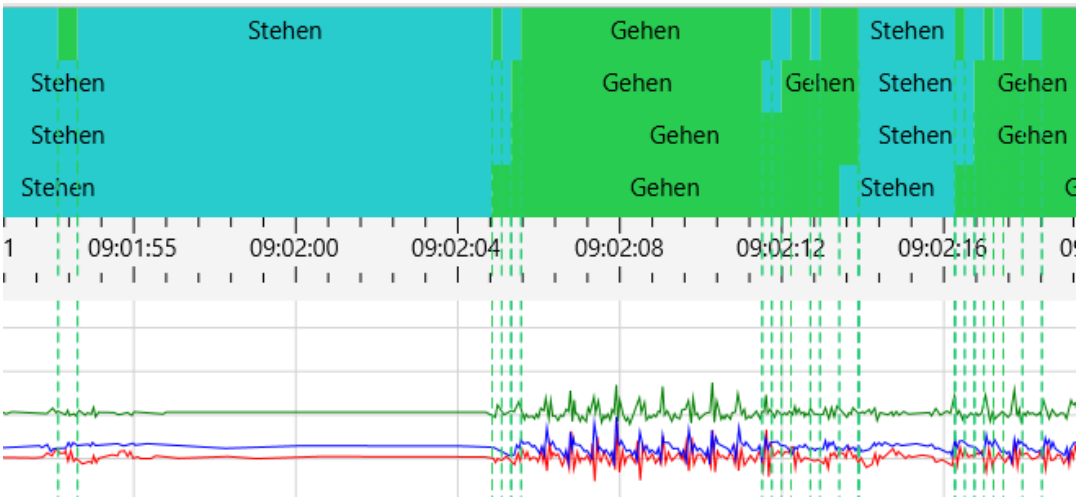


Abbildung 8.32.: Durchführung der Labelgenerierung 2

Im Bild sieht man vier verschiedene Labelgruppen. Alle wurden mit dem gleichen Modell klassifiziert. Die Gruppen zeigen absteigend folgende Zeitintervalle - 25Hz, 50Hz, 100Hz und 150Hz. Bei 25Hz sieht man deutlich noch sehr viele Störsignale in den Daten. Es werden Label generiert, wo sie oftmals keinen Sinn ergeben. Bei 50Hz sind die Label bereits etwas klarer zu sehen. Allerdings sind ebenfalls noch einige Störsignale vorhanden. Mit der Fenstergröße von 100Hz konnten erfahrungsgemäß die besten Label generiert werden. Sie haben eine recht gute Abgrenzung der Aktivitäten und zeigen sich überwiegend Störfrei. Sobald die Fenstergröße in den Bereich von 150Hz kommt, sind die Aktivitäten ebenfalls klar getrennt, allerdings sind die Anfangs- und Endpunkte oftmals etwas verschoben und wirken nicht ganz exakt.

In diesem Beispiel sind die gleichen Fenstergruppen wie bereits zuvor gewählt worden. Es zeigt sich deutlich, dass die 150Hz-Label die Transition Aufstehen mit einem Gehen überschrieben haben. Umso größer die Fenstergröße über die iteriert wird, umso schwieriger ist die genaue Erfassung von sehr kurzen Aktivitäten. Mit der Fenstergröße

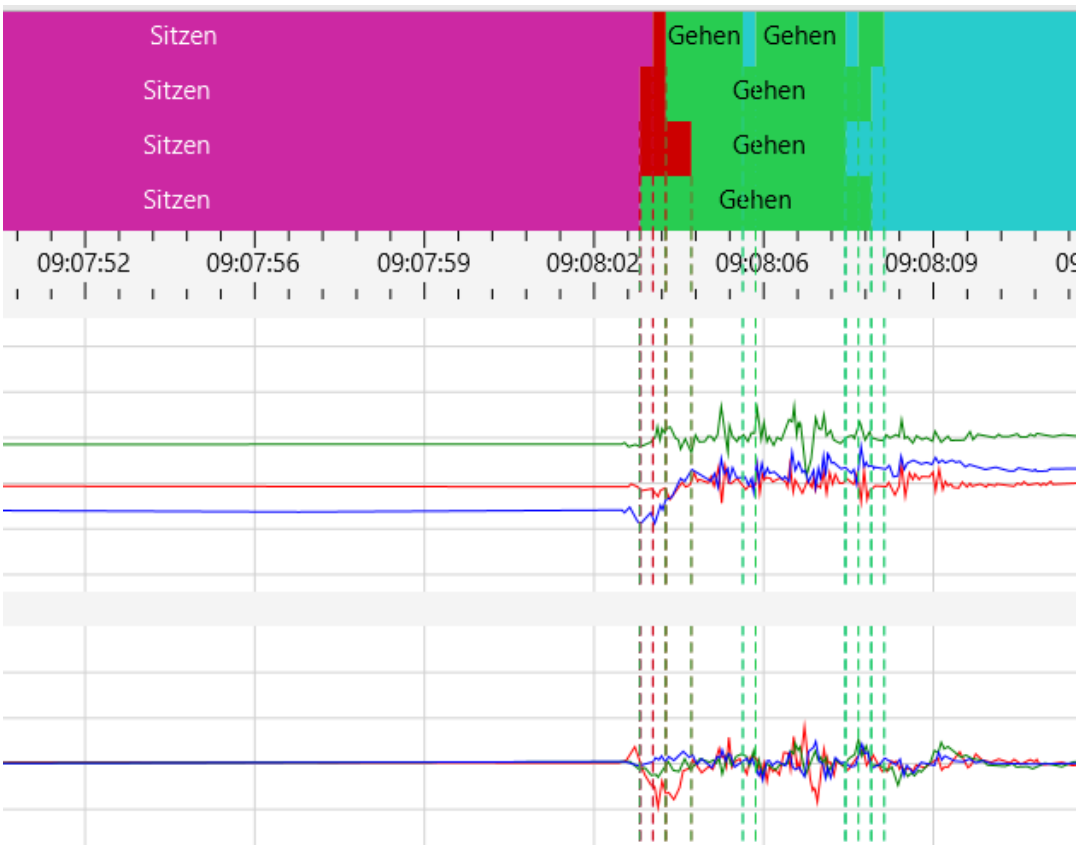


Abbildung 8.33.: Durchführung der Labelgenerierung 3

von 100Hz können diese Aktivitäten allerdings noch gewährleistet werden. Hierbei ist zu beachten, dass bei 100Hz mindestens 51Hz der Aktivität entsprechen (bei lediglich 2 Aktivitäten), um den Mehrheitsentscheid zu gewinnen. Es ist also möglich im Bestfall Aktivitäten ab einer halben Sekunde Durchführung zuverlässig zu labeln. Dies trägt deutlich zu einer Beseitung von gestreuten Daten bei. Sprich, einzelne Datenpunkte die Fehlerhaft erkannt wurden und nur einen sehr kleinen Bruchteil der Label beinhalten, werden dadurch ignoriert und die Label werden geglättet.

Klassifizierungsprogramm

Um eine einheitliche Schnittstelle für nachfolgende Arbeiten zu gewährleisten, wurde für die in Python angefertigten Modelle (Umfassend der Tensorflow und Scikit-Modelle) ein command-line-interface für MAMKS geschrieben. Dies kann aus MAMKS aufgerufen werden um Label mit Hilfe eines Modells zu generieren. Weiteres dazu findet sich im dokumentierten Handbuch. [HIER NOCH EIN VERWEIS AUF DAS HANDBUCH, WENN ICH WEIß WO ES IN DER DOKU IST]. Es ersetzt somit die proprietäre Testanwendung und ist außerdem etwas performanter. Zusätzlich ist es so konzipiert, dass es beliebig modular erweitert werden kann.

8.6.4. Deutung der Ergebnisse

Anhand der angefallenen Ergebnisdaten lässt sich feststellen, dass eine relativ hohe Genauigkeit mit dem KNN erzielt werden kann. Durch ausführliche Tests hat sich gezeigt dass ein gewichteter KNN auf den Bewegungsdaten immer eine höhere Wahrscheinlichkeit erzielt. Da es sich bei unseren Daten nicht um immer exakt gleiche Datenpunkte handelt, kann davon ausgegangen werden, dass eine Nachbarbetrachtung von $K = 1$ ein Overfitting darstellen würde. Der Algorithmus wäre entsprechend zu sehr an den Trainingsdatensatz angepasst. Wie in 8.28 zu sehen, ist der Abfall der Genauigkeit besonders stark bei einer Nachbarbetrachtung von 1 und 3. Anschließend leidet die Genauigkeit

relativ konstant, umso mehr Nachbarn gewählt wurden. Es ist also davon auszugehen das es sich bei einer Betrachtung von sehr wenigen Nachbarn deutlich um ein Overfitting handelt. Aus diesem Grund wurde für das finale Modell eine Nachbaranzahl von 7 festgelegt. Sie liefert eine hohe Erkennungsgenauigkeit und es kann davon ausgegangen werden, dass es sich um keine Anpassung an den Datensatz handelt.
Des Weiteren kann beobachtet werden, dass Aktivitäten, von denen mehr Daten vorhanden waren, zuverlässiger erkannt werden. Aus diesem Grund ist besonders die Erkennung von Gehen, Stehen und Sitzen mit sehr hohen Wahrscheinlichkeiten vertreten. Probleme treten allerdings bei besonders kurzen Aktivitäten auf. Beispielsweise die Transitionen wie das Hinsetzen oder Aufstehen können nicht immer zuverlässig klassifiziert werden.

8.6.5. Ausblick

Das Modell kann gut als Grundlage für die Klassifizierung von Bewegungsdaten verwendet werden. Allerdings ist eine zuverlässige Erkennung von Transitionen nicht immer gegeben. Eine Möglichkeit dieser Tatsache entgegen zu wirken wäre, es durch eine Nachbereitung (beispielsweise einen Logikfilter) zu verfeinern. Ein Logikfilter könnte unlogisch aneinander gereihte Aktivitäten verhindern und die Ergebnisse verbessern. Wenn beispielsweise auf ein Sitzen ein Gehen folgt, kann ein Logikfilter dies verhindern und die Transition Sitzen-Stehen ergänzen.
Das fertige Modell besitzt derzeit einen moderaten Speicherbedarf und eignet sich auch um auf mobilen Endgeräten abgelegt zu werden. 9.3 Hierdurch ist zum Beispiel eine Aufnahme und direkte Klassifizierung per Smartphone denkbar. Außerdem ist es denkbar das Modell um Daten zu ergänzen, die nicht besonders häufig vorhanden sind. Hierdurch vergrößert sich zwar die Modellgröße, es ist aber denkbar, dass die Erkennungsgenauigkeit weiter verbessert werden kann, bzw. weitere Bewegungsmuster ergänzt werden können. Ein weiterer Ausblick könnte sein, das Modell zur Echzeitklassifizierung zu verwenden. Als Beispiel in Kombination mit Apache Kafka oder RabbitMQ. [10]

9. Evaluation

In diesem Kapitel werden verschiedene Funktionen auf die Brauchbarkeit ihrer Ergebnisse überprüft. Es wird die Android App, die für die Studiendurchführung validiert. Außerdem werden die Funktionen zur Labelvorbereitung und Labelnachbereitung evaluiert und die Modelle, die zur Klassifizierung verwendet werden, werden verglichen.

9.1. Validierung der Android App

Die in der Studie verwendete Android App wurde auf ihre Funktionalität hin validiert, damit den dokumentierten Ergebnissen vertraut werden darf. Im folgenden wird zunächst der Ablauf der Validierung beschrieben und im Anschluss das Ergebnis der Validierung diskutiert.

9.1.1. Ablauf der Validierung

Die Validierung wurde von vier Personen mithilfe der vier zur Verfügung stehenden Tablets ausgeführt. Eine fünfte Person gab die Anweisungen, damit alle vier Personen etwa zur gleichen Zeit jeden Schritt ausführen konnten. In folgender Reihenfolge wurden die Anweisungen ausgeführt.

1. Start der App
2. Eingabe einer ID (ID merken)
3. Start drücken
4. Buttons ausprobieren
    a) Zuhause

i. Rückwärtsgehen
ii. 30 sec warten
iii. Springen
iv. 30 sec warten
v. Treppe heruntersteigen
vi. 30 sec warten
vii. Sonstiges: keinen Titel eingeben
viii. 30 sec warten
ix. Sonstiges
x. Sonstiges: Radfahren eingeben
xi. 30 sec warten
xii. Sonstiges

b) Draußen
i. Gehen
ii. sec warten
iii. Drehen
iv. 30 sec warten
v. Treppe hinaufsteigen
vi. 30 sec warten
vii. Hocken
viii. 30 sec warten
ix. Hocken

c) Drinnen
i. Stehen
ii. 30 sec warten
iii. Sitzen
iv. 30 sec warten
v. Liegen (Rücken)
vi. 30 sec warten

vii. Liegen (Seite)
viii. 30 sec warten
ix. Aufstehen
x. 30 sec warten
xi. Hinsetzen
xii. 30 sec warten
xiii. Liegen-Sitzen
xiv. 30 sec warten
xv. Sitzen-Liegen
xvi. 30 sec warten
xvii. Gehen
xviii. 60 sec warten
xix. Gehen

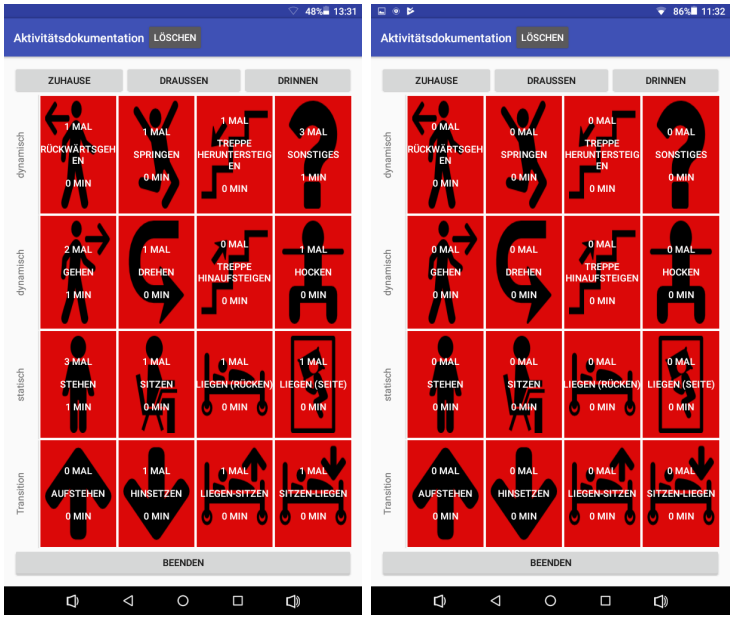
d) Drinnen
e) Löschen
   i. Einträge STAIR_CLIMB und SIT_STAND markieren
   ii. Button Löschen drücken
f) Stehen
g) 30 sec warten
h) Stehen
i) Stehen
j) 30 sec warten
k) Stehen
l) Beenden drücken
m) ID vom Anfang nochmal eingeben und START drücken (keine neue Messung)

Nach diesen Anweisungen sollte ein Screenshot gemacht werden. Der Screenshot sollte wie in Abbildung 9.1 a aussehen. Danach wurden noch weitere Anweisungen ausgeführt.

1. Nochmal Beenden

2. Dann die ID nochmal eingeben und Start (neue Messung)
3. Beenden

Jetzt wurde erneut ein Screenshot angefertigt. Die Ansicht sollte nun wie in Abbildung 9.1 b aussehen.



(a) Erster Screenshot (b) Zweiter Screenshot

Abbildung 9.1.: Screenshots für die Appvalidierung

9.1.2. Ergebnis der Validierung

Bei der Auswertung mit der Mamks Software können nur jeweils zwei Dateien verglichen werden. Also wurde jede Datei mit der gleichen anderen Datei verglichen. Dabei kamen folgende Ergebnisse heraus. Die Ergebnisse vom Vergleich von Datei 1 mit Datei 2 sind in Tabelle 9.2 zu sehen. Die Ergebnisse vom Vergleich von Datei 1 mit Datei 3 sind in Tabelle 9.3 zu sehen und die Ergebnisse vom Vergleich von Datei 1 mit Datei 4 sind in Tabelle 9.4 zu sehen. Die Genauigkeit der Vergleiche liegt bei allen drei Konfusionsmatrizen bei über 93%.

Schaut man sich die Labeldaten übereinandergelegt in der Mamks Software an, sieht man, dass alle vier Tablets sehr ähnliche Zeiträume und gleiche die Anzahl und Reihenfolge für jede Aktivität gespeichert haben. Es fällt auf, dass die bei den verschiedenen Tablets die gesamten Ergebnisse leicht verschoben sind. In den Abbildungen 9.5 bis 9.9 sind die Screenshots dieser Ansicht zu sehen.

	LIE_O...	LIE_O...	SIT	STAND	LIE_SIT	SIT_LIE	STAN...	BEND...	JUMP	STAI...	WALK	WAL...
LIE_O...	1781	1178	0	0	0	0	0	0	0	0	0	0
LIE_O...	0	1845	0	0	0	0	0	0	0	0	0	0
SIT	1149	0	1872	0	0	0	0	0	0	0	0	0
STAND	0	0	1162	5566	0	0	0	0	0	0	0	0
LIE_SIT	0	0	0	0	1836	1147	0	0	0	0	0	0
SIT_LIE	0	0	0	0	0	1820	0	0	0	0	1185	0
STAN...	0	0	0	0	1174	0	1825	0	0	0	0	0
BEND...	0	0	0	0	0	0	0	1837	0	0	0	0
JUMP	0	0	0	0	0	0	0	0	1802	1171	0	0
STAIR...	0	0	0	0	0	0	0	0	0	1876	0	0
WALK	0	0	0	0	0	0	0	0	0	0	6638	0
WALK...	0	0	0	0	0	0	0	0	1169	0	0	1894

(a) Konfusionsmatrix

	Recall	Precision	Accuracy	F1-Score
LIE_ON_BACK	0,6019	0,6078	0,9417	0,6049
LIE_ON_SIDE	1	0,6103	0,9705	0,758
SIT	0,6197	0,617	0,9421	0,6183
STAND	0,8273	1	0,9709	0,9055
LIE_SIT	0,6155	0,61	0,9419	0,6127
SIT_LIE	0,6057	0,6134	0,9416	0,6095
STAND_SIT	0,6085	1	0,9706	0,7566
BEND_OVER_OR_SQUAT	1	1	1	1
JUMP	0,6061	0,6065	0,9414	0,6063
STAIR_CLIMB_DOWN	1	0,6157	0,9707	0,7621
WALK	1	0,8485	0,9703	0,9181
WALK_BACKWARDS	0,6183	1	0,9707	0,7642
Average	0,7586	0,7608	0,961	0,743

(b) Präzisionsmatrix

Abbildung 9.2.: Vergleich von Datei 1 und Datei 2

	LIE_O...	LIE_O...	SIT	STAND	LIE_SIT	SIT_LIE	STAN...	BEND...	JUMP	STAI...	WALK	WAL...
LIE_O...	1817	0	1142	0	0	0	0	0	0	0	0	0
LIE_O...	1143	1873	0	0	0	0	0	0	0	0	0	0
SIT	0	0	1859	1162	0	0	0	0	0	0	0	0
STAND	0	0	0	5589	0	0	0	0	0	0	0	0
LIE_SIT	0	0	0	0	1819	0	1164	0	0	0	0	0
SIT_LIE	0	0	0	0	1167	1838	0	0	0	0	0	0
STAN...	0	0	0	0	0	0	1861	0	0	0	0	0
BEND...	0	0	0	0	0	0	0	1849	0	0	0	0
JUMP	0	0	0	0	0	0	0	0	1828	0	0	1145
STAIR...	0	0	0	0	0	0	0	0	1171	1876	0	0
WALK	0	0	0	0	0	1148	0	0	0	0	6703	0
WALK...	0	0	0	0	0	0	0	0	0	0	0	1919

(a) Konfusionsmatrix

	Recall	Precision	Accuracy	F1-Score
LIE_ON_BACK	0,6141	0,6139	0,943	0,614
LIE_ON_SIDE	0,621	1	0,9715	0,7662
SIT	0,6154	0,6195	0,9425	0,6174
STAND	1	0,8279	0,971	0,9058
LIE_SIT	0,6098	0,6092	0,9418	0,6095
SIT_LIE	0,6116	0,6155	0,9422	0,6136
STAND_SIT	1	0,6152	0,971	0,7618
BEND_OVER_OR_SQUAT	1	1	1	1
JUMP	0,6149	0,6095	0,9422	0,6122
STAIR_CLIMB_DOWN	0,6157	1	0,9708	0,7621
WALK	0,8538	1	0,9714	0,9211
WALK_BACKWARDS	1	0,6263	0,9714	0,7702
Average	0,763	0,7614	0,9616	0,7462

(b) Präzisionsmatrix

Abbildung 9.3.: Vergleich von Datei 1 und Datei 3

Die Verschiebung der Daten lässt sich damit erklären, dass zum Zeitpunkt der Validierung der App noch keine Lösung für die Zeitsynchronität der Tablets gegeben war. Die Uhrzeiten der Tablets unterscheiden sich also um einige Sekunden. Außer der Verschiebung der Daten lässt sich keine Ungleichheit in den Daten der App feststellen. Das lässt sich auch in Tabelle 9.1 erkennen. In der Tabelle sind die Zeiten, die das jeweilige Tablet für jede Aktivität aufgezeichnet hat in Millisekunden verzeichnet. Die Zeitdauern

	LIE_O...	LIE_O...	SIT	STAND	LIE_SIT	SIT_LIE	STAN...	BEND...	JUMP	STAI...	WALK	WAL...
LIE_O...	1340	0	1619	0	0	0	0	0	0	0	0	0
LIE_O...	1641	1375	0	0	0	0	0	0	0	0	0	0
SIT	0	0	1399	1622	0	0	0	0	0	0	0	0
STAND	0	0	0	4135	0	0	0	0	0	0	0	0
LIE_SIT	0	0	0	0	1349	0	1634	0	0	0	0	0
SIT_LIE	0	0	0	0	1652	1353	0	0	0	0	0	0
STAN...	0	0	0	0	0	0	1372	0	0	0	0	0
BEND...	0	0	0	0	0	0	0	1351	0	0	0	0
JUMP	0	0	0	0	0	0	0	0	1334	0	0	1639
STAIR...	0	0	0	0	0	0	0	0	1640	1407	0	0
WALK	0	0	0	0	0	1632	0	0	0	0	5724	0
WALK...	0	0	0	0	0	0	0	0	0	0	0	1422

(a) Konfusionsmatrix

	Recall	Precision	Accuracy	F1-Score
LIE_ON_BACK	0,4529	0,4495	0,911	0,4512
LIE_ON_SIDE	0,4559	1	0,9552	0,6263
SIT	0,4631	0,4636	0,9115	0,4633
STAND	1	0,7183	0,9557	0,836
LIE_SIT	0,4522	0,4495	0,9103	0,4509
SIT_LIE	0,4502	0,4533	0,9104	0,4518
STAND_SIT	1	0,4564	0,9554	0,6268
BEND_OVER_OR_SQUAT	1	1	1	1
JUMP	0,4487	0,4486	0,9105	0,4486
STAIR_CLIMB_DOWN	0,4618	1	0,9552	0,6318
WALK	0,7781	1	0,9555	0,8752
WALK_BACKWARDS	1	0,4646	0,9553	0,6344
Average	0,6636	0,6586	0,9405	0,6247

(b) Präzisionsmatrix

Abbildung 9.4.: Vergleich von Datei 1 und Datei 4

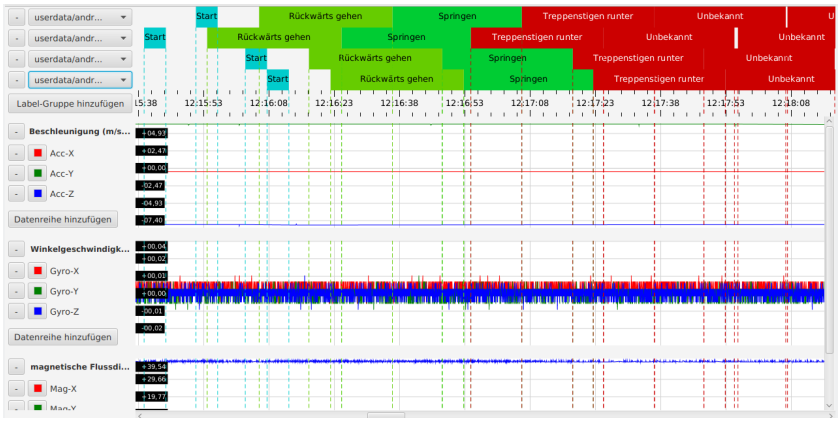


Abbildung 9.5.: Darstellung der Daten in Mamks: Bereich 1

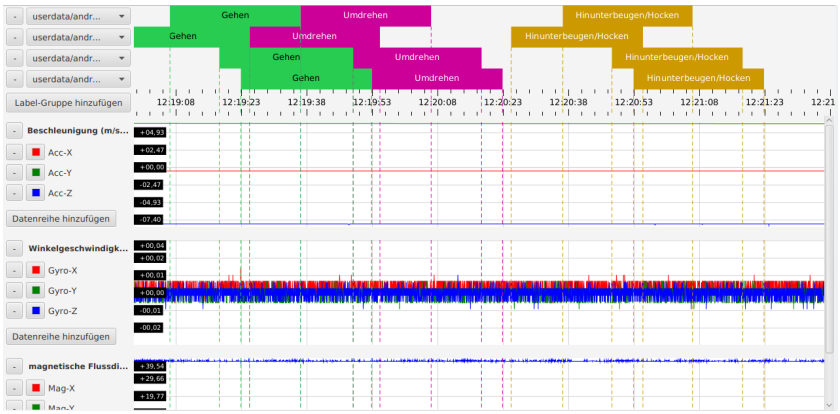


Abbildung 9.6.: Darstellung der Daten in Mamks: Bereich 2

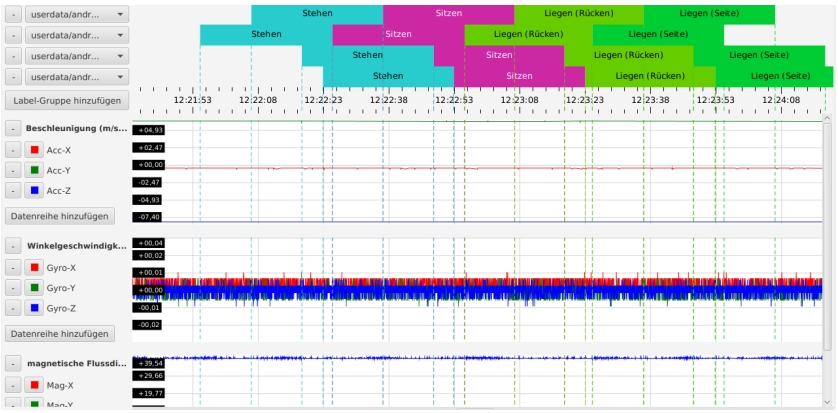


Abbildung 9.7.: Darstellung der Daten in Mamks: Bereich 3

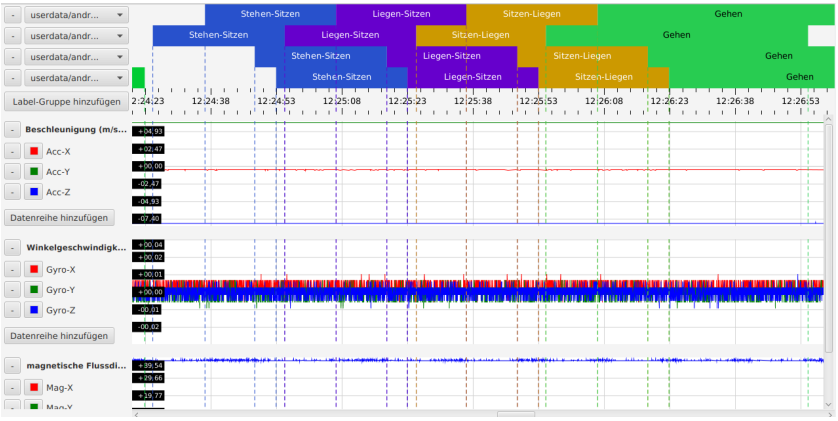


Abbildung 9.8.: Darstellung der Daten in Mamks: Bereich 4

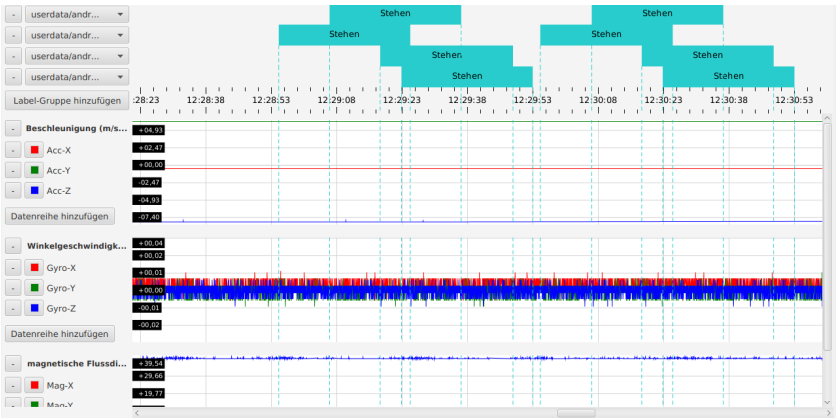
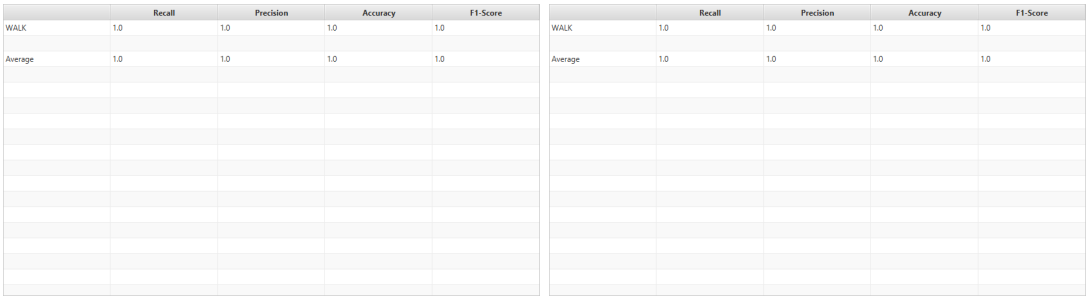


Abbildung 9.9.: Darstellung der Daten in Mamks: Bereich 5

Tabelle 9.1.: Dauer der einzelnen Aktivitäten

	Tablet 1	Tablet 2	Tablet 3	Tablet 4	Durschschnitt
Gehen	89837	90263	90762	89995	90214
Drehen	29936	29904	29446	30003	29822
Treppe hoch	0	0	0	0	0
Hocken	29738	30212	29969	29885	29951
Rückwärtsgehen	30630	30882	30637	30606	30688
Springen	29729	29705	29991	29745	29793
Treppe runter	30472	30469	30125	30351	30354
Sonstiges	59658	59384	58753	58616	59103
Sitzen	90554	90879	90959	90306	90675
Stehen	30215	30347	30007	30184	30197
Liegen (Seite)	29583	29299	29600	29802	29571
Liegen (Rücken)	30160	30229	30212	30012	30153
Aufstehen	0	0	0	0	0
Hinsetzen	29994	30193	30249	30061	30124
Liegen-Sitzen	29832	30101	29858	30008	29949
Sitzen-Liegen	30050	29672	29861	29851	29859

weichen maximal 555 Millisekunden (Kategorie Sonstiges) von den Durschschnittswerten der Zeitdauern ab. Um zu testen, ob die Verschiebungen an den Zeitdifferenzen der Tablets liegen, wurde noch ein zusätzlicher Test gemacht. Diesmal wurden alle Tablets zuvor mit dem gleichen Zeitserver synchronisiert. In Abbildung 9.10 sind die Präzisionsmatrizen für den Vergleich von Datei 1 mit Datei 2 und für den Vergleich von Datei 1 mit Datei 3 dargestellt. Dabei fällt auf, dass alle Werte bei 100% liegen. In Abbildung 9.11 sieht man die Ergebnisse aus der Mamks Software. Die Abbildungen bestätigen, dass die Verschiebungen an den unsynchronisierten Tablets lagen.



	Recall	Precision	Accuracy	F1-Score
WALK	1.0	1.0	1.0	1.0
Average	1.0	1.0	1.0	1.0

(a) Vergleich Datei 1 mit Datei 2

	Recall	Precision	Accuracy	F1-Score
WALK	1.0	1.0	1.0	1.0
Average	1.0	1.0	1.0	1.0

(b) Vergleich Datei 1 mit Datei 3

Abbildung 9.10.: Präzisionsmatrizen für die zeitsynchronen Daten

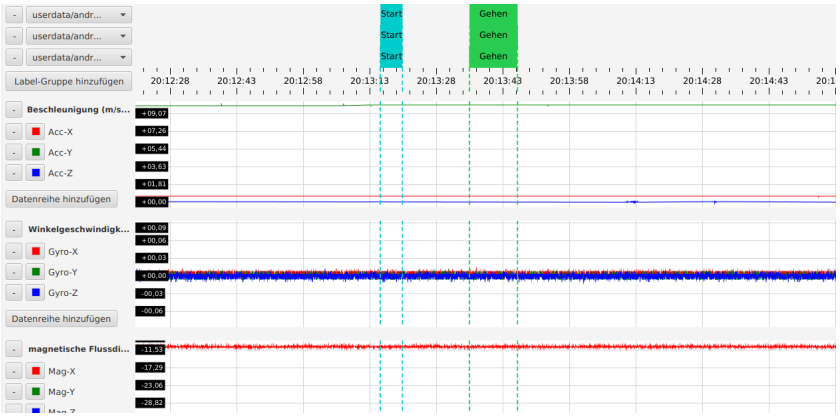


Abbildung 9.11.: Zeitsynchrone Daten in Mamks dargestellt

9.2. Labelvorbereitung

Es sollten Möglichkeiten untersucht werden, mit den es möglich ist ungelabelte Daten vorbereitend zu annotieren. Das Ziel war es das nachträgliche manuelle Labeln zu erleichtern. Um dies zu erreichen, wurden zwei einfache Verfahren umgesetzt. Das automatisierte Labeln von Drehen-Aktivitäten konnte anhand von Zusatzdaten, die für die Validierung von Drehungen von der Projektgruppe gesammelt worden sind, evaluiert werden.

9.2.1. Automatisierte Beschriftung von Drehungen

Da die Drehung im Stehen, Sitzen oder Gehen eine besonderes einfach zu beschreibende Aktivität (Sogar eher eine Basisbewegung) ist, erschien es sinnvoll diese rechnerisch zu bestimmen.
Eine Drehbewegung erfolgt auf der sagitto-frontalen Achse , was der Y-Achse des Gyroskop-Sensors entspricht. Die Dauer und Geschwindigkeit der Drehung sind stark variabel. Um die Rechnerische Bestimmung zu ermöglichen, wurde eine Mindestdauer der Drehung von 0.5 Sekunden festgelegt, sowie eine Mindestgeschwindigkeit von 0.5 rad/sec. Betrachtet wird ausschließlich die Y-Achse des Gyroskops.

Die Abbildung 9.12 zeigt das Ergebnis des Algorithmus. Abbildung 9.12 enthält Drehaktivitäten eines Probanden mit der Probanden ID 21. Eingeblendet sind drei Label. Die oberste Zeile beinhaltet die manuell annotierten Drehungen. Die zweite Label-Gruppe enthält die, von dem entwickelten Algorithmus zur Vorbereitung von Dreh-Bewegungen generierten Label. Die dritte Label-Gruppe enthält Annotationen von Zuständen, die mit Hilfe des Signalvektor Magnitude Merkmals generiert worden sind (dazu später).
Zu sehen ist, dass der Algorithmus im Vergleich zur manuellen Annotation ein einheitliches Ergebnis liefert. Der Algorithmus kann außerdem, je nach Anforderung in seinen

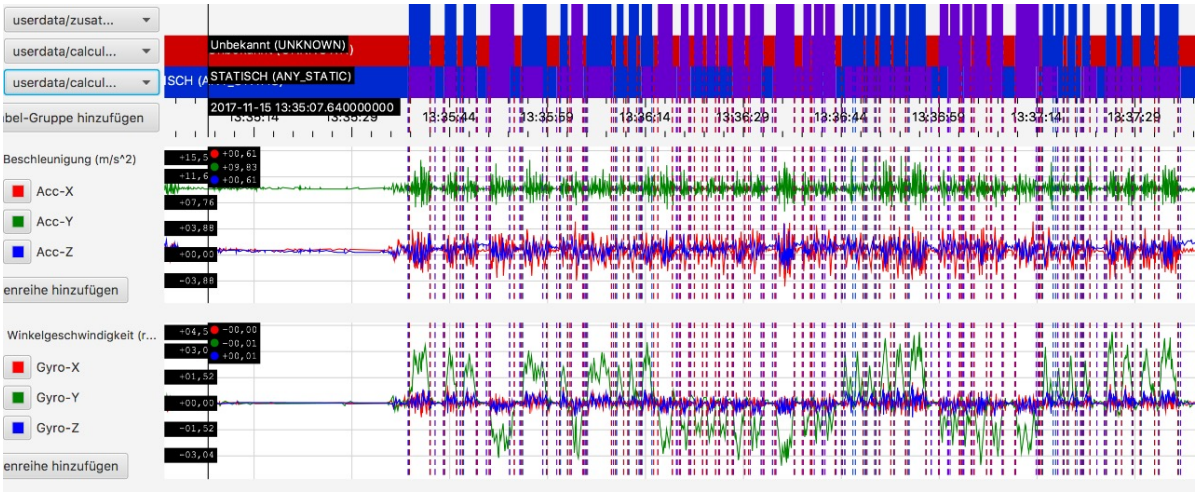


Abbildung 9.12.: Vorbereitende Annotation von Drehbewegungen

Parametern so angepasst werden, dass nur länger andauernde oder nur schnelle Drehbewegungen annotiert werden können.
Quantitativ wurde der Algorithmus mit drei Datensätzen ausgewertet, wobei ausschließlich Drehungen ausgeführt worden sind. Tabelle 9.2 zeigt das Ergebnis des Algorithmus.

Tabelle 9.2 beinhaltet Evaluationsergebnisse von drei Probanden-Datensätzen. Jeder Proband hat sich etwa 30 Minuten im Stehen, abwechselnd nach rechts und nach links gedreht. Die Ergbnisse zeigen Präzision, Recall, Accuracy und F1-Score für Drehungen nach rechts (Wert rechts) und nach links (Wert links). Jeder Proband hat sich etwa 30 Minuten im Stehen, abwechselnd nach rechts und nach links gedreht. Anschließend haben die PG-Teilnehmer die Datensätze annotiert. Da es zum Zeitpunkt der Aufnahme noch keine einheitlichen Annotationskonventionen gab, sind die Ergebnisse unterschiedliche ausgefallen. Das beste Anotationsergebnis des Algorithmus ergab sich auf dem Datensatz mit der PID 21. Hierbei konnte ein F1-Score von 97.1/97.5% erreicht werden.

Proband ID	Recall	Precision	Accuracy	F1	Bemerkung
21	0.944/0.9517	1.0/0.99	0.972/0.976	0.971/0.975	Keine Auffälligkeiten
454	0.655/0.72	1.0/1.0	0.99/0.726	0.79/0.84	Einige Drehungen nicht manuell gelabelt
48	0.83/0.75	0.99/1.00	0.94/0.836	0.902/0.859	Einige Drehungen nicht manuell gelabelt

Tabelle 9.2.: Evaluation der Drehbewegungen

9.2.2. Automatisierte Beschriftung von Zuständen

Die Projektgruppe MAMKS hatte, bei ihrer Definition von Aktivitäten eine Unterscheidung von statischen und dynamischen Zuständen eingeführt. Dabei ist eine Aktivität dynamisch, wenn die Signalmagnitude mehrerer Signalwert von dem Mittelwert abweicht. Diese Aufteilung scheint algorithmisch leicht umsetzbar zu sein. Eine genau Formulierung des Signalvektor-Merkmals wurde bereits in der Dokumentation der Projektgruppe MAMKS eingeführt.
Wie in der Abbildung 9.12 zu sehen ist, wurde ein Algorithmus umgesetzt, der eine Zustandsänderung markiert. Hierbei wird ein Merkmal namens Signal Vektor Magnitude (SVM) eingesetzt, das zum Ziel hat starke Schwankungen in Signaldaten zu detektieren. Für die Berechnung des Merkmals wurden alle Daten des Accelerometers und des Gyroskops genutzt und über ein Zeitfenster von 0.5 Sekunden (50 Signale) ausgewertet.
Da keine direkten Ground-Truth Daten zur Auswertung zur Verfügung stehen, wurde die Daten der Drehaktivitäten zur Evaluation eingesetzt. Die Abbildung 9.12 veranschaulicht eine Übereinstimmung der Markierungen mit den Drehaktivitäten, was zumindest qualitativ auf eine gute Abdeckung hinweist. Auch dieser Algorithmus, kann parame-

	Assessment	Datensatz 1	Datensatz 2
MAMKS Model	96,01% - XX MB	56,37% - XX MB	
KNN	96,2% - 51 MB	85,18% - 703 MB	85,12% - 703 MB
Bagging Trees	96,38% - 33 GB	92,17% - 100 GB	92,28% - 97 GB
CNN	85% - 0,1 MB	92% - 29 MB	92% - 29 MB
BDT	82,8% - 700 MB	77,34% - 500 MB	77,34% - 500 MB
QDA	89,8% - 8 KB	79,23% - 11 KB	75,41% - 11 KB

Tabelle 9.3.: Ergebnistabelle 1 von 2

trisiert werden, so dass kleiner oder größere Fenster, je nach Anforderung ausgerechnet werden können. Ein weiterer Vorteil ist auch bei diesem Algorithmus eine einheitliche Trennung zwischen den einzelnen Aktivitäten. Eine Ausnahme ist hier die fehlende Trennung zwischen zwei dynamischen Aktivitäten. Hierbei muss der Nachbearbeiter die Grenze manuell bestimmen.

9.3. Ergebnisse

Die Ergebnisse der einzelnen Algorithmen sind folgend in den Tabellen 9.3 und 9.4 dargestellt:

Bei näherer Betrachtung fällt auf, dass ein Bagging Tree Modell signifikant größer ist, als die anderen erzeugten Modelle. Im Gegensatz dazu ist das Modell der QDA mit maximal 12 KB das kleinste angelernte Modell. Bagging Trees hat zwar im Durchschnitt über alle Datensätze gesehen die besten F1-Score-Ergebnisse, allerdings ist die Größe von mindestens 33GB und der dafür benötigte Zeitaufwand zum Trainieren überdurchschnittlich hoch. Der von der PGMAMKSFZ final gewählte Datensatz zum Anlernen der Modelle, ist der Datensatz 0. Auf diesem Datensatz schneidet der CNN mit einer Wahrscheinlichkeit von 92.66% am besten ab. Außerdem ist die Modellgröße mit 118

	Datensatz 3	Datensatz 0
MAMKS Model		
KNN	87,55% - 706 MB	90,76% - 707 MB
Bagging Trees	92,19% - 99 GB	91,73% - 118 GB
CNN	92% - 29 MB	92,66% - 118 MB
BDT	77,36% - 500 MB	86,71% - 2 GB
QDA	72,33% - 10 KB	79,81% - 12 KB

Tabelle 9.4.: Ergebnistabelle 2 von 2

MB sehr klein. Das zweitbeste Ergebnis ist das optimierte Modell des Bagging Trees mit 91.73%, gefolgt vom KNN mit 90.76%. Da das Modell des Bagging Trees allerdings 118 GB beträgt, ist es generell nicht besonders gut zur weiteren Verwendung (insbesondere in Betracht auf eine eventuelle Implementierung im Mobil-Bereich) geeignet. Gegenüber dem CNN- ist das KNN-Modell sechsfach so groß, allerdings mit einer Größe von 707 MB noch zur Verwendung gut händelbar. Der BDT hat hingegen eine höhere Modellgröße (2 GB) und schneidet mit einem F1-Score von 86.71% sogar schlechter ab. Als kleinstes, allerdings auch schlechtestes Ergebnis, ist die das Modell der QDA zu nennen. Es erzielt auf dem Datensatz 0 einen F1-Score von 79.81%.

Bei der Betrachtung der Tabelle fällt außerdem auf, dass das ursprüngliche MAMKS Modell eine hohe Wahrscheinlichkeit beim Erkennen des Assessment-Datensatzes hat. Sobald es allerdings mit den Zusatzdaten antrainiert wird, werden die Erkennungswahrscheinlichkeiten signifikant schlechter (Verschlechterung um fast 40%). Eine Detailbetrachtung der einzelnen Algorithmen und den optimierten Modellen mit Ergebnissen und Deutungen, kann in den jeweiligen Optimierungskapiteln und den Seminararbeiten eingesehen werden.

9.4. Labelnachbereitung

Zur Verbesserung der Klassifikationsergebnisse der einzelnen Algorithmen wurde der implementierte Mehrheitsentscheid und die Rule-Engine getestet. Die Funktionen wurde auf den Labeldateien der Algorithmen CNN, Weighted KNN und QDA auf allen 29 Senior-Home-Datensätzen ausgeführt. Die Labeldateien der Algorithmen Bagged Trees und Boosted Decision Trees waren auf den Trainingsdaten überangepasst und wurden daher nicht verwendet. Der Mehrheitsentscheid wurde mit den Kombinationen CNN-WKNN-QDA, CNN-WKNN, CNN-QDA und WKNN-QDA getestet. Als Parameter für den Radius wurden 50, 100 und 150 Sample bei einen Entscheidungskriterium von 55% eingestellt. Die generierten Labeldateien wurden mit dem LabelScoring-Algorithmus der MAMKS-Software für die Datensätze einzeln ausgewertet und in der Tabelle 9.13 zusammengefasst. Für die Rule-Engine wurde eine Regel-Datei erstellt in der definiert wird, dass die einzelnen Label nur gültig sind, wenn sie eine Mindestdauer von 200 Millisekunden haben und sonst mit Unknown ersetzt werden.

Für die Algorithmen wurde mit dem Scoringverfahren ein F1-Score von 76,65% für CNN, 45,91% für QDA und 81,66% für Weighted KNN ermittelt. Für die verschiedenen Verfahren konnten die besten Ergebnisse beim Mehrheitsentscheid mit den Labeldateien der Algorithmen CNN und WKNN erzielt werden. Bei diesen liegt der F1-Score bei einem Wert von 84,65% bei einem Radius von 50 Sample und bei 84,33% bei einem Radius von 100. Dies entspricht einer Verbesserung von ca. 3% im Vergleich zum besten Einzelergebnis.
Bei den gewählten Radien von 50, 100 und 150 Sample lässt sich beim Mehrheitsentscheid für den Recall keine eindeutige Tendenz bezüglich einer Verbesserung erkennen. Für die Precision ergeben sich die besten Werte bei 100 Sample. Die Anwendung der Regeln ermöglichte eine Verbesserung der Precision bei allen Modellen, der Recall verschlechtert sich jedoch.

	Recall	Precision	Accuracy	F1-Score
cnn4har	81,40 %	72,42 %	98,68 %	76,65 %
qda	38,66 %	56,52 %	95,26 %	45,91 %
wknn_7	77,23 %	86,64 %	98,75 %	81,66 %
MajorityVoting_cnn4har_qda_wknn_7_50.0	74,77 %	86,33 %	98,58 %	80,14 %
MajorityVoting_cnn4har_qda_wknn_7_100.0	73,11 %	87,23 %	98,63 %	79,55 %
MajorityVoting_cnn4har_qda_wknn_7_150.0	74,62 %	82,17 %	98,54 %	78,21 %
MajorityVoting_qda_wknn_7_50.0	55,55 %	90,12 %	98,81 %	68,73 %
MajorityVoting_qda_wknn_7_100.0	55,55 %	90,12 %	98,81 %	68,73 %
MajorityVoting_qda_wknn_7_150.0	51,17 %	87,12 %	98,55 %	64,47 %
MajorityVoting_cnn4har_qda_7_50.0	48,51 %	70,57 %	98,15 %	57,50 %
MajorityVoting_cnn4har_qda_7_100.0	47,70 %	71,51 %	98,01 %	57,23 %
MajorityVoting_cnn4har_qda_7_150.0	50,39 %	69,46 %	97,80 %	58,41 %
MajorityVoting_cnn4har_wknn_7_50.0	80,43 %	89,33 %	99,23 %	84,65 %
MajorityVoting_cnn4har_wknn_7_100.0	79,55 %	89,73 %	99,18 %	84,33 %
MajorityVoting_cnn4har_wknn_7_150.0	76,99 %	85,55 %	99,05 %	81,04 %
cnn4harRulesEnforced	75,65 %	82,85 %	98,84 %	79,09 %
qdaRulesEnforced	36,68 %	61,73 %	95,58 %	46,02 %
wknn_7RulesEnforced	71,54 %	89,88 %	98,80 %	79,67 %

Abbildung 9.13.: Ergebnisse der Nachbearbeitung über die Labeldateien von CNN, WKNN und QDA

10. Fazit

In der diesjährigen Projektgruppe PGMAMKSFZ sollte die MAMKS-Software, die von der Projektgruppe im letzten Jahr entwickelt wurde, zum einen durch Klassifikationalgorithmen optimiert werden, als auch die Umsetzung von Labelvorbereitung und Nachbereitung, sowie Aktivitätstracking während der Studie erarbeitet werden.

Für die Optimierungen wurden verschiedene Algorithmen gewählt, die wiederum in MATLAB und Python ausgelagert wurden. Zu diesen gehören das Convolutional Neuroal Network (CNN), die Quadratische Diskriminanzanalyse kurz QDA, Bagging Trees, Boosted Decision Trees und K-Nearest Neighbor. Für das Anlernen der Algorithmen wurden verschiedene Datensatz-Gruppen gebildet, die sich in ihren Merkmalen der verschiedenen Aktivitäten unterscheiden, siehe Tabelle 8.15.

Bei dem entwickelten Modell des CNN wurde eine Genauigkeit von 92,66% des Datensatzes 0 (FULL) erreicht. Allerdings ist zu befürchten, dass das Ergebnis der Evaluation nicht aussagekräftig ist, da einige Aktivitäten seltener vorkommen als andere. Beim QDA wurde eine Genauigkeit von 79,81% erreicht. Dabei kann sich die geringe Genauigkeit auf die schlecht gelabelten Daten zurückführen oder auf die wenig vorhanden Aktivitäten, wie ”Treppe steigenöder ”Hinsetzen”. Der Bagging Trees Algorithmus erreichte eine Genauigkeit von 91,73%. Da die Berechnung sehr speicherintensiv war, wurde diese auf dem Cluster der Carl-von- Ossietzky Universität Oldenburg ausgelagert. Beim Boosted Decision Tree Algorithmus wurde eine Genauigkeit von 86,71% erzielt. Durch die hohe Rechenintensität wurde die Berechnung auch hier auf das Cluster ausgelagert. Insgesamt wurden die Aktivitäten Gehen und Stehen gut bis sehr gut erkannt, weniger gut das Sitzen. Oftmals wurde hierbei das Stehen annotiert. Der KNN Algorithmus erzielte eine Erkennungsrate von 90,76%. Es hat sich gezeigt, dass der Algorithmus nicht nur

eine gute Grundlage für die Klassifizierung von Bewegungsdaten ist, sondern durch seinen moderaten Speicherbedarf auch auf mobile Geräte direkt angewendet werden kann. Einzig die Erkennung von Transitionen, wie das Hinsetzen oder Aufstehen werden nicht immer zuverlässig klassifiziert. Durch die Anwendung eines Mehrheitsentscheids auf die Labeldaten von CNN und Weighted KNN konnte nochmal eine Verbesserung der Klassifikation erreicht werden.

Des Weiteren wurde eine Android basierte App entwickelt, die die Aktivität der Probanden dokumentiert und daraus Label zu den Gürteldaten generiert. Um diese Labeldaten zu importieren, wurde die MAMKS-Anwendung um einer entsprechenden Funktion erweitert. Durch weitere Änderungen an der MAMKS-Anwendung ist es möglich ganze Label oder die Ränder der Label zu verschieben. Weiterhin können durch Doppelklick auf einen leeren Bereich, neue Labels hinzugefügt werden. Für die Labelnachbearbeitung wurde die MAMKS-Software ebenfalls erweitert. Dabei stehen zwei Verfahren zur Auswahl: Der Mehrheitsentscheid und die regelbasierte Korrektur (Rule Engine). Ergänzend dazu, wurde ein Statistik Modul in der Software implementiert, dass sämtliche Label und Labelgruppen gegenüberstellt und diese visualisiert.

Zu dem Projektgruppenthema wurde außerdem von jedem Gruppenmitglied eine Ausarbeitung angefertigt, die aufbauend für die Arbeit dieser Projektgruppe war. Folgende Themen werden im Anschluss vorgestellt: Diskriminanzanalyse, Faltende Neuronale Netze zur Aktivitätserkennung in den Daten des Sensorgürtel, körpernahe Sensoren mittels IMU, K-Nearest-Neighbor, Algorithmische Lösungen zur Unterstützung der Labelnachbearbeitung, Boosting Neuronal Networks, Bagging Trees, Realtime event processing von Sensordaten und Aktivitätserkennung und Labelnachbearbeitung.

11. Ausblick

Für die Zukunft hat die PGMAMKSFZ Grundlagen in mehreren Bereichen geschaffen, die für kommende Projektgruppen Möglichkeiten zum Weiterarbeiten bietet. Zum Einen können in der Softwareentwicklung Erweiterungen der MAMKS Software vorgenommen werden, indem das bisherige Frontend komplett durch ein Web-Interface ersetzt oder um eines ergänzt wird, um die MAMKS Anwendung auch als mobile Applikation nutzen zu können. Ein weiteres Gebiet stellt die Erweiterung der Klassifikationsalgorithmen dar, die um weitere Modelle oder Aktivitäten erweitert werden und somit verbessert werden können. Bisher wurden die Klassifikationsalgorithmen mit weniger Aktivitäten angelernt. Deshalb können weitere Aktivitäten aufgenommen werden, wenn mithilfe weiterer Studien mit Probanden Daten gesammelt und annotiert werden. Weiterhin können Erweiterungen durchgeführt werden, die es möglich machen, Daten die während einer weiteren Studiendurchführung ermittelt werden, über eine drahtlose Kommunikationsschnittstelle an ein zentrales Cluster zu schicken. Dazu können die Gürtel mit Bluetooth Sender oder ähnlichem ausgestattet werden, die es möglich machen, mit dem Tablet oder Rechner zu kommunizieren und die Daten zu übertragen. Mit geeigneten Klassifikationsalgorithmen wäre auch der Schritt denkbar, die ermittelten Daten in Echtzeit auszuwerten, die ein Gürtelnutzer mithilfe einer drahtlosen Kommunikationsschnittstelle aufzeichnet. Dazu wären Plattformen wie Apache Kafka und Storm hilfreich, welche für die drahtlose Kommunikation- und Echtzeitdatenverarbeitung zuständig wären. Dadurch wäre es möglich die Daten drahtlos an zentrale Orte in Echtzeit zu schicken, was den Wegfall des manuellen Downloads der Daten vom Gürtel zum Rechner ermöglicht. Zusätzlich könnten damit die Klassifikationsalgorithmen im Praxistests erprobt und verbessert werden. Dies würde auch einem Schritt der Vision von Assistentzsystemen näher kommen, die in der Lage sind, die Aktivitäten von Gürtelnutzern richtig zu erkennen und den Benutzer über seine Fehlbewegungen zu informieren oder hinzuweisen.

Literatur

[1] S Abass und Z Abdulhassan. „Kinematic analysis of human climbing up and down stairs at different inclinations“. In: *Eng. &Tech. Journal* 31 (2013).

[2] Andreas Mueller. *Machine learning cheat sheet (for scikit-learn)*. URL: `http://peekaboo-vision.blogspot.de/2013/01/machine-learning-cheat-sheet-for-scikit.html` (besucht am 13. 03. 2018).

[3] Stephanie B. Michaud; Steven A. Gard; Dudley S. Childress. „A preliminary investigation of pelvic obliquity patterns during gait in persons with transtibial and transfemoral amputation“. In: *Journal of RehabilitationResearch and Development* 37.1 (Jan. 2000), S. 1–10. URL: `https://www.rehab.research.va.gov/jour/00/37/1/michaud.html`.

[4] Clinical Gait. *Assessment of Gait*. URL: `https://clinicalgate.com/assessment-of-gait/` (besucht am 13. 03. 2018).

[5] *Comparing Top Deep Learning Frameworks*. URL: `https://deeplearning4j.org/compare-dl4j-tensorflow-pytorch` (besucht am 01. 04. 2018).

[6] Diane Cook. *Overfitting in Decision Tree Learning*. URL: `http://www.eecs.wsu.edu/~cook/dm/lectures/l4/node17.html` (besucht am 13. 02. 2018).

[7] Tommaso Dorigo. *Decision Trees, Explained To Kids*. URL: `http://www.science20.com/tommaso_dorigo/decision_trees_explained_to_kids-224948` (besucht am 14. 03. 2018).

[8] Ron Kohavi u. a. „A study of cross-validation and bootstrap for accuracy estimation and model selection“. In: *Ijcai*. Bd. 14. 2. Montreal, Canada. 1995, S. 1137–1145.

[9] Leo Breiman. *Bagging Predictors*. 1996. URL: https://link.springer.com/article/10.1023/A:1018054314350 (besucht am 13. 02. 2018).

[10] Michael Becker. *Realtime predictive analytics using scikit-learn and RabbitMQ*. URL: https://www.youtube.com/watch?v=WPyNdHygBD0 (besucht am 15. 04. 2018).

[11] Carl von Ossietzky Universität Oldenburg. *AEQUIPA Technology - Versa*. URL: https://www.uni-oldenburg.de/medizintechnik/forschung/projekte/aequipa/versa-technology/ (besucht am 20. 03. 2018).

[12] Patrick. *What does recall mean in Machine Learning?* URL: https://stackoverflow.com/questions/14117997/what-does-recall-mean-in-machine-learning (besucht am 16. 04. 2018).

[13] Sebastian Raschka. *Python machine learning: Unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics*. Community experience distilled. Birmingham und Mumbai: Packt Publishing open source, 2016. ISBN: 978-1-78355-513-0.

[14] Margaret Schenkman u. a. „Whole-Body Movements During Rising To Standing From Sitting". In: *Physical therapy*. Bd. 70. Nov. 1990, 638–48; discussion 648.

[15] Brad J Schoenfeld. „Squatting Kinematics and Kinetics and Their Application to Exercise Performance". In: *The Journal of Strength & Conditioning Research*. 2010.

[16] scikit-learn. *scikit-learn FAQ*. URL: http://scikit-learn.org/stable/faq.html#what-are-the-inclusion-criteria-for-new-algorithms (besucht am 16. 03. 2018).

[17] *Squat showing hip/knee/ankle relationship from varying positions*. http://www.evolving-fitness.com/blog/2015/4/8/breaking-down-squats-from-a-biomechanics-perspective. (Besucht am 25. 03. 2018).

[18] Robert Staszkiewicz u. a. „Three-dimensional analysis of the pelvic and hip mobility during gait on a treadmill and on the ground“. In: *Acta of bioengineering and biomechanics / Wrocław University of Technology*. Bd. 14. Juli 2012, S. 83–9.

[19] Jeff Sutherland. *Jeff Sutherland's Scrum Handbook*. Jan. 2010.

A. Verfahrensanweisungen

Verfahrensanweisung	**VA-Nr. Versa-10**
Titel: **Terminplanung Hausbesuche**	**Gültig ab: 04.08.2017** **Geplante Revision: bei Bedarf**
	Anlagen: **VA-11**

Freigabe:

Prof. Andreas Hein	Datum	Unterschrift
Sandra Hellmers	Datum	Unterschrift
Sebastian Fudickar	Datum	Unterschrift
Sandra Lau	Datum	Unterschrift
Elke Onken	Datum	Unterschrift
Andrea Heinks	Datum	Unterschrift
Lena Dasenbrock	Datum	Unterschrift
Finjas Künemann	Datum	Unterschrift
PG Beatrice Coldewey	Datum	Unterschrift
PG Eugen Lange	Datum	Unterschrift
PG Marius Leyh	Datum	Unterschrift
PG Jan-Frederik Scharnowski	Datum	Unterschrift
PG Maren Steinkamp	Datum	Unterschrift

PG Alexander Thomas	Datum	Unterschrift
PG Felix Van Der Ahe	Datum	Unterschrift
PG Patrick Warszewik	Datum	Unterschrift
PG Marlon Willms	Datum	Unterschrift

1. Kurzbeschreibung des Vorgangs

Als Trainingsdaten und als Gold-Standard für die Validierung der Annotationen sollen die Aktivitäten der Senioren im häuslichen Bereich manuell gelabelt werden.

Dies erfordert die Information der Senioren über das Vorhaben. Nach erfolgtem Einverständnis wird ein geeigneter Termin mit den Senioren abgesprochen.

2. Mitarbeiter, für die die VA verbindlich ist

alle Mitarbeiter AEQUIPA-Versa (Andrea Heinks, Elke Onken, Sandra Lau, Sebastian Fudickar, Lena Dasenbrock, Sandra Hellmers, Finjas Künemann)

alle Teilnehmer der Projektgruppe PGMAMKS (Beatrice Coldewey, Eugen Lange, Marius Leyh, Jan-Frederik Scharnowski, Maren Steinkamp, Alexander Thomas, Felix Van Der Ahe, Patrick Warszewik, Marlon Willms)

3. Beschreibung des Vorgehens

Für die Kommunikation der Termine wurde das Funktions-Konto "VERSA-ZUHAUSE" (versa-zuhause@uni-oldenburg.de) eingerichtet.

Die Projektgruppenteilnehmer (PG-Teilnehmer) tragen mögliche Zeitslots, in denen Sie Termine wahrnehmen können, im Kalender des Kontos "VERSA-ZUHAUSE" mit Vor- und Nachname ein.

Die Studynurses selektieren anhand der Postleitzahlenliste die relevanten Probanden aus.

Den Studynurses liegt eine Liste der PID/SHIDs vor.

Die Studynurses informieren die Probanden über das Vorhaben und suchen nach erfolgtem Einverständnis zur Terminabsprache zusammen mit den Probanden einen passenden Zeitslot:

Dieser liegt optimaler Weise 1-3 Tage nach der 7-tägigen Heimmessung und hat eine Dauer von 2 bis 4 Stunden. Zwischen zwei Terminen muss eine Stunde frei sein, wenn sie von den gleichen PG-Teilnehmern durchgeführt werden.

Zu dem ausgewählten Zeitslot müssen 2 PG-Teilnehmer (oder einer + Finjas) Zeit haben. Wird für ein Ehepaar (2 Probanden) ein Termin vergeben, müssen 3 PG-Teilnehmer Zeit haben. Es ist darauf zu achten, dass die freien Zeitslots nicht doppelt vergeben werden (z. B. bei 4 freien Zeitslots max. 2 Besuchstermine).

Wurde ein passender Zeitslot gefunden, wird dieser als neuer Termin „Besuch“ in den Kalender eingetragen und in den Kommentaren der Name des Probanden, SHID, Adresse und Telefonnummer notiert. Die Adresse ist nochmals mit den Probanden zu kontrollieren (genau mit Adresse und Hausnummer und Telefonnummer).

Sollten die Probanden besondere Aktivitäten für den Zeitslot geplant haben z. B. Fahrrad fahren, Chorsingen, Einkaufen muss auch dies in den Kommentaren vermerkt werden.

Für die Auswertung der Gürtel-Daten wird des Weiteren ein Grundriss der Wohnungen benötigt. In den Kommentaren ist daher zu vermerken, ob dieser von den PG-Teilnehmer vor Ort erstellt werden muss oder ob ein Grundriss, der abgeschrieben/abfotografiert werden kann, vorhanden ist.

Liegt der Termin direkt nach der 7-tägigen Heimmessung, kann den Probanden angeboten werden, dass die PG-Teilnehmer den in dieser Zeit verwendeten Gürtel (und sonstiges Material) wieder mitnehmen (ebenfalls in den Kommentaren zu vermerken).

Die PG-Teilnehmer, die den Besuch durchführen, tragen nachträglich ihre Namen in die Kommentare ein.

Werden Termine abgesagt, werden diese wieder im Kalender gelöscht. Bei kurzfristiger Absage (24 Stunden vor dem Termin) werden die in den Kommentaren genannten PG-Teilnehmer telefonische informiert.

Wenn ein Proband keinen Termin machen kann (kein Kalender dabei oder kein passender Zeitslot), erfolgt eine weitere Terminabsprache über Finjas Künemann (Name, Telefonnummer, SHID notieren). Zur Not können ggf. weitere Terminslots mit den PG-Teilnehmern erfragt und abgesprochen werden.

4. Dokumentation

https://mail.uni-oldenburg.de/owa/#path=/calendar/view/Month

5. Verteiler

6. mitgeltende Unterlagen

VA-11

Funktions-Konto "VERSA-ZUHAUSE" (versa-zuhause@uni-oldenburg.de)

Verfahrensanweisung	**VA-Nr. Versa-11**
Titel: **Hausbesuche Durchführung und Datenarchivierung**	**Gültig ab: 04.08.2017** **Geplante Revision: bei Bedarf**
	Anlagen: **VA-10**

Freigabe:

Prof. Andreas Hein	Datum	Unterschrift
Sandra Hellmers	Datum	Unterschrift
Sebastian Fudickar	Datum	Unterschrift
Sandra Lau	Datum	Unterschrift
Elke Onken	Datum	Unterschrift
Andrea Heinks	Datum	Unterschrift
Lena Dasenbrock	Datum	Unterschrift
Finjas Künemann	Datum	Unterschrift
PG Beatrice Coldewey	Datum	Unterschrift
PG Eugen Lange	Datum	Unterschrift
PG Marius Leyh	Datum	Unterschrift
PG Jan-Frederik Scharnowski	Datum	Unterschrift
PG Maren Steinkamp	Datum	Unterschrift

PG Alexander Thomas	Datum	Unterschrift
PG Felix Van Der Ahe	Datum	Unterschrift
PG Patrick Warszewik	Datum	Unterschrift
PG Marlon Willms	Datum	Unterschrift

1. Kurzbeschreibung des Vorgangs

Als Trainingsdaten und als Gold-Standard für die Validierung der Annotationen sollen die Aktivitäten der Senioren im häuslichen Bereich manuell gelabelt werden.

Zu diesem Zweck besuchen die Projektgruppenteilnehmer (PG-Teilnehmer) die Probanden zuhause. Dort werden die über den Sensorgürtel aufgezeichneten Aktivitäten der Probanden mithilfe einer App dokumentiert und Informationen über die Gebäudegrundrisse erfasst.

2. Mitarbeiter, für die die VA verbindlich ist

alle Mitarbeiter AEQUIPA-Versa (Andrea Heinks, Elke Onken, Sandra Lau, Sebastian Fudickar, Sandra Hellmers, Finjas Künnemann)

alle Teilnehmer der Projektgruppe PGMAMKSFZ (Beatrice Coldewey, Eugen Lange, Marius Leyh, Jan-Frederik Scharnowski, Maren Steinkamp, Alexander Thomas, Felix Van Der Ahe, Patrick Warszewik, Marlon Willms)

3. Beschreibung des Vorgehens

Die Informationen über das Vorhaben sowie die Terminabsprache mit den Probanden erfolgt entsprechend VA-10. Der Besuch der Probanden erfolgt durch jeweils zwei/drei PG-Teilnehmer. Die PG-Teilnehmer begleiten die Probanden für ca. 2 - 4 Stunden bei Ihren Alltagsaktivitäten. Dabei sind die in den Kommentaren (siehe VA-10) vermerkten Besonderheiten (z. B. mit Fahrrad fahren) zu beachten.

Zur Vorbereitung der Besuche werden von den PG-Teilnehmern der verwendete Sensorgürtel (vollständig geladen bei 4,19 V, Messung kann gestartet werden ab 3,9V) und das Tablet aufgeladen und mit der SHID für die Messung konfiguriert. Eine Stunde vor Beginn des Besuchs werden die Probanden angerufen, um sicherzustellen, dass sie vor Ort sind.

In Notfällen (z.B. Proband ist nicht zu erreichen) kann unter folgenden Telefonnummern Kontakt zu den Projektmitverantwortlichen aufgenommen werden:

0441 798 4334 - Versa- Studie

0441 798 2310 - Büro Finjas Künemann

0441 798 2667 – Büro Sandra Hellmers

0441 798 2849 – Büro Sebastian Fudickar

0441 798 2614 – Versa - Studienraum

0441 798 2772 – Sekretariat Imke Garten (nach Sebastian, Sandra… fragen)

Benötigte Materialien für den Besuch:

- Sensorgürtel und Notebook mit Humotion Data Logger- Software zum Start des Sensorgürtels
- Uni-Tablet mit installierter mamksfz_activity_tracker-Applikation
- Kamera zum Abfotografieren des Grundrisses oder Papier/Stift zum Skizzieren der Raummaße (+ Türen, Treppen, Stufenhöhe), die mit einem Laserscanner vermessen werden
- Hausbesuch-Bogen

Probandenbesuch:

Die PG-Teilnehmer stellen sich den Probanden mit ihrem Studienausweis vor.

Zu Beginn des Besuchs wird dem Probanden (nochmals) das Vorgehen für den Zeitraum des Besuchs erklärt. Im Anschluss wird dem Probanden der Sensorgürtel angelegt und die Messung über den Humotion Data Logger gestartet. Das Labeln der Aktivitäten erfolgt durch einen der beiden PG-Teilnehmer mittels der mamksfz_activity_tracker-Applikation. Als Identifier wird der Senior-Home-Identifier (SHID) verwendet. Es ist darauf zu achten, dass sowohl das Tablet als auch der Rechner mit dem gleichen Zeitserver synchronisiert werden.

Für das Labelverfahren wurden für eine einheitliche Durchführung bestimmte Qualitätsstandards festgelegt, welche sich auf die in der Activity-Tracker-App befindlichen 11 Aktivitäten begrenzt. Außerdem gibt es noch für diverse andere Szenarien das Feld „Sonstiges“, welches mit einem beliebigen Kommentar versehen werden kann. Sämtliche Aktivitäten die in der App angezeigt werden (gehen, stehen, liegen, sitzen, aufstehen, …) sollen erfasst werden. Aktivitäten die nicht in der App vertreten sind, z.B. Fahrrad fahren, werden unter „Sonstiges“ mit einem zusätzlichen Kommentar wie „Fahrrad fahren“ gelabelt. Zusätzlich gelten bei den Aktivitäten bestimmte Kriterien die erfüllt sein müssen. Insofern erkennbar sollen Aktivitäten erst getrackt werden wenn gilt:

- Treppe herauf- und heruntersteigen ab zwei Stufen
- Drehung ab 90°

- Gehen (auch Rückwärts) ab drei Schritten. (Ausweichschritte, wie z.B. ein Schritt zur Seite, nicht tracken)
- Undefinierte Bereiche auslassen
- Stehen, Liegen, Sitzen ab zwei Sekunden (Bzw. wenn erkannt werden kann, dass die Aktivität nicht vorzeitig abgebrochen wird)
- Hocken bei unmittelbarer Ausführung
- Transitionen sind schwer mit der App zu tracken. Diese, wenn möglich und mit der App geübt, tracken. Ansonsten beim übertragen in MAMKS ergänzen.
- Alle „Sonstiges“ Aktivitäten, können nach eigenem Ermessen erfasst werden. Fahrradfahren, Busfahren, Toilettengang sollten getrackt werden.

Ist kein Grundriss vorhanden, der abfotografiert werden kann oder von den Probanden kopiert wurde, wird während des Besuches zusammen mit den Probanden die einzelnen Räume vermessen und ein Grundriss manuell erstellt. Ist ein Grundriss vorhanden, wird nur ein Raum zur Kontrolle der angegebene Werte vermessen bzw. fehlende Werte nachgetragen. Ist kein Grundriss vorhanden, werden sämtliche Räume ebenso wie die Position der Türen/Durchgänge vor Ort vermessen. Sowohl in den manuellen Grundriss, als auch den möglicherweise vorhandenen Grundriss, sind alle Möbel einzutragen, die Einfluss auf die Bewegungsfreiheit in den Räumen haben (z.B. Tische, Schränke, Betten, Sofas, Sanitäranlagen, Stühle und besondere große Objekte). Nach Erstellung bzw. Überarbeitung des Grundrisses wird der Hausbesuch-Bogen ausgefüllt.

Im Hausbesuch-Bogen ist einzutragen:

- Name des verantwortlichen Vermessers.
- Die SHID des/der Probanden.
- Das Datum am Tag der Vermessung.
- Die Anzahl von Treppenstufen (Fußboden der oberen Etage zählt als Stufe; Fußboden der unteren Etage zählt nicht als Stufe).
- Die Höhe der Treppenstufen.
- Der Verlauf der Treppe (von der untersten Stufe aus gesehen steigt die Treppe: im/ gegen den Uhrzeigersinn oder gradlinig).
- Ggf. die Anzahl der Stufen im Treppenhaus vor der Wohnung (Hauseingang bis Wohnungstür).
- Ggf. die Anzahl der Treppenstufen vor dem Haus.
- Ob der Grundriss manuell erstellt wurde.
- Ob min. 1 mal Länge und Breite jedes Raumes im Grundriss eingetragen wurde.
- Ob die Richtung der Türöffnungen korrekt eingetragen wurde.
- Ob alle Möbel, die die Bewegungsfreiheit einschränken, eingetragen wurden.
- Ggf. Dachschrägen vorhanden sind.
- Ggf. alle Bereiche im Grundriss gekennzeichnet wurden, in denen die Raumhöhe <150cm beträgt.
- Ggf. eine Garage vorhanden ist.

- Sonstiges das für wichtig erachtet wird.

Am Ende des Besuchs wird die Messung des Sensorgürtels beendet und der Sensorgürtel wieder mitgenommen. Gff. werden auch die Unterlagen der 7-tägigen Heimdatenmessung mitgenommen und zeitnah bei Finjas Künemann oder Sandra Hellmers abgegeben. Dazu gehört neben dem Gürtel das Ladekabel mit Stecker, eine einfolierte Anleitung, das ausgefüllte Tagebuch und ein Fragebogen sowie der Pappkarton in dem sich die ganzen genannten Unterlagen befinden.

Nach dem Probandenbesuch werden die Daten vom Gürtel übertagen:

a) DataLogger-Software öffnen
b) Gürtel anschließen → Gürtel wird automatisch erkannt und überträgt Daten.
c) Wenn fertig, auf „Exportieren“ klicken.
d) Es öffnet sich der Export Monitor
e) Dort auf „neue Datei suchen“
f) Gewünschte Datei auswählen und auf „ausgewählte Dateien exportieren“ drücken
g) Die Dateien werden unter **\\daten.uni-oldenburg\pgimu$\PG2017\/Daten\SeniorHome-Studie\Rohdaten** unter der Senior-Home-ID abgespeichert.

Übertagung der Daten vom Tablet:

a) Tablet anschließen
b) Daten befinden sich auf dem Tablet unter **InternerSpeicher/Android/data/com.example.maren.activitytracker/files**
c) Die gewünschte Dateien aus den Ordnern ActivityData und LocationData werden unter **\\daten.uni-oldenburg\pgimu$\PG2017\Daten\SeniorHome-Studie\Rohdaten** unter der Senior-Home-ID abgespeichert.

4. **Dokumentation**
   \\daten.uni-oldenburg\pgimu$\PG2017\/Daten\SeniorHome-Studie
5. **Verteiler**
6. **mitgeltende Unterlagen**

B. Seminararbeiten

Nachfolgend finden sich die von den einzelnen Projektteilnehmern ausgearbeiteten Seminararbeiten.

B.1. Körpernahe Sensoren mittels IMU

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Abteilung Assistenzsysteme und Medizintechnik

Projektgruppe Mobilitäts-Assessments
mit körpernahen Sensoren für zuhause
SoSe17 - WiSe17/18

Seminararbeit: Körpernahe Sensoren mittels IMU

Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autoren: Felix Van Der Ahe

12. März 2018

Zusammenfassung

In dieser Seminararbeit wird auf die Sensorik, in des enthaltenen Humotion Sensorgürtels eingegangen, um die Grundlagen der Sensoren nahe zu bringen. Weitere mögliche Optimierungen für das Messen und Auswerten der Daten am Gürtel werden erörtert, insbesondere für die Beschleunigung in Kombination mit Gyroskop und Magnetometer. Zur Messung beim Eintreten bzw. Verlassen vom Außen und Innenbereich in häuslicher Umgebung wird der verbaute Luftdrucksensor näher erläutert und mögliche Szenarien für die Erkennung von Aktivitäten gezeigt.

Inhaltsverzeichnis

1 Einleitung

Mit dem Alter nimmt die Mobilität bei älteren Menschen stetig ab. Dabei bezeichnet Mobilität die Möglichkeiten und Erfüllung von Bewegungen z.B. Einkaufen oder zum Arzt gehen [10]. Gegenwärtig sinkt nicht nur die Lebensqualität, sondern es können auch Herzerkrankungen durch mangelnde Bewegung die Folge sein [24]. Somit kann der Besuch zum Arzt durch Mobilitätseinschränkungen erschwert werden und damit eine Diagnose oder Versorgung nur bedingt erfüllt werden. Physiotherapeuten oder Ärzte müssten somit persönlich zu ihren Patienten kommen oder in regelmäßigen Abständen eine Übersicht über körperliche Aktivitäten ihrer Patienten verfügen. Um solche körperlichen Bewegungen aufzuzeichnen, gibt es heutzutage verschiedene Sensoren, die als kleine Baufabrikate z.B. in Smartphones, Fitness Trackern oder Gürteln verbaut sind. Solche Inertialsensoren kurz IMU (Inertial Measurement Unit), ist eine Kombinaton von verschiedenen Sensoren, wie Beschleunigungssensor oder Drehratensensor [22]. Dadurch ist es möglich, mittels verschiedener Algorithmen und spezieller Software, bestimmte Aktivitäten zu erfassen. In der Projektgruppe 2017 werden solche körpernahen Sensoren zur Analyse von Mobilitäts-Assessments für zu Hause eingesetzt. Damit sollen mögliche Einschränkungen und Abläufe der Bewegung aufgezeigt werden, die Physiotherapeuten oder Ärzte bei ihrer Arbeit unterstützen können.

Im weiteren Verlauf werden die einzelnen Sensoren zum Humotion Sensorgürtel erläutert, sowie Probleme und Maßnahmen behandelt, die zur Unterstützung dieser Projektgruppe beisteuern sollen.

2 Orientierungs- und Positionsbestimmung

Für die Orientierungsbestimmung wird der Beschleunigungssensor, sowie Magnetometer und Gyroskop verwendet. Die Zuverlässigkeit der Messung hängt neben den Sensoren auch von den Algorithmen sowie Filterung der Daten ab. In der Literatur wird für die Orientierungsbestimmung meist Quaternionen verwendet [14].

2.1 Quaternionen

Sie sind eine Erweiterung der reellen Zahlen und spannen einen vierdimensionalen Vektorraum auf, mit dem sich Drehungen von Körpern beschreiben lassen. Eine Quaternion besteht aus drei imaginären Zahlenkomponenten i,j und k (2.1) . Und einem dreidimensionalen Vektor Imaginärteil (2.2),(2.3). Mithilfe der komplexen Zahlen ähnlich mit $i^2 = j^2 = k^2 = i * j * k = -1$ lässt sich eine gleichwertige Schreibweise notieren [13]:

$$q = w + ix + jy + kz, wobei\, w, x, y, z \in \mathbf{R} \tag{2.1}$$

$$[w, (x, y, z)], wobei\, w, x, y, z \in \mathbf{R} \tag{2.2}$$

$$[w, v], wobei\, w \in \mathbf{R}, v \in \mathbf{R^3} \tag{2.3}$$

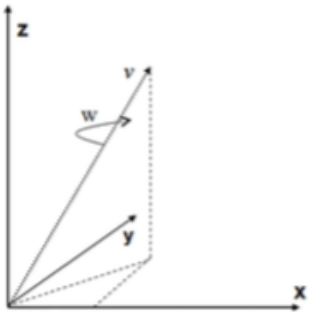


Abbildung 2.1: Quaternionen als Kombination aus Drehachse und Drehwinkel [13]

2.2 Eulerwinkel

Eine weitere Variante zur Orientierungsbestimmung bietet der Eulerische Winkel kurz Eulerwinkel genannt. Der Eulerwinkel stellt eine Position durch drei Winkel ϕ, θ, ψ dar. die jeweils um eine bestimmte Achse rotieren.

ψ Rotation um die ursprüngliche z-Achse
θ Rotation um die neue x-Achse
ϕ Rotation um die neue z-Achse
Dadurch ergibt sich die Rotationsmatrix:

$$R = \begin{pmatrix} cos\psi\, cos\phi - sin\psi\, cos\theta\, sin\phi & -cos\psi\, sin\phi - sin\psi\, cos\theta cos\phi & sin\psi\, sin\theta \\ sin\psi\, cos\phi + cos\psi\, cos\theta\, sin\phi & cos\psi\, cos\theta\, cos\phi - sin\psi\, sin\phi & -cos\psi\, sin\theta \\ sin\theta\, sin\phi & sin\theta\, cos\phi & cos\theta \end{pmatrix}$$

Dabei ist zu beachten, dass drei Koordinationsrotationen pro Drehung notwendig sind [23].

Ein weiterer Nachteil gegenüber Quanternion ist der sogenannte Gimbal Lock. Dieses Phänomen tritt auf, wenn bei der Rotation um die Achse, mit einem Winkel von 90°, eine zweite Achse überlagert wird. [6].

Quanternionen sind mit ihrer Darstellung von Drehachse und Drehwinkel in der Anwendung intuitiver. Zugleich ist der rechnerische Aufwand einer Rotation gegenüber Eulerwinkeln geringer. [6] [13].

2.3 Positionsbestimmung

Im Zusammenhang der Aktivitätserkennung muss berücksichtigt werden, dass die Achsenbeschleunigungen, je nach Richtung, von der Orientierung des Gürtels abhängen. Die Abbildung 3.2 zeigt dabei Richtungsinformationen des Humotion Sensorgürtels. Wird der Gürtel an einer falschen Position getragen, verändert sich demnach die Richtung der Achse von der resultierenden Bewegung. Möchte man keine feste Position festlegen, so ist die Verwendung der Gesamtbeschleunigung (Magnitude of Acceleration) anstelle der einzelnen Achsen eine effiziente Lösung: [7] [17][26]

$$SMA : \sqrt{x^2 + y^2 + z^2}$$

3 Beschleunigungssensor

3.1 Aufbau

Bei dem Beschleunigungssensor oder auch Accelerometer genannt, handelt es sich um einen Sensor, der die auf ihn einwirkende Beschleunigung misst. Dabei basiert die Messung auf dem Hooke'schen Gesetz. Das heißt, die einwirkende Kraft steht proportional zur Längenänderung. Die Abbildung 3.1 zeigt den mechanischen Aufbau eines Beschleunigungssensors. Es enthält eine Prüfmasse, die von außen beeinträchtigte Kräfte, mithilfe der Feder in Bewegung gebracht wird. Bei dem Hu-

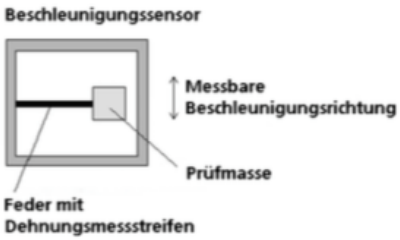


Abbildung 3.1: Zeigt den Aufbau eines Beschleunigungssensors [4]

motion Sensorgürtel ist u.a. ein dreiachsiger Beschleunigungssensor verbaut, der für jede Richtung (X,Y,Z) die Beschleunigung misst siehe 3.2. Bei der Messung muss die Erdbeschleunigung berücksichtigt werden, die je nach Lage des Sensors von $1g = 9,81m/^2$ misst. Bei einer senkrechten Haltung des Probanden wirkt die Erdbeschleunigung auf der Y-Achse senkrecht nach oben. Um die relative Beschleunigung des Sensorgürtels zu verarbeiten, muss die Erdbeschleunigung vorher abgezogen werden.

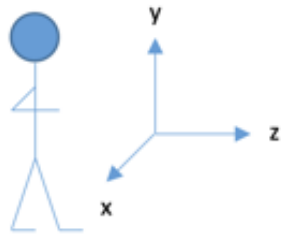
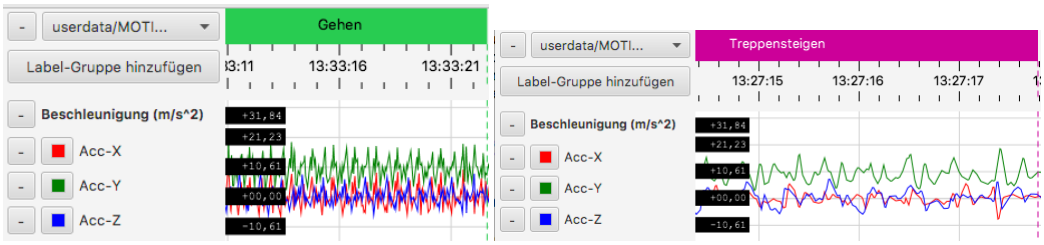


Abbildung 3.2: Zeigt die Richtung der Achsen beim Humotionsensorgürtel

3.2 Verarbeitung des Beschleunigungssensors

Im Kontext der Aktivitätserkennung sind die Beschleunigungswerte Abhängig von der Zeit. Bei einer Messung können die Werte jedoch von einem Rauschen überlagert werden. Damit die Daten verarbeitet werden können, müssen sie hinsichtlich der Zeit (sampling rate) und des Wertebereichs, abgetastet werden. Die Abtastrate gibt die Häufigkeit des Signals an, die abgetastet wird. Dabei wird eine Abfolge von Messwerten, auch Zeitreihe genannt, erfasst. Die grundlegende Einheit ist das Hertz (kurz Hz). Eine Abtastrate von beispielsweise 100Hz entspricht 100 Abtastungen pro Sekunde [18]. In der Regel ist eine Abtastrate für Aktivitäten, wie das gehen, stehen, liegen oder sitzen von 100Hz ausreichend. Erst bei größeren Bewegungen, wie zum Beispiel das Springen, wäre eine höhere Abtastrate >100Hz nötig [4].
Weitere Verarbeitungen bei der Digitalisierung ist das Abtastintervall, auch Frames genannt. Sie bezeichnen eine Anzahl von Samples, die aus einem vollständigen Satz von Sensormesswerten bestehen. Folgend aus den Beschleunigungswerten, der x,y und z Richtung. Häufig wird eine Fenstergröße von 1 - 5 Sekunden gewählt. Damit soll sichergestellt werden, das die Klassifikation nicht zu lange braucht, um neue Bewegungen zu erkennen [11].



(a) Beschleunigungssensor: Aktivität gehen (b) Beschleunigungssensor: Aktivität Treppen steigen

Abbildung 3.3: Zeigt die Aktivität Treppen steigen und gehen in der MAMKS Software

Die Abbildung 3.3 zeigt einen Ausschnitt aus der MAMKS Software, bei der Verwendung des Beschleunigungssensor anhand der Bewegung *gehen*. Verglichen mit der Aktivität in Abbildung 3.3, so sind deutliche Unterschiede des Sensors zu erkennen. Dies zeigt, das jede Aktivität unterschiedliche Muster besitzt, die durch sogenannte Feature Merkmale erkannt werden können.

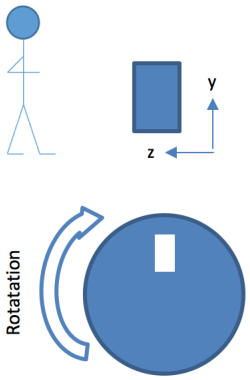


Abbildung 3.4: Funktionsweise Drehratensensor

4 Gyroskop

Bei dem Gyroskop oder auch Drehratensensor handelt es sich um einen Sensor, der die Rotationsgeschwindigkeit um die drei Achsen (X,Y,Z) misst. In der Abbildung 3.4 ist die Funktionsweise exemplarisch aufgeführt. Ein Ausschnitt des Sensors oben, zeigt die Richtungen der Achsen.Durch die Rotation wird die Corioliskraft erzeugt. Je nach dem, wie sich der Körper bewegt, spiegeln sich diese Werte in den Achsen wieder. Ein bekanntes Problem bei dem Gyroskop ist der sogenannte *Drift*. Bei

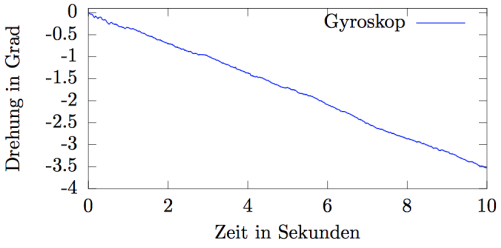


Abbildung 4.1: Drift beim Gyroskop [8]

der Messung der Drehgeschwindigkeit, wird diese über die Zeit integriert. Daher entsteht eine andauernde Drehung. Die Abbildung zeigt eine Rotation über die y-Achse, während der Gegenstand steht. Ein leichter *Drift* ist hier ebenfalls zu erkennen [8]. Abhilfe schaffen dabei sogenannte Filter oder durch komplementäre Sensorik, wie Magnetometer und Beschleunigungssensor [3].

5 Magnetometer

Das Magnetometer unterscheidet sich hingegen wesentlich zu den anderen Sensoren. Die Basis bildet das Erdmagnetfeld im inneren der Erde. Diese Magnetfelder können demnach in der Richtung, in der sie wirken, gemessen werden. Um das Magnetfeld auf einem bestimmten Punkt auf der Erdoberfläche zu beschreiben eignet sich die Definition seiner 3 räumlichen Komponenten. In Abbildung 5.1 zeigen die Achsen in die jeweils geographischen Richtungen. Die Deklination beschreibt dabei den Winkel, der Abweichung zwischen horizontale Feldanteil (H) und der geographischen Nordrichtung. Die Inklination zeigt den Winkel einer Magnetfeldlinie gegenüber der Horizontalen. Das magnetische Feld wird dabei in Tesla (T) gemessen [19].

- X-Komponente: Feldanteil in geografischer Nordrichtung (Süd nach Nord weisend)
- Y-Komponente: Feldanteil in geografischer Ostrichtung (West nach Ost weisend)
- Z-Komponente: Feldanteil in Richtung Nadir (Erdmittelpunkt)

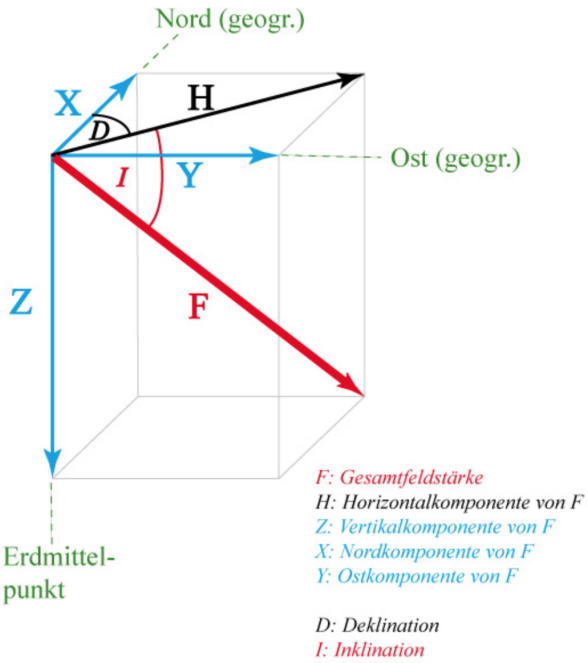
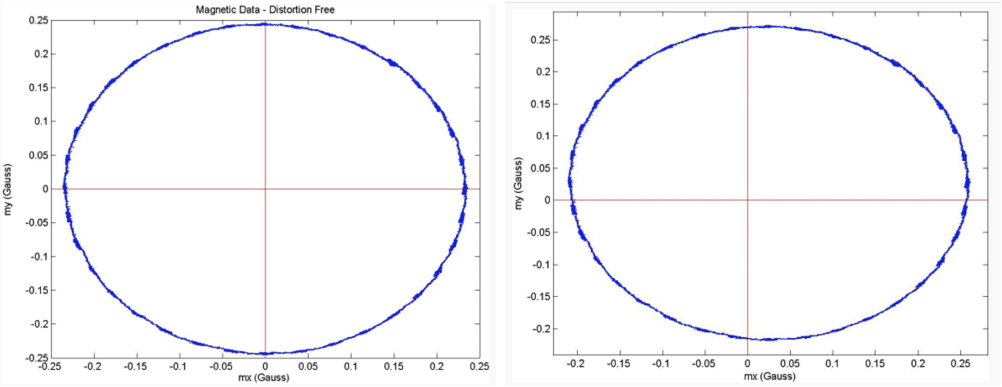


Abbildung 5.1: Beschreibung der magnetischen Feldstärke seiner 3 räumlichen Kompenten [19]

5.1 Hard-Iron Effekt

Anders als bei dem Gyroskop oder der Beschleunigung, gibt es beim Magnetometer den sogenannten *Hard-Iron Effect*. Das heißt, magnetische Komponenten die in Berührung mit dem Magnetometer kommen und das magnetische Feld so stören. In der Abbildung 5.2 wurde das Magnetometer um 360 Grad gedreht, welches einen Kreis der X und Y-Achse über (0,0) zeigt. Bei Annäherung von magnetischen Komponenten entsteht der *Hard-Iron Effect*, wie in der Grafik zu entnehmen.



(a) 360 Grad Rotation Magnetometer (ohne Verzerrung)

(b) 360 Grad Rotation Magnetometer (mit Verzerrung)

Abbildung 5.2: Magnetometer Daten - Vergleich mit und ohne Verzerrung [5]

Diese Verzerrung kann mit einer simplen Formel korrigiert werden. Dabei wird das Magnetometer um 360 Grad gedreht, wobei der Abstand von (0,0) zur Mitte des Kreises bestimmte wird, indem der Mittelwert, der Maximal- und Minimalwert für jeder der Achsen bestimmt wird [5] [9].

$$\alpha = \frac{x_{max} + x_{min}}{2} \tag{5.1}$$

$$\beta = \frac{y_{max} + y_{min}}{2} \tag{5.2}$$

wobei,

α = x-Achse Offset

β = y-Achse Offset

x_{max} = Maximalwert x

x_{min} = Minimalwert x

y_{max} = Maximalwert y

y_{min} = Minimalwert y

6 Filter

Zum jetzigen Stand werden die drei oben beschriebenen Sensoren vom Humotion Sensorgürtel zur Messung eingesetzt und ausgewertet. Wie oben beschrieben, sind die Rohdaten bei der Beschleunigung und beim Gyroskop meistens mit einem Rauschen überlagert. Abhilfe schaffen dabei Filter, wie der *Lowpass-Filter* bzw. *Highpass-Filter*. Sie glätten dabei die Sensordaten mit unterschiedlichen Frequenzen. Wird nur ein Beschleunigungssensor benutzt so kann der Tiefpassfilter genutzt werden, um die niederfrequenten Signale zu isolieren. Die Frequenz muss beim Filter so gesetzt werden, dass keine relevanten Frequenzanteile für die untersuchende Aktivität eliminiert werden. In der Regel sollte dabei eine Grenzfrequenz von bis zu 0,5 Hz verwendet werden. [1] [2] [12] [15]

6.1 Komplementärfilter

Der Komplementärfilter ist ein einfacher Weg, um die Messwerte des Beschleunigungssensors, des Gyroskops und optional den Magnetometer zu vereinen. Es besteht aus einem gemeinsamen Tiefpassfilter für die Beschleunigung, und einen Hochpassfilter für das Gyroskop.

$$\theta_n = \alpha \times (\theta_{n-1} + \omega \times \varrho\tau) + (1.0 - \alpha) \times \alpha \tag{6.1}$$

Mit α als Gewichtskonstante, a für Beschleunigung, ω für die Winkelgeschwindigkeit des Gyroskops und $\varrho\tau$ die Zeit, die zwischen den Messungen ist. Im unten stehenden Quellcode wurde eine manuelle Berechnung mit der Orientierung des Euler Winkels implementiert, der einen Komplementärfilter beinhaltet. Dabei wurde als

Verzerrung der Wert 0,98 gesetzt, was den üblichen Wert für alpha entspricht. Das bedeutet das 98 Prozent des Gewichts auf den Gyroskop Messungen liegen [21].

```
const options = { frequency: 50 };

const accl = new Accelerometer(options);
const gyro = new Gyroscope(options);

let timestamp = null;
let alpha = beta = gamma = 0;
const bias = 0.98;

gyro.onreading = () => {
let dt = timestamp ? (gyro.timestamp - timestamp) / 1000 : 0;
timestamp = gyro.timestamp;

// Treat the acceleration vector as an orientation vector by
    normalizing it.
// Keep in mind that the if the device is flipped, the vector will
    just be
// pointing in the other direction, so we have no way to know from the
// accelerometer data which way the device is oriented.
const norm = Math.sqrt(accl.x ** 2 + accl.y ** 2 + accl.z ** 2);

// As we only can cover half (PI rad) of the full spectrum (2*PI rad)
    we multiply
// the unit vector with values from [-1, 1] with PI/2, covering
    [-PI/2, PI/2].
const scale = Math.PI / 2;

alpha = alpha + gyro.z * dt;
beta = bias * (beta + gyro.x * dt) + (1.0 - bias) * (accl.x * scale /
```

```
    norm);
gamma = bias * (gamma + gyro.y * dt) + (1.0 - bias) * (accl.y *
    -scale / norm);

// Do something with Euler angles (alpha, beta, gamma).
};

accl.start();
gyro.start();
```

7 Luftdrucksensor

Ein weiterer Sensor soll in der PG Gruppe 2017 voraussichtlich eingesetzt werden, um die Zustände für den Innen und Außen Bereich zu erkennen. Das heißt, wenn der Proband das Haus verlässt bzw. wieder betritt.
Als Luftdrucksensor wird der Sensor BMP085 der Firma Bosch im Humotion Sensorgürtel verwendet [20]. Der Sensor liest den Luftdruck in 1Pa (=0.01hPa) und die Temperatur in 0.1 Celsius. Die absolute Höhe kann mit der internationalen Formel 7.1 berechnet werden. Für die Berechnung des Meeresspiegels gilt die Formel7.2.

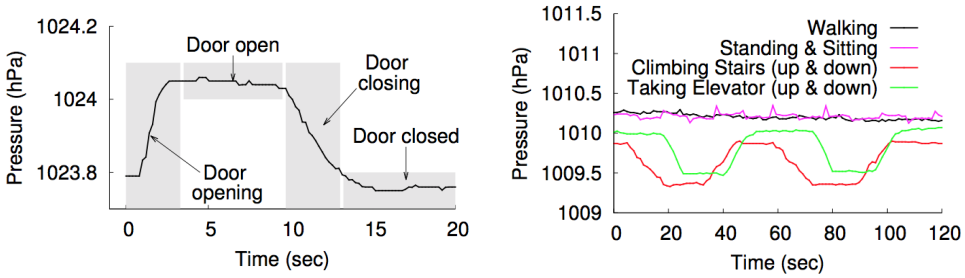
$$altitude = 44330 * (1 - (\frac{p}{p_0})^{\frac{1}{5.255}}) \tag{7.1}$$

$$p_0 = \frac{p}{(1 - \frac{altitude}{44330})^{5.255}} \tag{7.2}$$

mit p für den Luftdruck und p_0 für den Meeresspiegel.

7.1 Related Work

In 2014 wurde eine Studie veröffentlicht, die den Luftdruck mit mehreren Smartphones in unterschiedlichen Gebäuden getestet hat. Dabei sollte gezeigt werden, wie der Luftdrucksensor benutzt werden kann, um Änderungen der Etagen in Gebäuden wahrzunehmen. Zum Beispiel, wenn der Proband verschiedene Etagen durchquert. Die Fehleranfälligkeit bzw. die Luftdruck Abweichungen lagen dabei um 0.2hPa (für die gleichen Gebäude und Etagen über mehrere Tage in der gleichen Zeit). Übersetzt ist das eine Abweichung von 1,6 Meter. So lange die Entfernung zwischen den Etagen min. 1,6m betrifft, kann der Luftdrucksensor für die Erkennung von Änderungen der Etagen genutzt werden [16].



(a) Luftdruckdaten beim öffnen/schließen einer Tür

(b) Luftdruck Unterschiede, bei unterschiedlichen Aktivitäten

Abbildung 7.1: Zeigt die Luftdrucksensordaten einer Studie bei unterschiedlichen Aktivitäten [25]

In einer anderen Studie wurden ebenfalls mehrere Smartphones als Luftdrucksensoren genutzt, um verschiedene Ereignisse, wie das öffnen von Türen, zu ermitteln. Die Voraussetzung war allerdings eine *HVAC (Heating, Ventilating, and Air Conditioning)* in den jeweiligen Gebäuden, da laut dieser Studie solch ein HVAC ein komfortables System ist, um die Veränderungen zwischen dem Innen und Außenbereich zu erkennen. Zumal seien diese Systeme in dem amerikanischen Raum groß vertreten, was hier zu lande nicht der Fall ist. Der Luftdruck wurde mit 0.01 hPa und einer Abtastfrequenz von 20Hz gemessen. Die Abbildung 7.1 zeigt das Ereignis wie eine Tür geöffnet bzw. geschlossen wird. Je nach Aktivität des Probanden dürfte es zu einer Druckänderung kommen, siehe 7.1.

7.1.1 Feature Selection

Um eine möglichst genaue Erkennung der *open/close Events* zu bekommen wurden drei Feature Merkmale angelernt. Außerdem wurde eine Fenstergröße von 3 Sekunden festgelegt.

- Rate of change: Das Merkmal zeigt die Änderungsrate, bei der sich der Luft-

druck im jeweiligen Zeitfenster ändert.

- Mean-Crossing: Um plötzliche Druckschwankungen festzustellen wird die Anzahl der beobachteten *pressure crosses* (der Mittelwert des Luftdrucks im Zeitfenster) berechnet.

- Standard-deviation: Die Standard Varianz, um die allgemeinen Abweichungen vom Durchschnittswert des Luftdrucks zu erhalten.

Diese Daten wurden mit dem *Naive Bayes* Klassifikator angelernt. Die Erkennung für das öffnen von Türen lag bei 99,34 Prozent [25].

Um einen ersten Vergleich zu ziehen, wurde der Humotion Gürtel und ein Smartphone (iPhone 7) als Luftdrucksensor verwendet, um Änderungen des Luftdrucks bei verschiedenen Ereignissen zu zeigen. Auf dem Smartphone wurde die App *Barometer* verwendet. Die mobile App von Matlab, um die Rohdaten direkt zu exportieren, konnte nicht verwendet werden, da der Luftdrucksensor dort nicht enthalten war. Einer der Aktivitäten war das Treppen steigen, um einen möglichen Etagenwechsel zu demonstrieren. Der Startpunkt lag dabei im 4. Stock eines Gebäudes. Von da aus lief die Person 4 Etagen abwärts und wieder aufwärts mit je ca. 15 Stufen. In der Abbildung 7.2 wurden die Luftdruckdaten um -1 versetzt, um den Unterschied des Filters besser zu erkennen. Auf den Daten wurde ein Medianfilter angewendet, zu erkennen an der orangen Linie. Die höheren Peaks in den Rohdaten demonstrieren einen Handschlag auf die hinteren Sensoren. Zu diesem Zeitpunkt war eine Etage jeweils beendet und es wurde eine Stichprobe zur absoluten Höhe entnommen. Weitere Feature Merkmale wurde nicht auf die Daten angewandt.

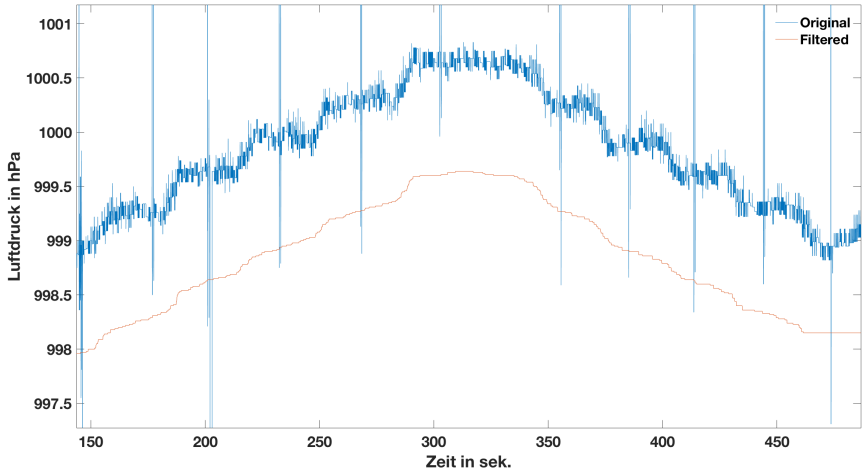
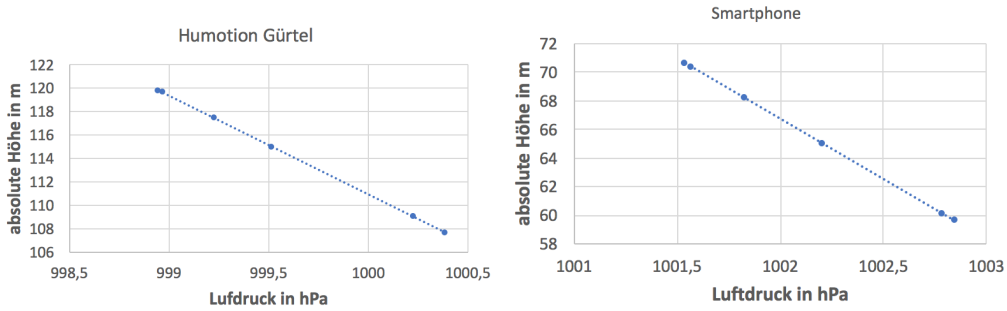


Abbildung 7.2: Luftdrucksensordaten vom Humotion Gürtel: Treppen steigen auf/abwärts

Verglichen mit dem Smartphone, errechnet der Humotion Gürtel laut der Formel 7.1 eine absolute Höhe im Durchschnitt von 114,7m, bei einem Meeresspiegel von 1013,25hPa. Bei dem Smartphone sind es dagegen 65,6m. Um einen Vergleich zu erhalten, wurden mehrere Stichproben während des Tests, im Bezug auf die absolute Höhe entnommen. In dem Streudiagramm 7.3 ist ein deutlicher Trend zu sehen, der zeigt, dass trotz der Unterschiede die absolute Höhe sich konstant bei beiden Geräten gleich verändert.



(a) Humotion Gürtel

(b) Smartphone

Abbildung 7.3: Verhältnis der absoluten Höhe beim Treppen steigen mit dem Humotion Gürtel und einem Smartphone.

Eine weitere Erkennung bezog sich auf die Luftdruckdaten beim Verlassen bzw. Betreten des Außen und Innenbereichs eines Gebäudes. Dabei befand sich die Zielperson in der 2. Etage eines Gebäudes. Um einen möglichen Wechsel in den Außenbereich zu erkennen, wurde mehrmals der Balkon betreten. Dabei wurden allerdings keine resultierende Erkenntnisse gefunden- siehe Abbildung 7.4.

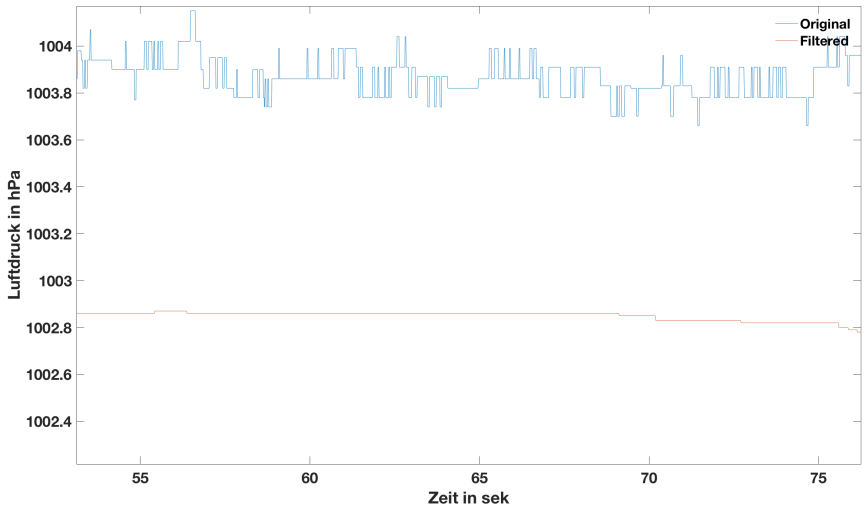
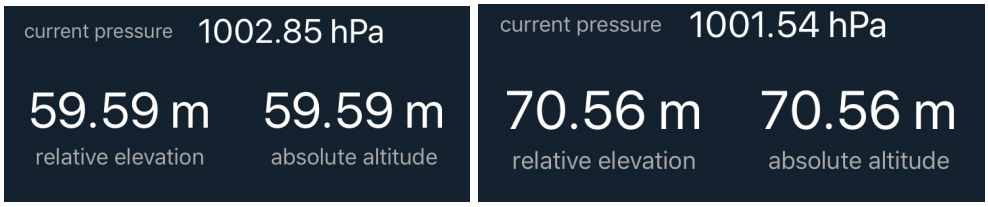


Abbildung 7.4: Luftdrucksensordaten vom Humotion Gürtel: Wechseln im Innen und Außenbereich



(a) Erdgeschoss (b) 1. Etage

Abbildung 7.5: Vergleich des Luftdrucks mit dem Smartphone im Erdgeschoss/ 4. Etage

8 Fazit

In dieser Seminararbeit sollten die Grundlagen der Sensorik, des enthaltenen Humotion Sensorgürtels vermittelt werden und mögliche Verbesserungen zur Analyse bzw. Verarbeitung bei der Erkennung von Aktivitäten in der Projektgruppe 2017 ermöglichen. Ein möglicher Ansatz wäre die Implementierung des Komplementärfilters in Matlab, zur Sensorfusion von Beschleunigung und Winkelgeschwindigkeit. Zugleich sollte der Luftdrucksensor im primären Fokus stehen, da dieser Sensor bis dato noch nicht zum Einsatz der Projektgruppe kam.

Als Rückschluss für den Luftdrucksensor lässt sich sagen, das sich dieser in einem ersten Test positiv für die Aktivität Treppen steigen bewährt hat. Allerdings ist es schwer abzuschätzen, in welchem Stockwerk derjenige sich befindet oder wie viel Stufen getätigt wurden.

Kein Ergebnis brachte der Versuch, anhand des Luftdrucks Veränderungen im Innen und Außenbereichs eines Gebäudes zu erkennen. Eine mögliche Variante wäre die Einbringung des Temperatursensors. Dennoch können die Feature Merkmale wie oben beschrieben, eine Interessante Möglichkeit sein, diese auch in Matlab zu implementieren sowie Filter für Gyroskop und Beschleunigungssensor, um ein präziseres Ergebnis zu erzielen.

Literatur

[1] Felicity R Allen u. a. „An adapted gaussian mixture model approach to accelerometry-based movement classification using time-domain features“. In: *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*. IEEE. 2006, S. 3600–3603.

[2] Jonghun Baek u. a. „Recognition of user activity for user interface on a mobile device“. In: *Procs. of the 24th South East Asia Regional Computer Conference. Thailand*. 2007.

[3] Matthias Bethge und Dipl-Ing Nils Gageik. „Optimierung einer magnetischen Orientierungskompensation durch Magnetfeldfehlererkennung“. In: ().

[4] Gerald Bieber. *Methodik zur mobilen Erfassung körperlicher Aktivität mittels Beschleunigungssensoren*.

[5] Michael J Caruso. „Applications of magnetic sensors for low cost compass systems“. In: *Position Location and Navigation Symposium, IEEE 2000*. IEEE. 2000, S. 177–184.

[6] James Diebel. „Representing attitude: Euler angles, unit quaternions, and rotation vectors“. In: *Matrix* 58.15-16 (2006), S. 1–35.

[7] Davide Figo u. a. „Preprocessing techniques for context recognition from accelerometer data“. In: *Personal and Ubiquitous Computing* 14.7 (2010), S. 645–662.

[8] Simon Gene Gottlieb. „Erweiterter Kalman-Filter zur Orientierungsabschätzung von humanoiden Robotern“. In: *Bachelor Arbeit, Freie Universität Berlin* (2013).

[9] HR Harrison. „Quaternions and Rotation Sequences: a Primer with Applications to Orbits, Aerospace and Virtual Reality, KuipersJ. B., Princeton University Press, 41 William Street, Princeton, NJ 08540, USA. 1999. 372pp. Illustrated.£ 35.00. ISBN 0-691-05872-5.“ In: *The Aeronautical Journal* 103.1021 (1999), S. 175–175.

[10] Tomas Hefter und Konrad Götz. „Mobilität älterer Menschen“. In: *State of the Art und Schlussfolgerungen für das Projekt COMPAGNO. Frankfurt am Main.= ISOE-Diskussionspapiere* 36 (2013).

[11] Samuli Hemminki, Petteri Nurmi und Sasu Tarkoma. „Accelerometer-based transportation mode detection on smartphones“. In: *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. ACM. 2013, S. 13.

[12] Dean M Karantonis u. a. „Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring“. In: *IEEE transactions on information technology in biomedicine* 10.1 (2006), S. 156–167.

[13] Edgar Kraft. „Ein Sensorsystem zur Bestimmung räumlicher Orientierung in Echtzeit“. Diss. Diplomarbeit, Physikalisches Institut Universität Bonn, BONNIB-2002-13, 2002.

[14] N Lovren und JK Pieper. „Error analysis of direction cosines and quaternion parameters techniques for aircraft attitude determination“. In: *IEEE Transactions on aerospace and electronic systems* 34.3 (1998), S. 983–989.

[15] Merryn J Mathie u. a. „Accelerometry: providing an integrated, practical method for long-term, ambulatory monitoring of human movement“. In: *Physiological measurement* 25.2 (2004), R1.

[16] Kartik Muralidharan u. a. „Barometric phone sensors: More hype than hope!“ In: *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*. ACM. 2014, S. 12.

[17] Ben Nham, Kanya Siangliulue und Serena Yeung. „Predicting mode of transport from iphone accelerometer data“. In: *Standford University Class Project* (2008).

[18] Alan V Oppenheim und Ronald W Schafer. *Zeitdiskrete Signalverarbeitung*. Walter de Gruyter GmbH & Co KG, 1998.

[19] Valérie Renaudin, Muhammad Haris Afzal und Gérard Lachapelle. „Complete triaxis magnetometer calibration in the magnetic domain“. In: *Journal of sensors* 2010 (2010).

[20] BOSCH Sensortec. *BMP085 digital pressure sensor data sheet*. 2013.

[21] Kenneth Rohde Christiansen Alexander Shalamov. *Motion Sensors Explainer*. 2017.

[22] David Titterton und John L Weston. *Strapdown inertial navigation technology*. Bd. 17. IET, 2004.

[23] Klaus Wittmann, Wilfried Ley und Willi Hallmann. *Handbuch der Raumfahrttechnik*. Carl Hanser Verlag, 2008.

[24] C.G. Wollmann. „Bewegung im Alter“. In: *Manuelle Medizin* 49.6 (2011), S. 461–464. ISSN: 1433-0466.

[25] Muchen Wu, Parth H. Pathak und Prasant Mohapatra. „Monitoring Building Door Events Using Barometer Sensor in Smartphones“. In: *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '15. Osaka, Japan: ACM, 2015, S. 319–323. ISBN: 978-1-4503-3574-4.

[26] Jun Yang. „Toward physical activity diary: motion recognition using simple acceleration features with mobile phones“. In: *Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics*. ACM. 2009, S. 1–10.

B.2. Faltende Neuronale Netze zur Aktivitätserkennung in den Daten des Sensorgürtels

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Masterstudiengänge ESMR, Informatik und Wirtschaftsinformatik

Seminararbeit: Faltende Neuronale Netze zur Aktivitätserkennung in den Daten des Sensorgürtels

Projektgruppe:
Mobilitäts-Assessments mit körpernahen Sensoren für Zuhause
SoSe 2017 - WiSe 2018

Themensteller: Prof. Dr.-Ing. Andreas Hein
Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autor: Eugen Lange

Oldenburg, den 4. Januar 2018

Abstract

Im Rahmen der Projektrgruppe Mobilitätsassesments mit körpernahen Sensoren für zuhause (MAMKSFZ) sollen Algorithmen, die von der Projektgruppe Mobilitätsassesments mit körpernahen Sensoren (MAMKS) zur Erkennung von Aktivitäten innerhalb eines Assessments entwickelt worden, auf ihre Effektivität im häuslichen Umfeld untersucht werden.

In dieser Seminararbeit wurde zunächst eine Motivation gegeben, weshalb sich die Teilnehmer der Projektgruppe MAMKSFZ für Untersuchung eines Deep-Learning Ansatzes entschieden haben. Im weiteren Schritt wurden die notwendigen Grundlagen Faltender Neuronaler Netze (FNN) erarbeitet, um verwandte Ansätze im Bereich der Human Activity Recognition zu untersuchen und beurtelen zu können. Im Kapitel zu verwandten Ansätzen wurden vier unterschiedlichen FNN-Architekturen im Kontext der Human Activity Recognition (Human Activity Recognition (HAR)) und Activities of Daily Living (Activities of Daily Living (ADL)) untersucht und mögliche Vor- beziehungsweise Nachteile diskutiert. Basierend auf erarbeiteten Grundlagen und untersuchten verwandet Ansätzen wurde eine Architektur und ein Trainingsverlauf eines Faltenden Neuronalen Netzes, für den Einsatz in unserer Projektgruppe, entwickelt und implementiert. Der Verlauf des Trainings und dessen Ergebnis wurde anhand von Pilot-Daten bestehend aus siebzehn Aktivitäten einer Person dokumentiert und interpretiert. Schließlich wurden mögliche Weiterentwicklungen, Verbesserungen und weitere Einsatzmöglichkeiten im letzten Kapitel als Ausblick für die Projektgruppe beschrieben.

Inhalt

Abkürzungen

ADL	Activities of Daily Living
CNN	Convolutional Neural Network
DBN	Deep Belief Network
FNN	Faltendes Neuronales Netzwerk
K-NN	K-Nearest Neighbor
NN	Neuronales Netzwerk
HAR	Human Activity Recognition
MAMKS	Mobilitätsassesments mit körpernahen Sensoren
MAMKSFZ	Mobilitätsassesments mit körpernahen Sensoren für zuhause
1-NN	1-Nearest Neighbor

Abbildungen

Tabellen

1 Motivation

Im Rahmen der Projektgruppe MAMKS sind Algorithmen zur Klassifizierung von Aktivitäten, wie Gehen, Stehen, Treppesteigen, Umdrehen u.s.w. entwickelt worden. Die Projektgruppe beschäftigte sich schwerpunktmäßig mit der Klassifikation von Signalen mit Hilfe von Techniken des klassischen Maschinellen Lernens. Dadurch ist ein Werkzeug entstanden, dass den in der Domäne der Sensordatenklassifizierung klassischen Workflow implementiert. Dieser umfasst im Wesentlichen vier Schritte – Vorverarbeitung, Segmentierung, Merkmalsextraktion und die Klassifikation mittels eines ML-Algorithmus. Dieses Vorgehen ist sehr aufwändig und erfordert sehr viele Versuche und/oder umfassendes Domänenwissen. Alleine im Vorverarbeitungsschritt gibt es unzählige Möglichkeiten, die Daten sinnvoll vorzubereiten. Es kommt hinzu, dass sich verschiedene Aktivitäten zeitlich stark unterscheiden. Eine Aktivität, wie Gehen kann von wenigen Sekunden bis Stunden andauern, während die Aktivität Drehen, Aufstehen, nur einige Sekundenbruchteile lang sein kann. Diese Informationen werden in dem klassischen Ansatz nicht berücksichtigt. Auch sind einige Aktivitäten zyklisch, während andere einmalig auftreten. Aus diesen Gründen ist es notwendig, um die Breite der Aktivitäten abzudecken, verschiedene Merkmale, wie Signalenergie, -magnitude, aber auch Korrelationen, Autokorrelation, Neigungswinkel oder momentane/durchschnittliche Änderungsraten u.v.m einzubeziehen. Um diese domänenspezifische Probleme meistern zu können, sind Klassifikationsalgorithmen notwendig, die komplexe nicht-lineare Zusammenhänge behandeln können. Bewährt haben sich in diesem Bereich daher Algorithmen wie Boosted Decision Trees und Neuronale Netze (NN), was auch im Ergebnis der Projektgruppe MAMKS zu sehen war.

Neuronale Netze sind jedoch in der Lage, Merkmale automatisiert zu extrahieren. Das Problem der vollvernetzten Neuronalen Netze ist, dass mit der steigenden Anzahl an Eingabeinformationen, auch die Struktur des Netzes wächst. Dies führt zu einem starken Anstieg der anlernbaren Parameter, was in der Anwendung zu Speicherproblemen führen kann. Des Weiteren ignoriert ein Neuronales Netzwerk (NN) die geordnete Struktur eines

Inputvektors und nützt daher auch nicht, zeitliche und räumliche Zusammenhänge. Es ist möglich, den zeitlichen Zusammenhang herzustellen, indem man das Sliding-Window Verfahren einsetzt. Dies erfordert jedoch entweder die Reduktion der Daten auf einen 1D-Eingabevektor mittels Merkmalsextraktion. Dadurch könnten die Daten verwischt werden.

Faltende Neuronale Netze (Convolutional Neural Network (CNN)) wurden ursprünglich im Bereich der Klassifizierung von Bildern eingesetzt. Mittlerweile sind diese jedoch so erfolgreich, dass sie auch für viele andere Daten, wie Audo-, Video-, Text-, sowie Zeitreihen, zur Lösung von Klassifikations- oder Regressionsproblemen eingesetzt werden.

Faltende Neuronale Netze lösen alle oben beschriebene Probleme durch drei wesentliche Design-Entscheidungen. Um die Anzahl der anlernbaren Parameter zu reduzieren wird das Prinzip der geteilten Gewichte „shared-weights“ umgesetzt. Das Problem der zeitlichen und räumlichen Korrelationen wird mittels eines Prinzips namens „Rezeptive Felder“ gelöst, das durch die Untersuchung des visuellen Systems von Katzen entdeckt wurde. Schließlich werden die Probleme der stark variierenden Typen von Aktivitäten durch das räumliche und zeitliche Subsampling behandelt. Nähere Erläuterungen zu diesen Design-Entscheidungen folgen im Grundlagenkapitel dieser Seminararbeit.

In dieser Arbeit sollen daher Grundlagen von Faltenden Neuronalen Netzen behandelt werden, die zur Lösung des Klassifikationsproblems in Signalreihen notwendig sind. Zusätzlich sollen bereits bekannte Ansätze zur Aktivitätserkennung, bzw. zur Klassifikation von Zeitreihen, mittels CNNbetrachtet werden, um anschließend ein eigenes Konzept zur Signaldatenklassifikation zu konzipieren und prototypisch umzusetzen. Neben den oben beschriebenen Aktivitäten, werden auch geeignete Frameworks zur Entwicklung von CNN betrachtet. Anschließend wird eine Möglichkeit zur Integration des Ansatzes in das MAMKS-Projekt aufgezeigt.

2 Grundlagen Faltender Neuronaler Netze

In diesem Kapitel werden die notwendigen Grundlagen der CNN aufgearbeitet, die für Entwicklung des eigenen Konzepts und das Verstehen der verwandten Ansätze notwendig sind. Zunächst wird auf einige relevante Grundlagen Neuronaler Netze eingegangen, um anschließend die zugrundeliegende Designprinzipien beschreiben zu können. Die Beschreibung erfolgt, im Wesentlichen, basierend auf der Publikation von Yann LeCun [LB+95], sowie auf dem umfassenden Guide für Faltende Neuronale Netze von Hamad Aghdam [AH]. Im Gegensatz zu den Beschreibungen aus den verwendeten Quellen, bezieht sich die Beschreibung in diesem Kapitel schwerpunktmäßig auf Signaldaten. Aus diesem Grund sind, für dieses Kapitel angefertigte Grafiken sinngemäße Ableitungen und sind als solche gekennzeichnet.

2.1 Zugrunde liegende Design-Prinzipien

Strukturen Faltender Neuronalen Netze basieren im Wesentlichen auf drei Ideen. Um die Invarianz gegenüber Translationen und Stauchungen der Daten innerhalb eines Samples zu gewährleisten, wurde die Idee der Rezeptiven Felder entwickelt. Diese wurde durch die Entdeckung von lokalen rezeptiven Feldern im visuellen System von Katzen inspiriert[LB+95]. Dabei handelt es sich, oberflächlich betrachtet, um eine Verbindung mehrerer Farb-/Helligkeitsrezeptoren mit einem Neuron. Auf der Grundannahme, dass ein Merkmal in jedem Abschnitt eines Samples (bspw. Bild) auftreten kann und die Extraktion dieser in jedem Bereich dieses Samples gleich sein muss, basiert die Idee der geteilten Gewichte. Die Idee der Reduktion von Merkmalen hat mehrere Effekte. Neben der Reduktion der Dimension, werden die Merkmale auf das Wesentliche reduziert, sowie der Seiteneffekt der Invarianz gegenüber Translation, Rotation und Stauchungen in den Daten

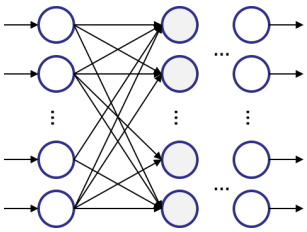


Abbildung 1: Feed Forward Net

erzielt. Die oben genannten Ideen werden im Folgenden genauer beschrieben.

Die Einheiten Künstlicher Neuronaler Netze können mathematisch, wie in der Formel 2.1, ausgedrückt werden.

$$z = \phi(w * x^T + b, \theta) \tag{2.1}$$

Hierbei ist x der transponierte Eingabevektor, w ist der anlernbare Gewichtsvektor und b ist ein für eine neuronale Schicht spezifischer, Biasvektor. Die Funktion ϕ ist eine beliebige Aktivierungsfunktion, die nach Überschreiten des Schwellwertes θ den Wert z feuert.

Ein vorwärtsgerichtetes, künstliches Neuronles Netz (engl. Feedforward Net)besteht aus mehreren Schichten. Man unterscheidet hier die Eingabeschicht (engl. Input Layer), die versteckte Schicht (engl. Hidden Layer) und die Ausgabeschicht (engl. Output Layer), wie in der Abbildung 1 dargestellt. Ein Problem dieser Struktur besteht darin, dass bei Eingabe großer Eingabevektoren auch die Größe der Gewichtsvektoren wächst. Angenommen, man wollte mit den Daten des Akzelerometers und des Gyroskops aus dem Sensorgürtel, verschiedene Aktivitäten erkennen und nimmt man als Fenstergröße 1 Sekunde (100 Werte), so benötigt man als Eingabe einen $6 * 100 = 600$-wertigen Vektor. Eine vollvernetzte Versteckte Schicht, die beispielsweise aus 32 Neuronen besteht, würde dann $600 * 32 = 19200$-Werte für die Gewichtsvektoren benötigen. Diese Anzahl an Gewichten ist sehr schwer anzulernen. Um die Anzahl dieser Parameter zu reduzieren, wird nun angenommen, dass die Werte des Eingabevektors in lokaler Korrelation zu einander stehen. Weiterhin wird angenommen, dass an jeder Stelle des Eingabevektors die gewonnenen Merkmale gleich wichtig sind. Diese Annahmen führen dazu, dass nur noch ein Gewichtsvektor für eine begrenzte lokale Umgebung eines Wertes benötigt wird. So kann die gesamte Schicht auf ein Neuron mit einem Gewichtsvektor, das beispielsweise eine lokale Umgebung der Größe 6 scant. Die Abbildung 2 veranschaulicht diese Reduktion. Da die sequentielle Berechnung der Merkmale nicht parallelisierbar ist, wird der Gewichtsvektor für die

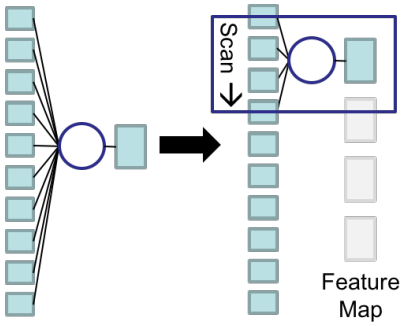


Abbildung 2: Reduktion der versteckten Schicht auf ein rezeptives Feld. Abgeleitet nach [LB+95], [AH].

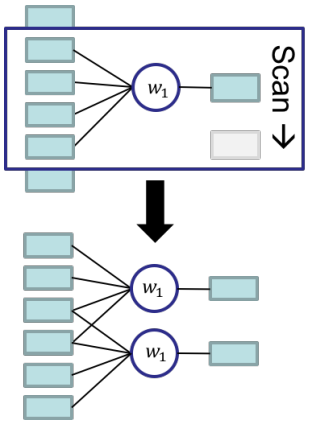


Abbildung 3: Von sequentieller Berechung zum shared-weights Prinzip. Abgeleitet nach [LB+95], [AH].

parallele Berechung eingesetzt, in dem er von mehreren neuronalen Einheiten benutzt wird. Die Abbildung 3 zeigt das Prinzip. Hierbei entsteht bei einer Schrittweite von $step = 1$ und einer Größe eines rezeptiven Feldes von $kernel = 4$, sowie bei einem Eingabevektor mit $width = 100$ Werten ein $width - kernel + step = 97$-wetiger Ausgabevektor. Im Kontext der Faltenden Neuronalen Netze - speziell in Verbindung mit Bildern - wird dieser Tensor auch Feature-Map genannt. Mathematisch gesehen, entspricht das Ergebnis dieser Reduktionsschritte einer diskreten Faltungsoperation, wie sie mit der Formel 2.2 ausgedrückt werden kann.

$$(f * w) = \sum_{k \in D} (f(x - k) * w(k)) \tag{2.2}$$

Eine Faltungsschicht besteht nun aus einer Menge faltender Einheiten. Diese Schicht extrahiert, je nach Kernelgröße, verschiedene Merkmale in Form von Feature-Maps. Diese Faltungsschichten besitzen bereits die Eigenschaft, den Merkmalsraum zu reduzieren. Den-

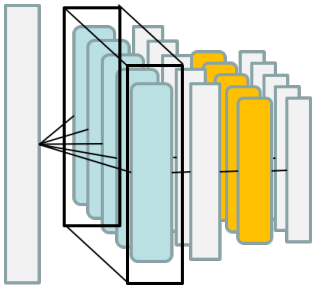


Abbildung 4: Faltungsschicht (blaue Kästen) und Reduktionsschicht (orange Kästen). Abgeleitet nach [LB+95], [AH].

noch folgt einer Faltungsschicht oft eine weitere Reduktionsschicht, die ein Feature-Map auf die wesentlichen Merkmale reduziert. Die Reduktionsschicht besitzt ebenfalls einen Kern und eine Schrittweite, die über den Grad der Reduktion entscheiden. So reduziert eine Schicht, die einen Kern $kernel = 3$ und eine Schrittweite $step = 1$ besitzt, einen 97-wertigen Vektor um $width - kernel + step = 95$. Vorstellbar sind Reduktionen nach dem Mittelwert, nach Minimal- oder Maximalwert pro Fenster (Kernel). Der maximale Wert hat sich jedoch bewährt[AH]. Daher werden die Reduktionsschichten oft einfach Max-Pooling Layer oder Subsampling Layer genannt. Die Idee des Subsamplings wird in der Abbildung 4 dargestellt. Die schwarz umrahmte Menge der blauen Kästen stellt die Faltungsschicht dar. Diese liefert eine Menge an Feature-Maps (grau), die anschließend im Max-Pooling Layer (orange) weiter reduziert werden. Das gesamte Faltende Neuronale Netz besteht aus einer Reihe von Faltungsschichten. Im einfachsten Fall der Klassifikation kann ein Netz aus einer Reihe von Faltungsschichten bestehen, die Schicht für Schicht von allgemeinen Merkmalen zu speziellen reduziert werden. Anschließend folgen oft zwei vollvernetzte vorwärtsgerichtete Schichten. Dabei werden die Merkmal in der ersten vollvernetzten Schicht zunächst zu einem eindimensionalen Vektor transformiert und an die letzte Schicht zur eigentlichen Klassifikation übergeben. Je nach Komplexität der Daten kann die vorletzte Schicht aus 256 bis 1024 Neuronen bestehen. Die letzte Schicht besteht immer aus c Neuronen, wobei c die Anzahl der Klassen ist. Als Ergebnis liefert diese Schicht einen c-wertigen Vektor mit der Angabe der Konfidenz (in Prozent) für die Voraussage einer Klasse. Die Abbildung 5 zeigt die grobe Gesamtstruktur eines Faltenden Neuronalen Netzes.

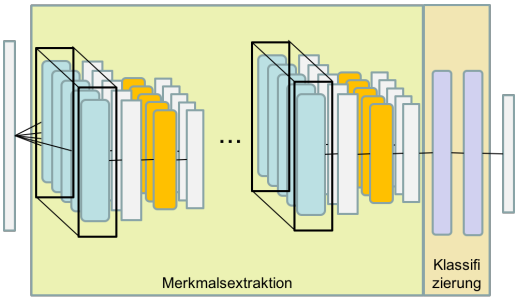


Abbildung 5: Grobe Gesamtstruktur eines Faltenden Neuronalen Netzes für ein Klassifikationproblem [LB+95], [AH].

2.1.1 Aktivierungsfunktionen

In der Formel 2.1, die eine mögliche formale Beschreibung eines Neurons darstellt, wird die Funktion ϕ genutzt. Diese Funktion entscheidet darüber, ob ein Neuron nach Eingabe von x aktiviert wird oder nicht. Die geeignete Wahl einer Aktivierungsfunktion ist entscheidend, da zur Lösung von nicht - linearen Problemen mittels Optimierungsverfahren, wie das Gradientenabstiegsverfahren, die Aktivierungsfunktion bestimmte Eigenschaften besitzen muss. Zum einem sollte die Aktivierungsfunktion selbst eine nicht lineare Funktion sein. Zum anderen muss sie eine stetige und an jeder Stelle eine differenzierbare Funktion sein, um den Gradienten $\delta\phi$, der eine partielle Ableitung ist, berechnen zu können. Eine weitere wünschenswerte Eigenschaft, ist das Verhalten der Aktivierungsfunktion nahe des Ursprungs. In der Nähe des Koordinatenursprungs sollte die Funktion möglichst gut die Identitätsfunktion abbilden können. Diese Eigenschaft beschleunigt die Optimierung mittels des Gradientenabstiegsverfahrens, da der Gradient $\delta\phi$ schneller gegen 0 konvergiert. Weitere Informationen dazu folgen im Kapitel zum Training der Faltenden Neuronalen Netze.

Im Folgenden werden, die am häufigsten verwendeten Aktivierungsfunktionen beschrieben, um eine optimale Auswahl für den Einsatz in der Projektgruppe treffen zu können. Die *Sigmoidfunktion* $\phi_{sigmoid}$ ist eine der häufug verwendeten Aktivierungsfunktionen.

$$\phi_{sigmoid}(x) = 1/(1 + e^{-x}) \tag{2.3}$$

Der Wertebereich der Sigmoidfunktion liegt in $[0, 1]$. Diese Funktion hat zwei Probleme. Zum einem verhält sie sich nicht ähnlich zur Identitätsfunktion nahe des Ursprungs. Dies führt zu suboptimalen Initialisierung zu Beginn des Trainings und damit zur längeren Lerndauer. Ein weiteres Problem ist das Verhalten der Funktion bei größeren Werten

von x. Die Struktur der Funktion bewirkt, dass die Gradienten bei großen x sehr klein werden. Man spricht davon, dass der Gradient in diesem Fall schwindet (engl. Vanishing Gradients Problem). Bei Netzwerken mit mehreren Schichten und der Anwendung des Backpropagation-Verfahrens zum Trainieren führt es dazu, dass das Training entweder nur sehr langsam voranschreitet oder ganz stockt, da sich die Werte der Gewichte kaum ändern. Diese Funktion ist daher nicht für den Einsatz mit deep-learning Netzen geeignet und wird höchstenz für die Output-Schicht verwendet.

Softsign $\phi_{softsign}$ ist eine Aktivierungsfunktion, die gegenüber der Sigmoidfunktion mehrere wünschenswerte Eigenschaften besitzt.

$$\phi_{softsign}(x) = x/(1 + |x|) \tag{2.4}$$

Der Wertebereich der Funktion liegt in $[-1, 1]$. Sie verhält sich ähnlich zur Identitätsfunktion nahe des Ursprungs, was den Trainingsverlauf positiv beeinflusst. Aufgrund ihrer Struktur leidet diese Funktion auch nicht unter dem vanishing-gradients Problem. Ein weiterer Vorteil dieser Funktion, ist einfachere Berechenbarkeit. Trotz vergleichsweise weniger komplexen Berechnung, ist diese Funktion immer noch ineffizient für die Berechung innerhalb von sehr vielen Schichten, wie es in den Faltenden Neuronalen Netzen der Fall ist.

Rectified Linear Unit ϕ_{relu} ist eine sehr einfach zu berechnende Funktion.

$$\phi_{relu}(x) = max(0, x) \tag{2.5}$$

Der Wertebereich dieser Funktion liegt in $[0, \infty)$. Strukturbedingt leidet ReLu nicht unter dem vanishing-gradients Problem, was sie für den Einsatz in Deep-Learning Netzwerken gut geeignet macht. Ein Nachteil dieser Funktion ist, dass einige Neuronen, deren Gewicht so angelernt wurden, dass der Ausgabewert negativ ist, immer mit 0 antworten. Dies kann zur Verschlechterung der Genauigkeit des gesamten Netzes führen. Dieser Effekt ist unter dem Namen Dead-Neuron Problem bekannt.

Leaky Rectified Linear Unit ϕ_{relu} löst das Dead-Neuron Problem.

$$\phi_{lerelu}(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha * x & \text{if } x < 0 \end{cases} \tag{2.6}$$

Der konstante Wert α wird zwischen $[0, 1]$ gewählt. Meist genutzter Wert ist 0.001. Die-

ser wurde empirisch ermittelt und kann sich je nach Eigenschaften des Lösungsraums unterscheiden.

2.1.2 Weitere Schichten der Faltenden Neuronalen Netze

Neben der oben beschriebenen Eingabe-Schicht, Ausgabe-Schicht, faltenden und der reduzierenden Schichten sind weitere Strukturen bekannt, die für den Einsatz in der Projektgruppe interessant sein können.

Local Response Normalization Layer ist eine Schicht, die auf dem Prinzip der lateralen Hemmung aus der Natur beruht. Laterale Hemmung hat zum Ziel, Reizunterschiede benachbarter Nervenzellen zu verstärken. Im Auge hat dies beispielsweise den Effekt der Kontrastverstärkung bei Einfall von hellen und dunklen Lichtspektern auf benachbarte Rezeptoren.[lat]

Auf Faltende Neuronale Netze bezogen, bewirkt dieses Prinzip die Verstärkung der Werteunterschiede auf benachbarten Feature-Maps einer faltenden Schicht. Wie bereits beschriieben, werden in den faltenden Schichten N Fetaure-Maps mit unterschiedlichen Kernelwerten v_i mit $i \in 0, ..., N$ berechnet. Anschließend werden die Werte mittels einer Aktivierungsfunktion zum Wert $a^i_{x,y}$ verarbeitet. Dabei ist $a^i_{x,y}$ ein Wert eines Feature-Map, an der Stelle (x, y), der mit dem i-ten Kern errechnet worden ist. Eine local-response Schicht wird direkt nach der Aktivierungsschicht angeordnet und verarbeitet die Antwort $a^i_{x,y}$ zum normalisierten Wert $b^i_{x,y}$ in der lokalen Umgebung des Wertes begrenzt auf n Schichten.[KSH12] Die Formel 2.7 veranschaulicht diese Berechnung.

$$b^i_{x,y} = a^i_{x,y}/(k + \alpha * \sum_{j=max(0,i-n/2)}^{min(N,i+n/2)} (a^j_{x,y})^2)^\beta \tag{2.7}$$

Die Werte n, k, α, β sind empirisch zu ermittelnde Faktoren. Die Autoren aus [KSH12] schlugen die Werte $n = 5, k = 2, \alpha = 10^{-4}, \beta = 0.75$ vor.

Incepption Layer ist keine einzelne Schicht im eigentlichen Sinne. Sie wird von ihren Entwicklern [SLJ+15] Modul genannt, weil sie aus mehreren unterschiedlich skalierten faltenden und reduzierenden Schichten besteht. Die Autoren in [SLJ+15] stellten ein Inception-Modul vor, das aus insgesamt sieben Schichten gebildet war. Dabei dienten drei Schichten der Dimensionsreduktion, die die vorherige faltende Schicht, die beispielsweise aus einem $6 \times 100 \times 100$-Tensor besteht auf einen $6 \times 100 \times 20$-Tensor reduzierten.

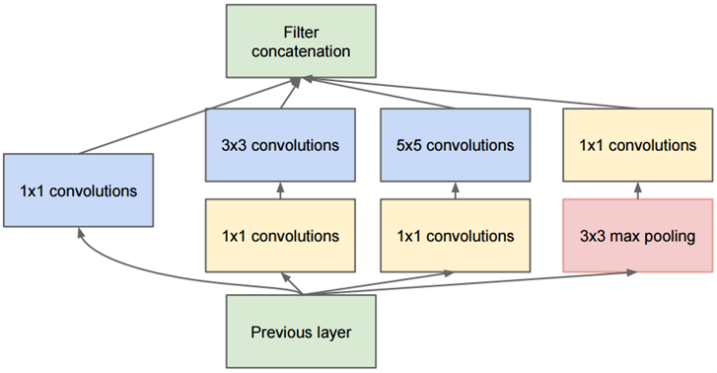


Abbildung 6: Inception-Modul, entnommen aus [SLJ+15]

Anschließend erfolgen drei Anwendungen von Faltungsoperationen mit unterschiedlichen Faltungskernen. Die Abbildung 6 zeigt die Struktur eines Inception-Moduls.

Die wesentlichen Vorteile dieser Struktur sind zu einem die Dimensionsreduktion, die durch die Anwendung der Faltungsoperationen mit dem 1×1-Kern erfolgt. Ein zweiter Vorteil ist die Extraktion von Merkmalen auf mehreren Skalierungsebenen. Während die faltenden Schichten mit der Kerngröße 5×5 grobskalierte Merkmale extrahieren, liefern 1×1-Kerne Feature-Maps mit feineren Merkmalen. Auf dem Bild ist ein weiterer Vorteil zu erkennen, nämlich die Parallelisierbarkeit der Operationen im Gegensatz zu einer sequentliellen Struktur klassischer CNN.

2.1.3 Bewährte Regeln zur Entwicklung Faltender Neuronaler Netze

In der Beschreibung zu Grundlagen Faltender Neuronaler Netze ist bereits deutlich geworden, dass es unzählige Möglichkeiten zur Entwicklung einer konkreten Architektur geben kann. Es gibt keine Anleitung, nach der man sagen kann, dass für ein bestimmtes Problem eine bestimmte Architektur optimal wäre. Hamed Aghdam [AH] hat durch Untersuchungen erfolgreicher Architekturen einige *Daumenregeln (engl. rule of thumbs)*, wie er sie selbst genannte hat, aufgestellt, an die man sich bei der Entwicklung einer konkreten Netzarchitektur halten sollte.

Aghdam hat angenommen, das ein Faltendes Neuronales Netz immer ein azyklischer Gerichteter Graph sei. Alle Schichten des Netzes sind dabei die Knoten des Graphen. Bevor die Regeln angewendet, beziehungsweise überprüft werden können, müssen zwei Grundvoraussetzugen gelten. Zum einem muss es immer genau einen Knoten zur Klassifikation, beziehungsweise zur Formulierung einer Zielfunktion (engl. Loss Function), geben.

Andererseits müssen alle Eingaben eines Knoten, Tensoren mit der selben Dimension sein. Sind diese Voraussetzungen erfüllt, so ist der Graph valide.
Die erste Regel (*R-I*) besagt, dass die vollvernetzten Schichten am Ende des Netzes, als Eingabe einen Tensor nicht größer als 8×8 haben sollten. Übliche Größen seien hier 2×2 bis 4×4. Die zweite Regel (*R-II*) beschreibt häufige Strukturierung der Schichten. Üblicherweise haben Schichten nahe der Eingabeschichte eher wenige Feature-Maps, während die Anzahl der Feature-Maps nahe des Outputs kontinuierlich wächst. Die dritte Regel (*R-III*) hält fest, dass sich bestimmte Kernel-Größen in den erfolgreichen Netzstrukturen durchgesetzt haben. Am meisten vertretene Kernelgrößen seien 3×3,5×5,7×7. Ebenfalls weitverbreitet hat sich die Strategie, eine Aktivirungsfunktion unmittelbar direkt nach der faltenden Schicht anzuordnen und nicht nach der reduzierenden Schicht. Diese Beoabachtung wird als die vierte Regel (*R-IV*) in Aghdams Guide formuliert. Dass es keinen Sinn macht mehrere Aktivierungsfunktionen direkt hintereinander zu schalten, wird in der fünften Regel (*R-V*) festgehalten. Sehr gut haben sich für faltende Schichten ReLu ähnliche Aktivierungsfunktionen, aufgrund ihrer Eigenschaften bewährt (*R-VI*).

2.2 Training eines CNN zur Klassifikation

In diesem Kapitel werden die für das Training Faltender Neuronale Netze spezifische Techniken beschrieben. Als ein allgemeines Problem, kann oft ein Mangel an representativen Daten genannt werden. Aus diesem Grund sind Techniken notwendig, die dafür sorgen, dass das CNN im Laufe des Trainings eine gute Generalisierungsfähigkeit entwickelt und sich möglichst wenig an die Beispieldaten anpasst oder im schlimmsten Fall überanpasst. Um die überwachenden Techniken des Trainings beschreiben zu können wird zunächst grob der Trainingsprozess selbst beschrieben. Die Anpassung der Gewichte eines Neuronalen Netzes kann man als ein Optimierungsproblem begreifen. In der numerischen Mathematik hat sich für diverse Optimierungsprobleme die *Methode des steilsten Abstiegs (od. Gradientenabstiegsverfahren)* bewährt. Das Prinzip dieser Methode bezogen auf das Training eines Neuronalen Netzes besteht darin, die Gewichte des gesamten Netzes Iterationsweise anzupassen. Die Anpassung erfolgt auf den Werten der Gradienten, also den partialen Ableitungen über die Gewichte, die für jede Iteration errechnet werden. Wie stark diese Anpassungen sind, hängt unter anderem auch von der aktuellen Lernrate ab. Das Verfahren wiederholt sich solange, bis ein lokales (od. globales) Minimum erreicht wird.

[Mei11] Für das Finden eines Minimus beziehungsweise Optimums, ist die Formulierung der Zielfunktion entscheidend. Mit der richtigen und problemspezifischen Formulierung der Zielfunktion, ist es möglich zu definieren, welche Fehler gemacht werden dürfen und welche nicht.

Die Beschreibung des Trainingsalgorithmus erklärt noch nicht, wie die Gewichte des gesamten Netzes angepasst werden sollen, wie hoch die Lernrate sein soll und welche Zielfunktion zur Bildung des Lösungsraumes optimal ist.

Das Verfahren, das die Gewichte des Netzes pro Iteration anpasst heißt *Backpropagation*. Dieses Verfahren hat sich auch für Faltenden Neuronale Netze in einer etwas abgeänderter Form durchgesetzt. Auf nähere Beschreibung wird an dieser Stelle verzichtet, da das Verfahren im Rahmen der Projektgruppe MAMKS bereits behandelt worden ist.

Für Klassifikationsprobleme allgemein existieren mehrere vorformulierte *Zielfunktionen*, die unterschiedliche Eigenschaften besitzen. Um die Auswahl einer Zielfunktion für den Einsatz in der Projektgruppe zu begründen, werden zwei von ihnen kurz beschrieben. Eine bekannte Zielfunktion ist die *Multiclass Hinge-Loss* . Wie der Name bereits verrät, ist diese Funktion eine Verallgemeinerung der im Zusammenhang mit Stützvektormaschinen bekannten binären Hinge-Loss. Die Multiclass berechnet den erlittenen Verlust, der durch die Fehler des Klassifikator verursacht worden ist, wie folgt. Angenommen (x_i, y_i) ist ein Beispiel aus den Mengen X, Y, wobei x_i das i-te Beispiel und y_i die dazugehörige Annotation sind. Weiterhin ist y_i ein c-wertiger Vektor und c ist die Anzahl der Klassen. Wenn beispielsweise $c = 3$ ist und x_i zur Klasse 2 gehört, dann ist $y_i = (0, 0, 1)$. Dann berechnet die Zielfunktion L_{hinge} die Differenz der, durch das Netz mit einer Menge von Gewichten W, erzielten Lösung $s = f(x_i, W)$ zum Wert s_{y_i} an der Stelle y_i.

$$L_{hinge} = \sum_{j \neq y_i} max(0, s_j - (s_{y_i} + 1) \tag{2.8}$$

Die Lösung s ist ebenfalls ein c-wertiger Vektor und enthält positive und negative Werte. Der maximale Wert dieses Vektors wird dabei als Aussage über die vorhergesagte Klasse verstanden. Auf das obere Beispiel bezogen, kann $s = (1, 1.1, -0.5)$ sein. Da der Maximalwert von s hier 1.1 ist, ist die vorhergesagte Klasse 1. Der errechnete Verlust für das Beispiel ist damit $L_{hinge} = max(0, (1-(-0.5+1))) + max(0, (1.1-(-0.5+1))) = 1.1$. Die Konstante 1 verhindert, dass nach Erreichen eines Wertes für s, beispielsweise $s = (1.0, 1.0, 1.0)$ die Funktion bereits den Verlust von 0 liefert. An dieser Stelle würden die Gewichte nicht weiter optimiert werden, obwohl die Klassifikation falsch wäre. Ein

Nachteil dieser Funktion ist, dass nach Erreichen eines Mindestwertes für eine korrekte Aussage, die Funktion nur noch 0 liefert und daher keine weitere Optmierung zulässt. [LJY]
Eine weit verbreitete Zielfunktion, die dieses Problem behebt ist die SoftPlus-Loss $L_{softplus}$.

$$L_{softplus} = ln(1 + e^x) \tag{2.9}$$

Eine weitere positive Eigenschaft ist, dass SoftPlus im Gegensatz zu ReLu an jeder Stelle differenzierbar ist. Eine ebenfalls wünschenswerte Eigenschaft für das Gradientenabstiegsverfahren ist, dass der Wertebereich der Ableitung $L'_{softplus}$ in $[0, .., 1]$ liegt. [AH]

Bis jetzt wurden Methoden und Techniken des Trainings beschrieben, die dafür sorgen dass ein Modell bis zu einem, auf einen Datensatz bezogenen, Maximum optimiert wird. Ein häufiges Problem nicht-linearer Modelle, verglichen mit linearen Modellen, ist die hohe Anpassungsfähigkeit an Trainingsdaten. Man spricht dann von hoher *Varianz*. Im Falle von linearen Modellen, die tendenziell weniger anpassungsfähig sind spricht man von hohem *Bias*. Das Ziel eines Trainings ist daher ein optimales *Bias-Varianz-Trade Off* zu finden.[AH] Da Faltende Neuronale Netze nicht-lineare Modelle sind, neigen sie zu Überanpassung an Trainingsdaten (Overfitting). Für die hohe Modell-Varianz eines Neuronalen Netzes sind zwei Faktoren verantwortlich. Zum einem ist es die Anzahl und Komplexität der versteckten Schichten. Zum anderen spielt aber auch die Magnitude der Gewichtsvektoren eine Rolle.[AH] Hier wird versucht die Magnitude möglichst niedrig zu halten, um die Komplexität des Modell gering zu halten. Wenn man sich einen $1 \times n$-Gewichtsvektor als eine Menge von Koeffizienten eines Polynoms n-ten Grades vorstellt, dann würde es bedeuten, dass wenn viele Koeffizienten den Wert 0 haben, sich der Polynom stark vereinfachen würde. Die hier zugrunde liegende Intuition ist das *Ockhams Rasiermesser-Prinzip*, welches besagt, dass die einfachste Hypothese, die einen Sachverhalt beschreibt, vermutlich die richtigste ist.[LJY]

Techniken, die nach diesem Prinzip das Training kontrollieren, sind sogenannte *Regularisierungen*. Im Allgemeinen ist eine Regularisierungsfunktion $R(W)$, wobei W ein Gewichtsvektor eines Modells ist, Teil einer Zielfunktion L.

$$L(X) = L_*(X) + \lambda * R(W) \tag{2.10}$$

Wie in der Formel 2.10 zu sehen ist, wird der Wert einer Regularisierungsfunktion zum

Wert von $L_*(X)$ hinzu addiert. Das heißt, dass das Netzwerk dafür 'bestraft' wird, wenn W zu einem hohen Wert $R(W)$ führt. Der Faktor λ gibt dabei an, wie starkt der Wert $R(W)$ ins Gewicht fällt. Ist λ groß, so werden stark komplexe Werte des Gwichtsvektors stärker bestraft, was insgesamt zu einem höheren Bias des Modells führt. Analog bewirkt ein niedriger Wert von λ eine höhere Varianz.

Eine Möglichkeit die hohe Magnitude von W zu bestrafen ist es, die sogenannte L_2-Norm von W zu berechnen. Normen sind mathematische Mittel, die in der Numerik dazu verwendet werden, um bestimmte Eigenschaften von Vektorräumen, wie Orthogonalität zu erhalten[Mei11]. Auf nähere Beschreibung wird hier verzichtet, da mathematische Grundlagen euklidischer Vektorräume nicht Inhalt dieser Seminararbeit sind.

$$R_{L2}(W) = ||W||_2 = (\sum_{i=0}^{n} W_i^2)^{1/2} \tag{2.11}$$

Es gibt weitere Regularisierungen, die unterschiedliche Eigenschaften aufweisen. Analog zu L_2 kann auch die L_1-Norm verwendet werden. Eine Kombination aus beiden ist in der Literatur als *Elastic Net* bekannt. Da L_2-Regularisierung in Verbindung mit Klassifikation am erfolgreichsten und damit am häufigsten eingesetzt wird, wird auf Beschreibung anderer Regularisierungen hier verzichtet.

Eine weitere Maßnahme gegen die Überanpassung setzt an der Struktur der Netze an. Dabei werden Teile eines Netzwerks mit Hilfe der *DropOut*-Technik 'ausgeschaltet', um zu verhindern, dass in einzelnen Netzbereichen eine Überanpassung stattfindet. Realisiert wird diese Technik, indem eine Drop-Out Layer definiert wird, welche die Werte einzelner Neuronen der vorangegangenen Schicht nach gleichverteilter Wahrscheinlichkeit $p \in [0, ..., 1]$ durchlässt. Häufiger Wert für p ist 0.8. Üblicherweise werden Drop-Out Layer nach der Klassifizierungsschicht eingesetzt. Technisch ist auch der Einsatz zwischen den faltenden Schichten vorstellbar, der sich jedoch nicht durchgesetzt hat. Ebenfalls möglich ist der Einsatz direkt nach der Input-Schicht, was einen 'verrauschten' Input der Daten simulieren und damit eine Art *Datenerweiterung* darstellen würde.[AH]

3 Verwandte Ansätze

Der Inhalt dieses Kapitels dient als Entscheidungsgrundlage für den Einsatz einer CNN-Architektur im Rahmen der Projektgruppe MAMKSFZ. Es werden vier Architekturen untersucht und miteinander verglichen. Basierend auf den Ergebnissen dieser Untersuchung werde spätere Architekturentscheidungen im Kapitel zum Einsatz in unserer Projektgruppe getroffen.
Yang et al.[YNS+15] entwickelten eine Architektur zur Erkennung von Aktivitäten des alltäglichen Lebens (ADL). Sie nutzten Dafür den *Opportunity Activity Recognition dataset*[CSC+13]. Dieser Datensatz besteht aus insgesamt 18 Klassen. Die Aufzeichung der Daten erfolgte mit 72 körpernahen und ambienten Sensoren. Alle Daten wurden auf eine gemeinsame Frequenz von 30Hz synchronisiert.

Als Eingabe definierten die Autoren daher einen Tensor der Größe $D \times 30$, wobei D die Anzahl der Sensoren und 30 einer Fenstergröße von einer Sekunde entspricht. Der Eingabeschicht folgt eine faltende Schicht mit einer Kernelgröße von 5, wobei 50 Feature-Maps generiert werden. Als Aktivierungsfunktion wird die hyperbolische Tangenzfunktion eingesetzt. Mittels einer Max-Pooling Schicht mit einer Kernelgröße von 2 reduziert sich der Merkmals-Tensor von $D \times 30 \times 50$ auf $D \times 13 \times 50$. Am Ende dieser Schicht nutzen die Autoren Local-Normalization Layer. Weitere Reduktion findet in der folgenden faltenden Schicht mittels eines 5-Kernels auf $D \times 9 \times 40$. Auch hier folgt eine Max-Pooling Schicht mit dem Kernel 2, jedoch keine Local-Normalization Schicht. Die Merkmalsextraktion wird abgeschlossen mit einer letzten faltenden Schicht mit Kernel 3, die die Merkmale noch einmal auf $D \times 1 \times 30$ faltet. Danach folgen zwei vollvernetzte Schichten, wie sie aus klassischen vollvernetzten, vorwärtsgerichteten Netzwerken bekannt sind. Die letzte Output-Schicht liefert einen 18-wertigen Vektor als Klassifikationsergebnis.

Im Trainingsprozess nutzen die Autoren fensterbasierte Segmentierung der Daten. Dabei ist die Fenstergröße 30 und die Schrittweite 3 gewählt worden. Dieses Vorgehen hat einen positiven Seiteneffekt, der Vervielfältigung von Daten. Techniken zur Verhinderung

der Überanpassung, wie Droput und Regularisierung wurden aus Performanzgründen nicht genutzt. Als Zielfunktion wurde die SoftMax-Loss benutzt. Ohne Postprocessing-Techniken erreichte das CNN eine Accuracy von 92.2%. Dieses Ergebnis ist signifikant besser als andere State of the Art Ansätze, wie Deep Belief Netzwerke und K-Nearest Neighbor (K-NN)[YNS+15].

Einen weiteren interessanten Ansatz lieferten *Zeng et al.* [ZLC+14]. Ihre Architektur wurde ebenfalls mit dem Opportunity Dataset trainiert und validiert. Im Gegensatz zu [YNS+15] nutze dieser Ansatz jedoch nur die XYZ-Signale eines körpernahen Akzelerometers am rechten Handgelenk. Als Eingabe wurde hier ein 3×64-Tensor genutzt, was ca. 1 Sekunde dieses Sensors entspricht. Interessanterweise wurde hier jeder Kanal, also X-,Y-,Z-Kanal einer eigenen faltenden Schicht zugefügt. Ähnlich der Inception-Schicht kann die Faltung hier parallelisiert erfolgen. Anschließend wurden die drei erzeugten Feature-Maps mit Max-Pooling reduziert, konkateniert und der Klassifikationsschicht übergeben.

Im Trainingsprozess wurden hier sowohl Regularisierung, als auch Droput-Techniken mit unterschiedlichen Ausfallwahrscheinlichkeiten eingesetzt. Trotz einer vergleichweise schlechteren Accuracy von 76.83% ist dieser Ansatz performanter, da eine sehr viel einfachere, parallelisierbare und mit nur einem Akzelerometer nutzbare Architektur, als in [YNS+15] entwickelt wurde.

Auch *Zheng et al.*[ZLC+14] nutzen nur drei Kanäle eines Langzeit-EKG auf ähnliche Weise wie in [YNS+15]. Hier wird die Architektur jedoch um je eine weitere faltende Schicht pro Kanal erweitert. Eine zweite Besonderheit dieser Architektur ist die abnehmende Tiefe der Feature-Maps mit jeder faltenden, beziehungsweise reduzierenden Schicht. Die erste faltende Schicht enthält dabei 256 Feature-Maps, während die letzte reduzierende Schicht nur noch 61 Feature-Maps liefert, was eher unüblich ist.

Das Training erfolgte auf den Daten der *American Heart Association*[GAG+00]. Sie besteht aus annotierten Aufzeichungen von Langzeit-EKGs gesunder Menschen und Patienten mit Erkrankungen, wie Epilepsie, Herzschwäche oder Schlafapnoe. Die Annotationen bestehen unter anderem aus 'normalen Herzschlag' , 'ventrikulare Fibriliation' und anderen. Vor dem Training wurden die Daten normalisiert, indem die zuvor errechneten Mittelwert μ und die Standardabweichung σ eliminiert wurden ($x' = (x - \mu)/\sigma$). Diese Vorgehensweise wurde nicht begründet. Es ist vorstellbar, dass dieser Vorverarbeitungsschritt das System insgesamt nur offline einsetzbar macht, da für den Online-Einsatz statistische Werte nicht

bekannt sein können. Ähnlich wie in [YNS+15] wurden die Daten auch hier fensterweise segmentiert und erweitert. Experimete haben gezeigt, dass verglichen mit einem 1-Nearest Neighbor (1-NN), das mit steigender Masse an Daten linear in seiner Berechungsdauer wächst, die Berechnungsdauer der CNN konstant bleibt. Die Erweiterung des Datensatzes, wie in [YNS+15] führt auch zur Verbesserung des Gesamtergebnisses um 3%. Die erzielte Accuracy mit einem Datensatz, der mit einer Schrittbreite von 256 erzeugt wurde, lag hierbei bei 90.34%, während der größere Datensatz, der mit einer Schrittweite von 8 erzeugt wurde eine Accuracy von 93.36% zur Folge hatte.

Viega et al. [VOW+17] haben einen ungewöhnlichen Ansatz vorgestellt. Die Hauptidee dieses Ansatzes war die Wiederverwendung bereits trainierter Modelle aus dem Bereich der Bidverarbeitung. Als Architektur haben sich die Autoren für Inception, wie sie aus [SLJ+15] bekannt ist. Die Eigenchaften dieser Architektur wurden in dem Kapitel zu Grundlagen Faltender Neuronaler Netze beschrieben. Als Grund für diese Entscheidung nannten die Autoren die Vorteile bereits trainierter Modelle. Das Modell Inception wurde bereits auf etwa $14.000.000$ Bildern angelernt. Die Idee des Transfer-Learning verspricht eine kürzere Trainingszeit und ein besseres Ergebnis auf kleinen datensätzen, verglichen mit dem Training eines komplet neuen Modells.

Das Training basierte auf einem eigenen Datensatz. Um sportliche Aktivität aufzeichnen zu können, nutzten die Autoren einen Shimmer-Sensors, der am Rücken im Lendenbereich befestigt war. Der Datensatz bestand aus 10 definierten Übungen, wie 'Dead Lift', 'Squat','Lunge' und andere. Diese wurden von 82 Person je 20 mal ausgeführt. Das Einzigartige an diesem Ansatz ist die Art auf die diese Daten segmentiert worden sind. Anstatt 1D-Faltungen für Zeitreihen zu nutzen, überführten die Autoren die Zeitreihen als Plots und speicherten diese als JPG-Bilder. Die Bildgröße betrug 434×434 Pixel. Dieser Schritt war notwendig, um die Inception-Architektur nutzen zu können. Für der Überführung fand ein Vorverarbeitungsschritt statt, wobei mittels eines Low-Pass Filter das Messrauschen reduziert wurde. Der Segmentierungsschritt erfolgte so, dass zunächst einzelne Übungseinheiten mittels Peak-Detection und Zero-Crossing Techniken markiert und anschließend auf 250 Signale pro Übungseinheit reduziert. Dieses Vorgehen erscheint, nicht zielführend, da der Sinn der Klassifizierung damit erlischt. Unter realen Bedingungen kann eine Übungseinheit auch länger oder kürzer als 250 Signale lang sein. Das Ergebnis dieses Trainings war eine Gesamtgenauigkeit von 95.89%. Die Hauptkritik dieses Ansatz ist die Nutzung einer ungeeigneten Repräsentation der Daten. Die erstellten Bildsegmente bestehen zu

etwa $3/4$ aus weißem Hintergrund. So wächst ein Segment aus 6×250 Signalen zu einem Tensor der Größe $434 \times 434 \times 3$, ohne dass es zusätzliche Inforationen liefert. Des Weiteren ist die komplexe Struktur der Inception-Architektur für diesen Einsatzbereich vermutlich nicht nötig.

4 Einsatz in der Projektgruppe

In der Projektgruppe MAMKSFZ geht es in erster Linie um die Optimierung der Klassifikationsgenauigkeit einfacher Aktivitäten wie Gehen, Stehe, Sitzen, Liegen, Umdrehen, usw. innerhalb der MAMKS-Software. Des Weiteren ist ein Einsatz im häuslichen Umfeld geplant. Damit ist auch eine Realzeitfähigkeit des Algorithmus wünschenswert.

Für die Entwicklung der Architektur wird TensorFlow als Deep-Learning-Framework, sowie TFLearn, als ein High-Level Modul verwendet. TensorFlow bietet die Möglichkeit, angelernte Modelle zu exportieren und beispielsweise in einer Java-Umgebung (Android, od. MAMKS-Software) einzusetzen. TFLearn ist eine Bibliothek, mit der es möglich ist eine eigene CNN-Architektur ohne einen erhöhten Programmieraufwand zu umzusetzen, da sie bereits vordefinierte Schichten, wie Convolution, Pooling, Merge, Fully-Connected implementiert.

Um den Aufwand der Entwicklung zu reduzieren und möglichst schnell Ergebnisse zu erzielen, werden zunächst eigene Daten mittels SenseHat und Raspberry PI 3 gesammelt und annotiert. Dies hat den Vorteil, dass im Gegensatz zum Sensorgürtel nicht zwei Betriebssysteme zur Datenaufnahme und -vorverarbeitung nötig sind, sondern nur eins. Es wird ein Ablauf zur Sammulng von Aktivitäten Gehen, Treppe aufsteigen, Treppe absteigen, Stehen, Springen, Sitzen, Liegen, Aufstehen, Hinsetzen, Umdrehen stehend, Umdrehen sitzend, Umdrehen liegend, Umdrehen gehend festgelegt. Jede Aktivität wird so ausgeführt, dass etwa 10 Minuten pro Aktivität aufgezeichnet werden können. Als Proband dient zunächst eine Person.

Die Vorverarbeitung der Daten besteht aus drei Schritten. Zuerst werden die Daten annotiert. Im Segmentierungsschritt wird eine Segmentdauer von einer Sekunde gewählt, was 20 Signalen des SenseHat entspricht. Wie in [GAG+00] und in [YNS+15] bereits gezeigt, erfolgt auch hier die Segmentierung mit einer kleineren Schrittweite als die Fenstergröße, um die Anzahl der Datensegmente und die Translationsvarianz zu erhöhen. Die

Annotation der Segmente erfolgt nach dem ArgMax-Kriterium, das heißt die maximale Anzahl der Annotationen wird für das gesamte Segment übernommen. Dies ist der Grund, warum die Fenstergröße nicht zu groß sein darf. Bei zu großer Fenstergröße kann es passieren, dass eine kurze Aktivität, von einer umgebenden Aktivität 'geschluckt' wird. Anschließend, findet nach der Segmentierung das Upsampling statt. Upsampling ist eine Data-Augmentation Technik, die dafür sorgt, dass die Anzahl der Klassen-Beispiele innerhalb eines Datensatzes ausgeglichen ist. Diese Technik ist nur dann sinnvoll, wenn es ausreichend viele unterschiedliche Samples gibt und die Inbalance nicht zu groß ist. Dabei werden die unterrepräsentierten Klassen mit bereits vorhandenen Samples ausgefüllt und gegebenenfalls mit zufälligen Variationen versehen, indem sie beispielsweise um eine kleine Zufallszahl erhöht werden.

Die Eingabeschicht der zu entwickelnden Architektur erwartet als Eingabe einen 6×20-Tensor. Um möglichst viele Merkmale auf unterschiedlichen Skalierungsebenen extrahieren zu können, besteht die Architektur aus drei parallel angeordneten faltenden Schichten mit Kernelgrößen 1×2,1×4 und 1×8 und 16 Feature-Maps. Als Aktivierungsschicht wird ReLu-Funktion, aufgrund ihrer einfachen Berechenbarkeit und Stabilität gegenüber dem Vanishing-gradients Effekt gewählt. Anschließend folgen je eine Max-Pooling Schicht mit der Kernelgröße 1×2. Eine Konkatenation verknüpft die drei zuvor erzeugten Feature-Maps, so dass ein $6 \times 46 \times 16$-Tensor entsteht. Vor der Klasssifizierung liegt eine weitere Faltung mit einem Kernle 1×4 und die letzte Reduktion mit Max-Pooling Kernel 1×2. Zum Schluß überführt eine vollvernetzte Schicht aus 512 Neuronen die Feature-Maps in einen 1D-Eingabevektro, der von einer Klassifikationsschicht mit c Neuronen interpretiert wird. Als Aktivierungsfunktion für die Klassifikationsschicht dient die SoftMax-Funktion.

5 Evaluation

Die entwickelte Architektur wurde mittels stochastischen Gradientenabstiegsverfahrens und mit der SoftPlus-Zielfunktion $L_{softplus}$ über 600 Epochen und 180.000 Steps angelernt. Das Ergebnis wird in diesem Kapitel beschrieben. Wie bereits erwähnt wurden die gesammelten Daten vor dem Training auf Balance-Eigenschaften analysiert und anschließend mittels Upsampling erweitert. Tabelle 1 zeigt die Verteilung des Pilot-Datensatzes nach der Segmentierung mit Fensterbreite 20 und Schrittweite 1.

Nach der Segmentierung erfolgte das Upsampling, wobei alle unterrepräsentierten Klassen um die Anzahl der am meisten vorkommenden Klassen-Segmente erhöht wurden. Nach dem Upsampling hatten alle Klassen also je 10660 Segmente. Dieses Vorgehen ist nicht optimal, da stark unterrepräsentierte Klassen damit ums zehnfache vervielfältigt wurden, ohne dass neue Informationen erzeugt werden konnten. Für den Einsatz in der Projektgruppe mit größerem Datensatz darf die Imbalance nicht so hoch sein. Nach dem Upsampling standen insgesamt 153504 Segmente für das Training und 38376 (20 %) Segmente für die Validierung zur Verfügung.

Die Überwachung des Trainingsprozesses wird in TensorFlow durch die Web basierte Applikation namens TensorBoard unterstützt. Die Applikation erlaubt Visualisierung des entwickelten Netzes in Form eines gerichteten Graphen. Weiterhin wird die Entwicklung der Gewichtsvektoren im Laufe des Trainings visualisiert, sodass kontinuierlicher Fortschritt oder Stagnation der Entwicklung sichtbar wird. Abbildung 7 zeigt eine Übersicht der TensorBoard-Applikation. Im Laufe des Trainings hat sich gezeigt, dass durch die starke Imbalance der Pilot-Daten das Anlernen trotz Upsampling sehr unregelmäßig verlief. Ein Grund dafür kann die Bestrafung des Netzes durch die L2-Regularisierer bei versuchten Überanpassungen der Gewichte sein. Die Grafik 8, die den Verlauf der Zielfunktion im Training zeigt, beinhaltet die beschriebenen Unregelmäßigkeiten. Verdächtig ist außerdem die Tatsache, dass der Verlauf der Validation-Loss Kurve unterhalb der Training-Loss liegt. Eine mögliche Erklärung dafür könnte eine zu erhöhte Anzahl gleicher Daten im

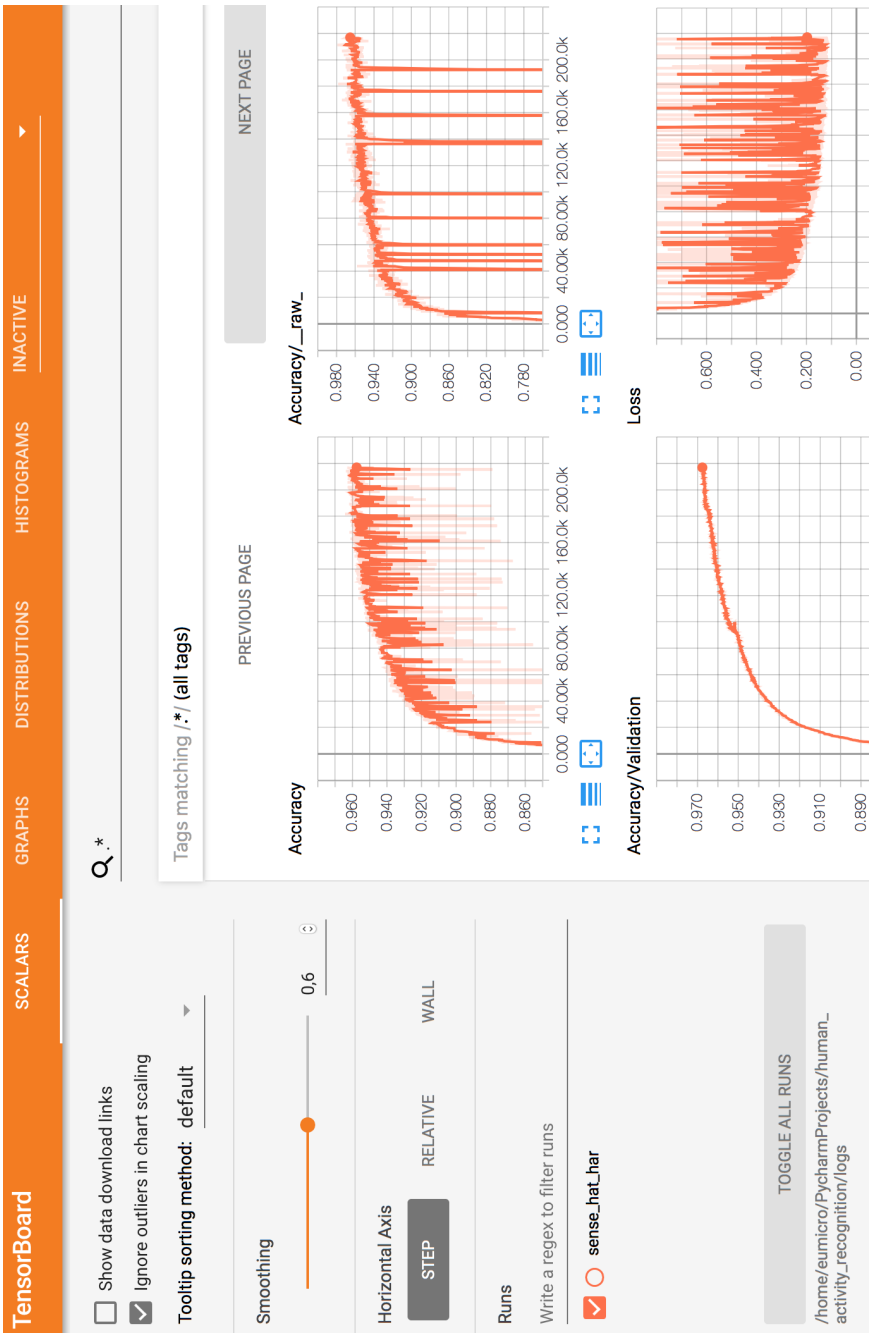


Abbildung 7: TensorBoard

Tabelle 1: Datenverteilung im Pilot-Datensatz

Category	*Anzahl Segmente*
Standing	10660
Sitting	6771
Walking	7379
Walking upstairs	1690
Walking downstairs	1604
Setting down	2833
Standing up	2696
Turn around left standing	2612
Turn around right standing	2766
Turn around right walking	1357
Turn around left walking	1552
Lying	2914
Turn around left lying	535
Turn around right lying	596
Turn around left sitting	828
Turn around right sitting	1435
Jumping	955

Pilot-Datensatz sein.

Ähnliches Verhältnis zeigt auch der Verlauf der Accuracy im Training und in der Validierung. Hier ist die Accuracy bei der Validierung sogar etwas höher als im Training. Dieses Verhältnis ist ein starkes Indiz dafür, dass eine Überanpassung stattgefunden hat. Diese ist jedoch nicht auf die Struktur des Netzes zurückzuführen, da durch die L2-Regularisierung starke Überanpassungen der Gewichte in jeder Schicht 'abgestraft' werden. Der Grund für die scheinbare Überanpassung ist starke Ähnlichkeit des Validierungsdatensatzes zum Traininigsdatensatz.

Die Abbildung 9 zeigt den Verlauf der Accuracy-Kurven im Trainingsprozess. TFLEarn bietet mehrere Möglichkeit zur Auswahl von Evaluatonsmetriken. Da die Metrik Accuracy in vielemn Publikationen verwendet wird, ist sie auch in TFLearn standardmäßig voreingestellt. Auch für diese Seminararbeit wird Accuracy verwendet. Die Schritte zur Berechnung von Accuracy für binäre Klassifikationsprobleme[Pow11] werden in der Formel 5.1 dargestellt

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \tag{5.1}$$

Hierbei wird die Anzahl der richtig erkannten Positivebeispiele und Negativbeispiele $(TP + TN)$ im Verhältnis zu allen anderen Voraussagen $(TP + TN + FP + FN)$

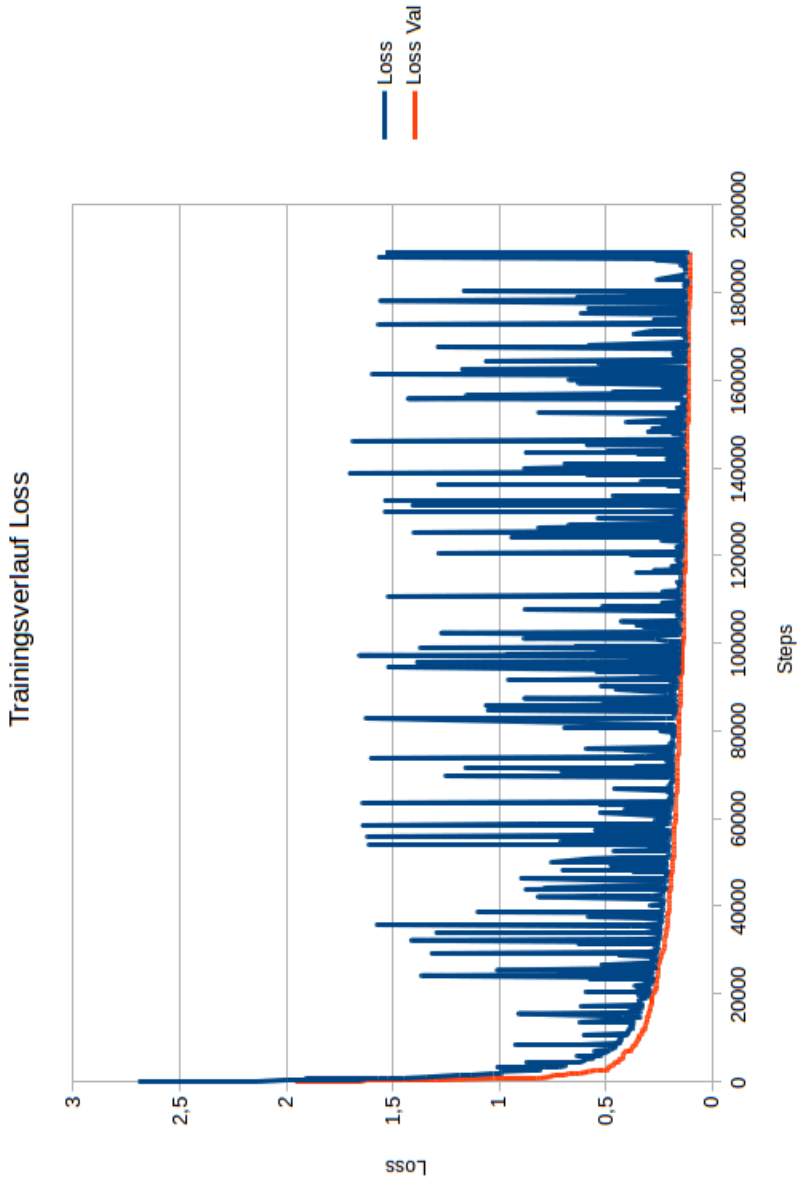


Abbildung 8: Verlauf der Zielfunktion im Training

Abbildung 9: Verlauf der Accuracy-Kurven im Training (ca. 96,6%)

eingeschlossen falsch erkannte Positiv-/Negativbeispiele gesetzt.

Nach der Trainingsphase wurde das Modell zur Online-Klassifikation auf einem Raspberry PI+SenseHat Modul verwendet. Auffällig war die Verwechslung zwischen Stehen und Sitzen. Dies ist jedoch dadurch bedingt, dass die Daten beim aufrechten Sitzen nicht vom Stehen zu unterscheiden sind. Des Weiteren wurden die Aktivitäten Treppesteigen und Gehen oft verwechselt. Qualitative Experimente haben gezeigt, dass Sprünge in der Klassifikation, wie STEHEN-SITZEN-STEHEN nicht vorhanden waren. Bemerkenswert war die Eigenschaft des Netzes, dass vor dem SITZEN immer die Aktivität HINSETZEN und zwischen SITZEN und STEHEN immer das AUFSTEHEN detektiert wurde.

6 Ausblick

In dieser Seminararbeit sollte die Machbarkeit und Nützlichkeit der Faltenden Neuronalen Netze für den Einsatz in der Projektgruppe untersucht werden. In dem Kapitel zu Grundlagen Faltender Neuronaler Netze wurde deutlich, dass die Prinzipien der Netzwerkstrukturen, den klassischen Workflow einerseits stark vereinfachen, indem die Eigenschaften von CNN die Vorverarbeitung und manuelle Merkmalsextraktion überflüßig machen. Auf der anderen Seite stellen sich weitere Fragen bezüglich der optimalen Struktur eines Neuronalen Netzwerkes. Diese Fragen können im weiteren Verlauf der Projektgruppe untersucht werden. Die im Rahmen dieser Seminararbeit implementierte Trainingsumgebung erlaubt es auf einfache Weise neue Netzwerkstrukturen umzusetzten, da eine High-Level API TFLearn für diesen Zweck verwendet wurde.

Die verwandten Ansätze zur Klassifikation von Zeitreihen zwecks Aktivitätenerkennung zeigen vielseitige Möglichkeiten auf. Im Rahmen dieser Seminararbeit konnten leider nicht alle Einsatzmöglichkeiten untersucht werden. Beispielsweise ist neben der Klassifikationsaufgabe auch Bestimmung von Geschwindigkeiten (bspw. Ganggeschwindigkeit), Höhen (bspw. beim Treppensteigen) und anderer quantitativen Größen, wie Power möglich. Diese Fragestellung kann im Rahmen dieser Projektgruppe, aufgrund fehlender Ground-Truth Daten leider nicht beantwortet werden.

Dank der integrierten Technik des Transfer-Learning in TensorFlow ist es jedoch möglich das im Rahmen dieser Seminararbeit entwickelte Modell mit neuen Daten zu verfeinern. Dies ist ein Vorteil gegenüber anderen Modellen, da sich dadurch die Trainingszeit stark reduziert. Wie der Ansatz in [VOW+17] gezeigt hat, kann sich die Trainingszeit, bei Nutzung bereits vorhandener Modelle, von mehreren Wochen auf wenige Stunden reduzieren. Dies kann wahlweise auf Datenbasis des Sensorgürtels oder mit SenseHat-Daten erfolgen. Der Einsatz mit Raspberry Pi und SenseHat bietet potentiell mehrere Vorteile, da Raspberry PI ein vollwertiges Betriebssystem mit allen daraus folgenden Vorteilen besitzt. In Kombination mit dem Sensorgürtel kann Raspberry PI beispielsweise zur Online-Annotierung von

Daten des Gürtels genutzt werden.

Vor dem Finetuning ist im Segmentierungsschritt jedoch darauf zu achten, dass die Daten besser ausbalanciert sind. Experimente mit dem Pilot-Datensatz haben gezeigt, dass die Technik des Upsampling die Überanpassung zwar bis zu einem gewissen Grad verhindern kann, jedoch dürfen die Verteilungsunterschiede nicht zu stark sein. Wahlweise kann eine Kombination aus Up- und Downsampling vorgenommen werden. Dies verhindert zu viele Duplikate in den Trainings-/Validierungsdaten.

Literatur

[AH] Hamed Habibi Aghdam and Elnaz Jahani Heravi. *Guide to Convolutional Neural Networks*. Springer.

[CSC+13] Ricardo Chavarriaga, Hesam Sagha, Alberto Calatroni, Sundara Tejaswi Digumarti, Gerhard Tröster, José del R Millán, and Daniel Roggen. The opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15):2033–2042, 2013.

[GAG+00] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet. *Circulation*, 101(23):e215–e220, 2000.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[lat] Lexikon der neurowissenschaft -laterale hemmung.

[LB+95] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[LJY] Fei Fei Li, Justin Johnson, and Serena Yeung. Convolutional neural networks for visual recognition.

[Mei11] Andreas Meister. *Numerik linearer gleichungssysteme*, volume 5. Springer, 2011.

[Pow11] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[VOW+17] Jose Juan Dominguez Veiga, Martin O'Reilly, Darragh Whelan, Brian Caulfield, and Tomas E Ward. Feature-free activity classification of inertial sensor

data with machine vision techniques: Method, development, and evaluation. *JMIR mHealth and uHealth*, 5(8), 2017.

[YNS+15] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *IJCAI*, pages 3995–4001, 2015.

[ZLC+14] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, pages 298–310. Springer, 2014.

31

Anhang

B.3. K-Nearest-Neighbor

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Abteilung Assistenzsysteme und Medizintechnik

Projektgruppe Mobilitäts-Assessments
mit körpernahen Sensoren für zuhause
SoSe17 - WiSe17/18

Seminararbeit: K-Nearest-Neighbor

Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autoren: Frederik Scharnowski

15. Februar 2018

Inhaltsverzeichnis

1 Einleitung

Um Daten erfolgreich klassifizieren zu können, werden vermehrt Machine Learning Algorithmen angewandt. Durch die gestiegene Rechenleistung in den vergangenen Jahren ist es mittlerweile gelungen, größere Datenmengen schneller zu verarbeiten. Des Weiteren wurden open Source Programme entwickelt, die es auch Privatpersonen möglich macht, sich mit Machine Learning auseinander zu setzen. Einige dieser Programme sind Beispielsweise “Tensorflow”[6], als auch die freie Standardbibliothek “scikit learn”[12], welche beide in Python umgesetzt und über eine Umfangreiche Dokumentation verfügen. Dies macht es möglich mit geringem Aufwand bereits gute Ergebnisse zu erzielen und verschiedene Modelle anzutrainieren. Das Antrainieren von neuen Modellen ist deswegen nötig, da die von der Projektgruppe erhobenen Daten in einem unkontrollierten Umfeld, der natürlichen Umgebung des Probanden, erhoben wurden. Die Bewegungsdaten der vorigen Projektgruppe hingegen wurden in einer kontrollierten Umgebung mit genauen Bewegungsabläufen aufgenommen, weshalb sich die Daten signifikant voneinander unterscheiden. Aus diesem Grund soll verglichen werden, welcher Algorithmus am besten mit den von uns erhobenen Daten umgehen und diese möglichst exakt klassifizieren kann.
In dieser Projektarbeit wird der Algorithmus K-Nearest-Neighbor behandelt. Des Weiteren werden einige Ausprägungen und Optimierungen betrachtet. Es soll im Bezug auf die Projektgruppe ermittelt werden, ob der Algorithmus eingesetzt werden kann und in wie fern er sich zum Klassifizieren eignet. In dieser Ausarbeitung wird auf die Standard Python Bibliothek für Maschine Learning “scikit learn” [12]

zurückgegriffen, um eine entsprechende Implementierung zu testen und erste Testergebnisse auf vorliegenden Daten zu erzielen. Anschließend werden mit den klassifizierten Daten “Label” erzeugt, welche in Mamks importiert werden können. Dies hat den Vorteil einen visuellen Vergleich zu haben um Ergebnisse und Vermutungen ableiten zu können.

2 Hauptteil

2.1 Grundlagen

Zur besseren Erklärung und Vermittlung werden in dieser Sektion vorerst einige Grundlagen beschrieben. Es handelt sich hierbei um eine kurze Abgrenzung von Klassifizierungsalgorithmen und deren Funktionsweise. Die nachfolgenden Kapitel sind dazu da um eine theoretische Basis zu vermitteln. Im Praxisbericht werden die Konzepte anschließend geprüft und mit Hilfe von Tests und Echtdaten bewertet.

2.1.1 Klassifizierungsgruppen

Klassifizierungsalgorithmen kann man grundsätzlich in drei verschiedene Gruppen einordnen. Zum einen gibt es Algorithmen die beim Anlernen keine zusätzlichen Informationen zu den Daten benötigen. Sie versuchen mit Hilfe der Daten Ansammlungen zu finden und deuten diese anschließend als Gruppe. Bei diesen Algorithmen handelt handelt es sich um ein “Unsupervised Learning”. Sie verwenden Clustering-Methoden um entsprechende Gruppen zu definieren.

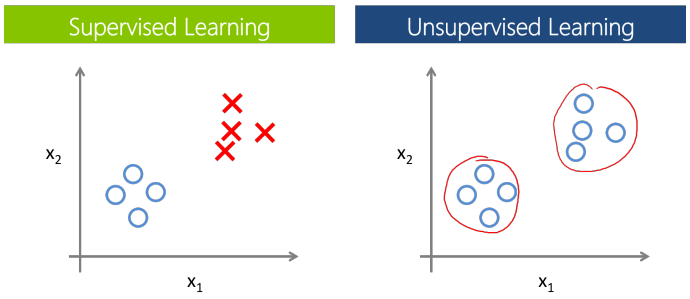


Abbildung 2.1: Super-/Unsupervised [9]

Zum anderen gibt es die “Supervised Learning” Algorithmen. Hierbei ist es nötig das der Datensatz mit dem angelernt wird, eine genaue Beschreibung der Daten vorweisen kann (im Laufe dieser Ausarbeitung “Label” genannt). Dies macht es den Algorithmen möglich ihre Grenzen zu wählen und entsprechend ein passendes Modell anzutrainieren. Bei dem K-Nearest-Neighor Algorithmus (folgend KNN), handelt es sich um einen “Supervised Learning” Algorithmus. Er kann also nur verwendet werden, wenn die Klassifizierungsdaten vorab eindeutig bestimmt wurden.

Als dritte Methode ist das “Reinforcement Learning” (Bestärkendes Lernen) zu nennen. Hierbei wird ein Umfeld erschaffen, in dem das Modell agieren kann und bekommt anschließend einen Belohnungsparameter. Jedes mal wenn das Modell eine Interaktion ausführt, bekommt das Modell einen gewissen Belohnungswert und den neuen Stand der Umwelt. Auf Grundlage dieses Wertes passt sich das Modell an und versucht für die nächste Iteration eine Verbesserung aus der Rückmeldung zu erzielen (Abbildung 2.2).

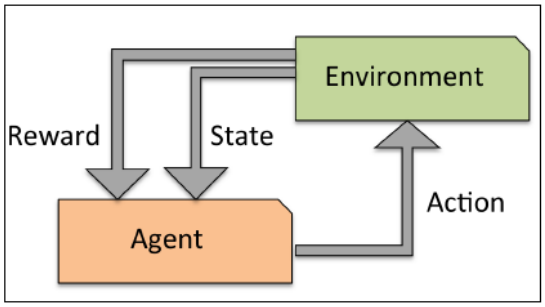


Abbildung 2.2: Reinforcement learning ([10], S. 6)

Diese Methode kann zum Beispiel zum Ermitteln eines Weges durch ein Labyrinth verwendet werden. Hierbei ist es denkbar, dass der Belohnungswert sich erhöht, um so näher die Position sich dem Ausgang nähert (vgl. [8]). Ein weiteres Beispiel ist das derzeit weit verbreitete AlphaGo Deep-Learning Modell von Google (vgl. [14]).

2.1.2 Modellparameter

Des Weiteren unterscheidet man zwischen den parametrischen und nicht-parametrischen Modellen. Bei den parametrischen Modellen handelt es sich um eine Funktion, welche eigene Parameter besitzt. Durch das Anlernen werden diese Parameter bestimmt und ergeben somit die Zielfunktion. Bei den nicht-parametrischen Modellen existieren solche Parameter nicht. Die Modelle bilden also die Ausgangsdaten ab und verwenden diese direkt zur Validierung. Hierbei ist zu beachten, dass die Modellgröße abhängig von der Anzahl der Ausgangsdaten ist. Bei den parametrischen Modellen hingegen werden nur die Parameter festgelegt und der Ursprungsdatensatz kann verworfen werden. Ein Modell mit dieser Vorgehensweise ist die “diskriminanzanalyse”/“quadratische Diskriminanzanalyse” (Seminararbeit Alexander Thomas). Ein Gegenbeispiel hierzu sind zum Beispiel “Descision trees” oder der “Random forest-Classifier” (Seminararbeit Marlon Willms). KNN zählt ebenfalls zu den nicht-parametrischen Modellen und speichert somit den Ausgangsdatensatz.

Dies hat den Vorteil, dass beim Anlernen des Modells keine Rechenleistung notwendig ist, sondern nur die Daten gespeichert werden. Der Nachteil hingegen ist dafür der hohe Rechenaufwand beim Klassifizieren von ungelabelten Daten. Die parametrischen Modelle verhalten sich entgegengesetzt. Das Anlernen nimmt einen erhöhten Rechenaufwand in Anspruch, dafür ist das Klassifizieren deutlich schneller (vgl. [13], S93). Um dies in einem Beispiel zu veranschaulichen nachfolgend die Anlern- und Prüfzeiten von einem KNN-Modell. Bei den verwendeten Daten handelt es sich um Humotion-Bewegungsdaten, um ein möglichst praxisnahes Beispiel zu produzieren. Die Ausgangsdatei wird hierbei in 50% Anlerndaten und 50% zum Validieren geteilt, damit ein direkter Vergleich gezogen werden kann.

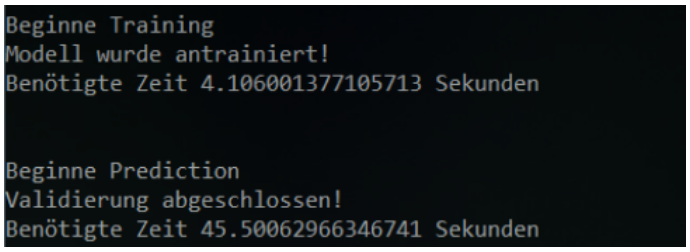


Abbildung 2.3: Script 1 - Trainingszeit

Aus dem Beispiel (2.3) wird ersichtlich, dass der Klassifizierung mehr als elf mal so lange dauert wie das eigentliche Training. Um dieses Modell im Laufe der Projektgruppe zu verwenden muss also abgewogen werden, ob eine entsprechend lange Klassifizierungszeit vertretbar ist.

2.2 Funktionsweise

In dieser Sektion wird die Funktionsweise des eigentlichen KNN-Modells beschrieben. Es wird ein Überblick verschafft nach welchen Kriterien sich das Klassifizieren richtet und wie das Modell angepasst werden kann.

2.2.1 Modelltraining

Wie der Name des Modells bereits darauf schließen lässt, werden die nähesten Nachbarn am zu suchenden Punkt ermittelt. Das Modell kann durch zwei Parameter beeinflusst werden. Zum einen lässt sich K definieren und die Distanzmetrik festlegen. Bei K handelt es sich um die Anzahl der zu betrachtenden Nachbarn. Bei der Distanzmetrik um die Herangehensweise, wie das Modell die nähesten Nachbarn ermittelt. Die Grundannahme des Modells ist, dass sich ein Punkt, welcher sich in unmittelbarer Nähe von identischen Punkten befindet, zur gleichen Gruppe gehören muss. Um die genaue Funktionsweise zu visualisieren wurde nachfolgende Abbildung erstellt.

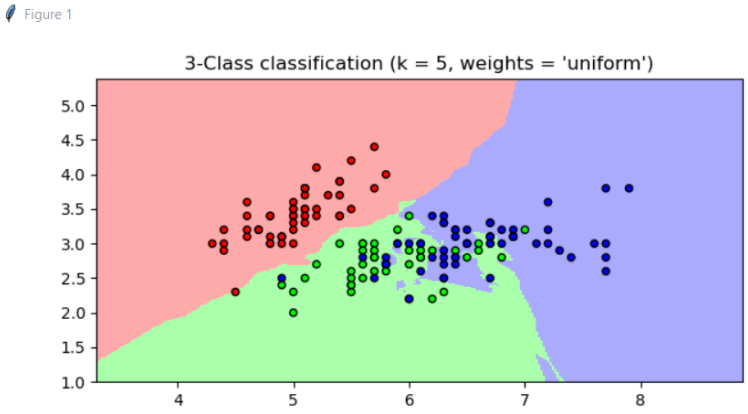


Abbildung 2.4: Script 2 - 5NN

Der Merkmalsraum besteht aus drei unterschiedlichen Punkten (rot, blau, grün). Die Anzahl der Nachbarn ist auf 5 eingestellt, was bedeutet, dass am zu überprüfenden Platz im Koordinatensystem die nähesten 5 Nachbarn ermittelt werden und per Merhheitsentscheid festgestellt wird, um was für einen Punkt es sich handeln müsste. Die Distanzmetrik ist als Standard auf euklidische Distanz gesetzt (siehe dazu **Distanzmetriken**). Der Hintergrund des Merkmalsraums ist jeweils so eingefärbt, wie das Modell auf Grundlage seines trainierten Datensatzes klassifizieren

würde.

2.2.2 Overfitting

Nachfolgend ist der gleiche Merkmalsraum noch einmal abgebildet, allerdings wird nur **ein** nähester Nachbar zum Klassifizieren verwendet.

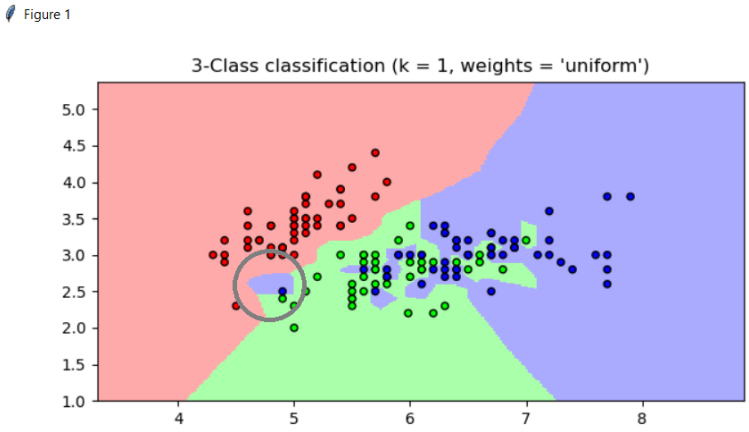


Abbildung 2.5: Script 2 - 1NN

Auf den ersten Blick fällt direkt auf, dass jedem einzelnen Punkt gleiche Aufmerksamkeit geschenkt wird. Hierdurch ist ersichtlich das offensichtliche Außenseiter (in der Abbildung grau eingekreist) beim Klassifizieren Probleme bereiten. Es herrscht also ein Overfitting des Modells. Dies bedeutet, dass Modell wurde sehr stark den Ausgangsdaten angepasst und ist womöglich nicht gut für die Klassifizierung von neuen Daten geeignet. Um einem Overfitting entgegen zu wirken, muss die Anzahl K des Modells angepasst werden.

2.2.3 Underfitting

Ein weiteres Problem ist, dass bei einer zu hohen Auswahl von K ein Underfitting des Modells herrschen kann. Dies bedeutet, dass die Daten zu stark generalisiert wurden. Die Grenzen werden dadurch zwar stark eingegrenzt und sind leicht zu identifizieren, allerdings werden kleine Gruppen von Datenpunkten eventuell fälschlicherweise nicht berücksichtigt. Nachfolgend wurde das bereits verwendete Beispiel mit K = 100 generiert. Hierbei fällt auf, dass drei eindeutige Merkmalsräume entstehen. Allerdings wird bereits bei diesem kleinen Datensatz ersichtlich, dass kleinere Gruppen (in der Abbildung grau eingekreist) nicht genug Datenpunkte besitzen um beim Klassifizieren relevant zu sein, obwohl sie es eventuell sind. Es herrscht also ein offensichtliches Underfitting. Dies bedeutet, das Modell hat die Daten so stark generalisiert das keine aussagekräftigen Grenzen mehr existieren.

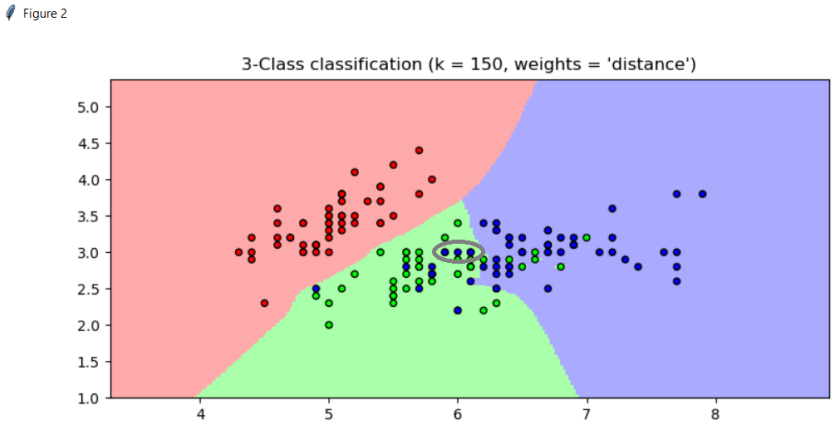


Abbildung 2.6: Script 2 - 150NN

Um kein Over- oder Underfitting zu erzeugen muss bei jedem Datensatz vereinzelt darauf geachtet werden einen guten Wert für K zu ermitteln. Dies hängt je nach Daten und Streuung des Datensatzes ab. Es sollte also vor dem Trainieren eine Bewertung der Daten vorgenommen werden um ein möglichst guten K-Wert zu ermitteln und somit ein besseres Ergebnis bei der zukünftigen Klassifizierung zu

erzielen. Hierbei ist darauf zu achten, dass sich jeder Datensatz voneinander unterscheidet und ein gutes Mittelmaß zwischen Generalisierung und Spezialisierung gewählt wird. Um einen guten Wert zu ermitteln, sollten mehrere Modelle angelernt und anschließend miteinander verglichen werden.

2.3 Distanzmetriken

Eine weitere Möglichkeit das Modell anzupassen, ist die Distanzmetrik zu wechseln. Sie legt fest, wie die Nachbarn ermittelt werden. Die Auswahl der Distanzmetrik kann je nach Datensatz verbesserte Ergebnisse erzielen. Nachfolgend werden einige der möglichen Distanzmetriken beleuchtet und mit einem Anwendungsfall verknüpft.

2.3.1 Euklidische Distanz

Die euklidische Distanz stellt eine Möglichkeit dar, die direkte Distanz zwischen zwei Punkten zu ermitteln. Hierzu benutzt sie das Prinzip von Pythagoras. Sei d die Distanz und gegeben sind 2 Punkte mit den Werten x und y, kann nachfolgende Formel verwendet werden.

$$d(x, y) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.1}$$

Die Funktion kann je nach Dimension durch mehrere Punkte erweitert werden. Somit ist zum Beispiel auch die Distanzbestimmung in einem mehrdimensionalen Raum möglich (vgl. [15]). Die euklidische Distanz eignet sich gut, wenn der Datensatz nicht aus gemischten Daten (Strings, Zahlenwerte), sondern lediglich aus Zahlenwerten besteht. Im Fall der Projektgruppe werden ausschließlich Zahlenwerte verwendet, was für einen Einsatz dieser Distanzmetrik spricht.

2.3.2 Manhattan Distanz

Bei der Manhattan Distanz wird zur Abstandsmessung nur der feste Weg über die jeweiligen Koordinaten betrachtet (vgl. [4], S. 5). Der Name ist an den Stadtaufbau von Manhattan angelehnt, welcher sehr quadratisch orientiert entwickelt wurde (vgl. [7]). In Abbildung (2.7) zeigt die rote Linie die Manhattan Distanz an. Wenn angenommen wird, das jeder Quader 1 Meter * 1 Meter misst, so erhalten wir die Manhattan Distanz von 6 Metern. Der blaue Weg zeigt eine equivalente Darstellung der Manhattan Distanz an (6 Meter). Der grüne Weg hingegen stellt die zuvor angesprochene euklidische Distanz dar. Der Start und Zielpunkt wird hierbei direkt verbunden und ergibt eine ungefähre Distanz von 4.24 Metern.

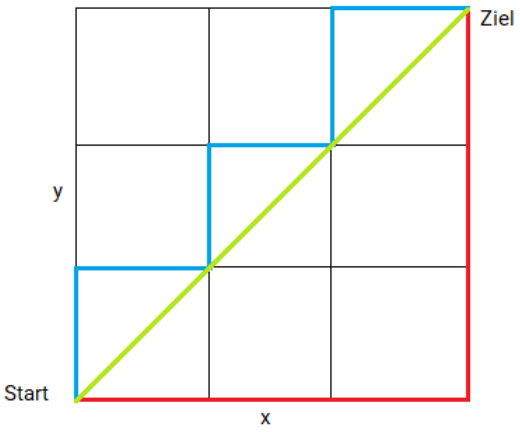


Abbildung 2.7: Manhattan Distanz

Die Verwendung der Hamming Distanz ist sehr Anwendungsdomänen abhängig. Sie wird vor allem oft in der Geoinformationstechnik verwendet (vgl. [13], S. 100 ff.). In der Praxis wird dieser Ansatz auch als Taxicab Metrik bezeichnet (vgl. [2]).

X	Y	Hamming Distanz
Tomate	Apfel	1
Tomate	Tomate	0
Apfel	Tomate	1
Apfel	Apfel	0

Tabelle 2.1: Manhattan Distanz

2.3.3 Hamming Distanz

Die Hamming Distanz (auch Hamming Abstand) wird genutzt um den Abstand von Strings, als auch Bildern zu ermitteln. Sie wird verwendet, wenn die Gruppierung von Daten ausschließlich in Kategorien vorliegt (vgl. [11]). Sie vergleicht die beiden Strings und gibt den Abstand 0 aus, falls sie identisch sind. Andernfalls ist die Distanz 1 (vgl [3]). Dies könnte zum Beispiel wie in Tabelle (2.1) aussehen.

Diese Methode wird außerdem dazu eingesetzt Handschriften zu erkennen und richtig zuzuordnen. Bei der Hamming Distanz werden die Ausgangsdaten in das Binärformat übersetzt damit sie miteinander verglichen werden können. (vgl. [5], S.59 ff.)

2.3.4 Gewichtete Distanzmetrik

Bei der gewichteten Distanzmetrik handelt es sich um ein Gewichtungsverfahren bei der Betrachtung von mehreren Nachbarn unter Berücksichtigung der Distanz vom Ausgangspunkt. Umso näher ein Nachbar sich am zu klassifizierenden Punkt befindet, desto mehr Gewicht wird ihm zugeschrieben. Hierdurch soll vor allem dem Rauschen, welches durch Ausreißer entsteht, vorgebeugt werden.

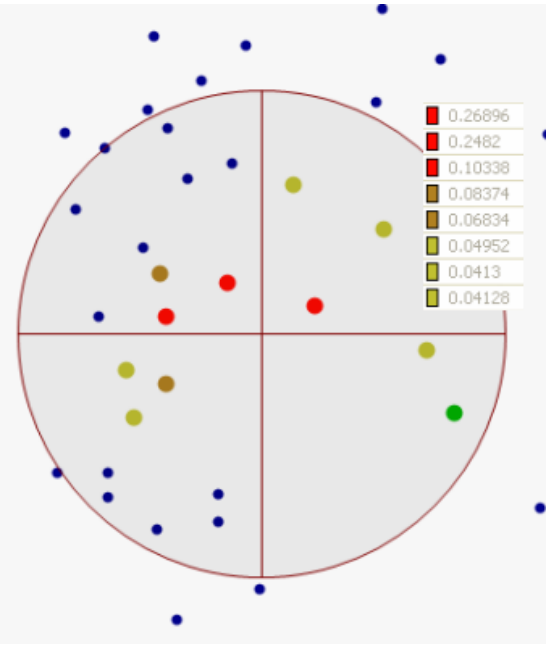


Abbildung 2.8: Weighted KNN [1]

Wie in (Abbildung 2.8) zu sehen, werden in diesem Fall 8 Nachbarn vom Mittelpunkt des Fadenkreuzes mit Hilfe der euklidischen Distanz betrachtet. Es wurden 3 gelbe, 2 braune und 3 rote Nachbarn mit der kürzesten Entfernung festgelegt. Bei einem normalen KNN wäre ein Gleichstand zwischen den roten und gelben Datenpunkten. Das Modell würde zufällig entscheiden, für welches Label es sich entscheidet. Durch die inverse Distanzwichtung werden alle Abstände der Nachbarn in Relation zum zu ermittelnden Punkt gesetzt (die Gewichtungssumme aller Nachbarn ist 1). Somit ergibt sich, dass die roten Nachbarn einen höheren Stellenwert bilden. Der zu klassifizierende Punkt im Raster erhält also ein rotes Label. Mit Hilfe dieser Erweiterung ist es möglich die ursprüngliche Distanzmetrik zu erweitern und sein Modell entsprechend seinen Bedürfnissen anzupassen.

2.4 Praxibericht mit Echtdaten

Um die theoretisch betrachteten Aspekte des K-Nearest-Neighbor Algorithmus zu überprüfen und einen direkten Praxisbezug zur Projektgruppe herzustellen, werden in diesem Kapitel erste Ansätze der Implementierung getestet. Hierbei werden bereits vorliegende Daten des Humotion-Gürtels verwendet und mit den Parametern von KNN experimentiert um erste Ergebnisse zu erzielen und zu visualisieren.

2.4.1 Distanz, KNN und WKNN

Bei erster Betrachtung der vorliegenden Daten fällt auf, dass es sich bei den vom Humotion-Gürtel aufgenommenen Daten ausschließlich um Zahlenwerte handelt (Abbildung 2.9). Hinzukommend haben wir für jeden Datenpunkt ein klar identifizierbares Label, welches durch die Projektgruppenteilnehmer bei den Hausbesuchen bestimmt wurde.

	accx	accy	accz	gyrox	gyroy	gyroz
0	-1.404	-9.577	-2.042	-0.003	0.023	-0.010
1	-1.404	-9.502	-2.041	-0.012	0.028	-0.001
2	-1.404	-9.502	-2.041	-0.012	0.028	-0.010
3	-1.405	-9.426	-2.041	-0.008	0.028	-0.024
4	-1.329	-9.577	-2.118	-0.008	0.023	-0.029
5	-1.329	-9.577	-2.118	-0.008	0.019	-0.020
6	-1.329	-9.502	-2.117	-0.021	0.023	0.009
7	-1.255	-9.426	-2.118	-0.021	0.019	0.004
8	-1.254	-9.502	-2.118	-0.017	0.009	-0.020
9	-1.180	-9.426	-2.043	-0.017	0.004	-0.025
10	-1.179	-9.502	-2.044	-0.017	-0.001	-0.025
11	-1.104	-9.577	-1.970	-0.012	-0.010	-0.039

Abbildung 2.9: Accelerometer- und Gyroskopdaten

Die Distanz zwischen den einzelnen Punkten kann also leicht mit der euklidischen Distanz bestimmt werden. Da wir nicht die Anforderung haben, uns streng nach den Achsen zu richten, setzen wir nicht die Manhattan Distanz ein. Des Weiteren handelt es sich vom Datentyp nicht um Mischdaten, weshalb die Hamming Distanz ebenfalls vernachlässigt werden kann.

Im nächsten Schritt muss die Anzahl der Nachbarn festgelegt werden. Um einen ersten Vergleich zu bekommen, wurde eine Teilmenge der vorliegenden Daten analysiert und mit unterschiedlichen Nachbarn antrainiert. Wie in Abbildung (2.10) zu sehen, handelt es sich bei der Prüfung von nur einem Nachbarn um das beste Ergebnis.

```
KNN - Genauigkeit                    WKNN - Genauigkeit

Genauigkeit: 99.645 % 1 Nachbar      Genauigkeit: 99.645 % 1 Nachbar
Genauigkeit: 99.505 % 3 Nachbarn     Genauigkeit: 99.555 % 3 Nachbarn
Genauigkeit: 99.32 % 5 Nachbarn      Genauigkeit: 99.415 % 5 Nachbarn
Genauigkeit: 99.265 % 7 Nachbarn     Genauigkeit: 99.36 % 7 Nachbarn
Genauigkeit: 99.16 % 9 Nachbarn      Genauigkeit: 99.29 % 9 Nachbarn
Genauigkeit: 99.055 % 11 Nachbarn    Genauigkeit: 99.24 % 11 Nachbarn
Genauigkeit: 98.945 % 13 Nachbarn    Genauigkeit: 99.16 % 13 Nachbarn
Genauigkeit: 98.91 % 15 Nachbarn     Genauigkeit: 99.15 % 15 Nachbarn
Genauigkeit: 98.835 % 17 Nachbarn    Genauigkeit: 99.065 % 17 Nachbarr
Genauigkeit: 98.755 % 19 Nachbarn    Genauigkeit: 99.02 % 19 Nachbarn
Genauigkeit: 98.71 % 21 Nachbarn     Genauigkeit: 98.98 % 21 Nachbarn
Genauigkeit: 98.32 % 39 Nachbarn     Genauigkeit: 98.64 % 39 Nachbarn
Genauigkeit: 98.28 % 41 Nachbarn     Genauigkeit: 98.605 % 41 Nachbarr
Genauigkeit: 98.26 % 43 Nachbarn     Genauigkeit: 98.565 % 43 Nachbarr
Genauigkeit: 98.265 % 45 Nachbarn    Genauigkeit: 98.535 % 45 Nachbarr
Genauigkeit: 98.255 % 47 Nachbarn    Genauigkeit: 98.5 % 47 Nachbarn
Genauigkeit: 97.735 % 99 Nachbarn    Genauigkeit: 97.94 % 99 Nachbarn
```

Abbildung 2.10: Script 3 - KNN vs. WKNN

Hierbei ist davon auszugehen, dass es sich um ein starkes Overfitting des Modells handelt. Außerdem kann man beobachten, desto mehr Nachbarn verwendet werden, desto schlechter wird das Gesamtergebnis der Genauigkeit.
Im Vergleich dazu wurde der gleiche Datensatz mit dem gleichen Modell validiert, allerdings mit der Erweiterung der Distanzwichtung. Bei K = 1 ist kein Unterschied ersichtlich, da die Gewichtung erst ab zwei Vergleichswerten zum Tragen kommt (1 Nachbar = volle Gewichtung). Allerdings sieht man beim direkten Vergleich das die Gewichtung, egal wie viele Nachbarn ausgewählt, eine höhere Erkennungsgenauigkeit liefert. Hieraus kann geschlussfolgert werden, dass für die uns vorliegenden Daten eine Gewichtung mit Weighted KNN sinnvoll ist.

2.4.2 Defizite Bewegungserkennung

Um die Erkennungsgenauigkeit genauer zu analysieren wurde nachfolgend eine Konfusionsmatrix mit den einzelnen Bewegungsaktivitäten und der dazugehörigen Trefferquote generiert. Bei dem angelernten Modell handelt es sich um einen Weighted KNN mit euklidischem Abstand, einer Nachbarbetrachtung von 21 und inverser Distanzwichtung. Das Modell wurde mit Hilfe von einigen vorliegenden Probandendaten angelernt und erzielte eine Genauigkeit von 96,2%.

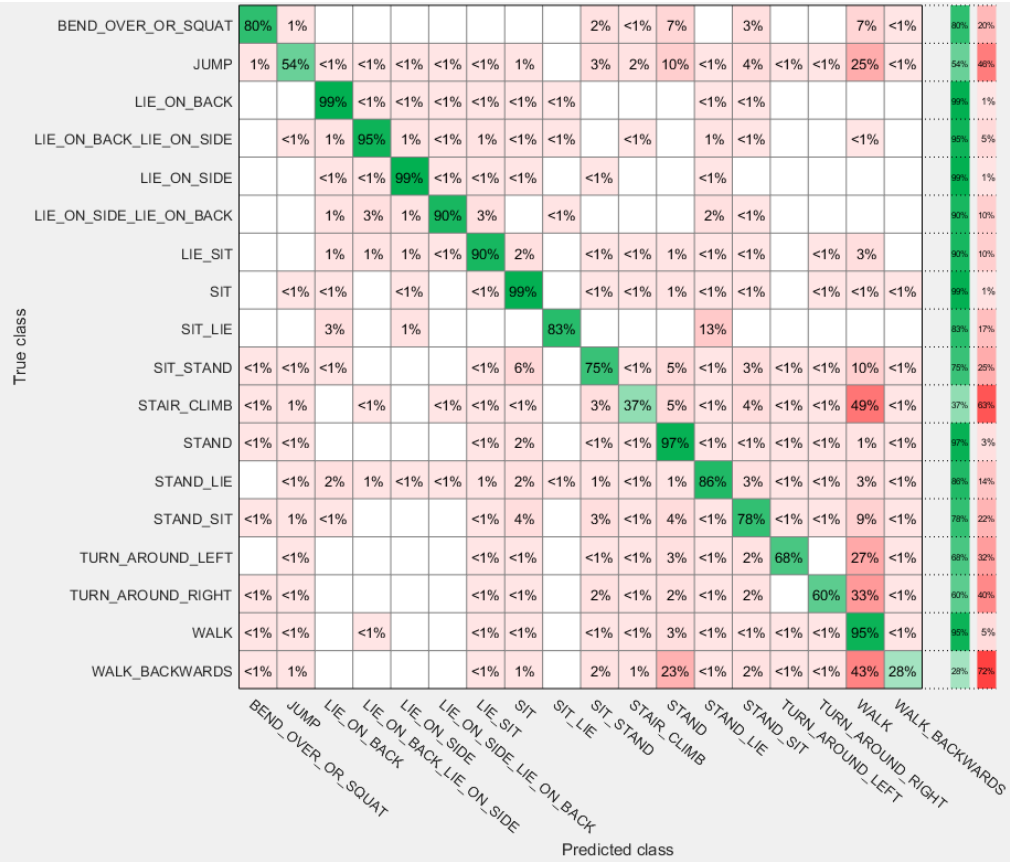


Abbildung 2.11: WKNN - Konfusionsmatrix

Besonders auffällig in Abbildung (2.11) sind die hohen Erkennungsraten beim Lie-

gen, Sitzen, Stehen und Gehen. Schlecht erkannt werden hingegen das Springen, Treppensteigen und Rückwärtslaufen. Alle drei Aktivitäten werden hierbei sehr oft für ein normales Gehen interpretiert. Hieraus lässt sich deuten, dass die falsch klassifizierten Aktivitäten einen sehr ähnlichen Charakter besitzen. Der Algorithmus erkennt keinen eindeutigen Wert mit dem er die Aktivitäten zuverlässig voneinander abgrenzen kann. Ebenfalls werden Drehungen mit einer hohen Wahrscheinlichkeit dem Laufen zugeordnet. Dies könnte den Grund haben das viele Drehungen beim Laufen durchgeführt werden. Es entsteht also eine Mischform von einem Drehen und Laufen, welche nicht eindeutig identifiziert werden kann. Die restlichen Aktivitäten haben einen guten Erkennungsgrad von rund 80% und die Fehlerverteilung ist eher unauffällig.

2.4.3 Labelgenerierung

Um einen weiteren visuellen Vergleich zu erlangen, wurde ein Script erstellt, mit dem unbekannte Daten klassifiziert werden und anschließend eine für die MAMKS-Software kompatible XML-Datei erstellt. Hierzu werden die Binärdateien aus MAMKS und das antrainierte Modell (es funktioniert jedes in “scikit learn” erstellte Modell, wie zum Beispiel SVM, QDA etc.), in den Ordner des Scripts kopiert. Anschließend wird das Script ausgeführt. Das entstandene Label wird im gleichen Ordner abgelegt und kann direkt in MAMKS importiert werden.

Allerdings sollte beachtet werden das diese Vorgehensweise nur dazu dient, um sich einen ersten Eindruck über das antrainierte Modell zu verschaffen.

Nachfolgende Label wurden mit dem oben genannten Script erstellt. Im Script selber kann definiert werden, welchen Zeitintervall das Label mindesten haben soll, damit es in der Darstellung berücksichtigt wird. Die Label in den folgenden Abbildungen wurden so erstellt, dass sie mindestens einen Zeitraum von einer halben Sekunde abdecken und ansonsten verworfen werden. Dies hat den Vorteil das sich

ein Rauschen in den Daten schnell feststellen lässt. In den Bildern werden zwei Labelgruppen dargestellt. Zum einen die von der Projektgruppe manuell festgelegten Label per Android-App und darunter die vom KNN-Modell erstellten Label.

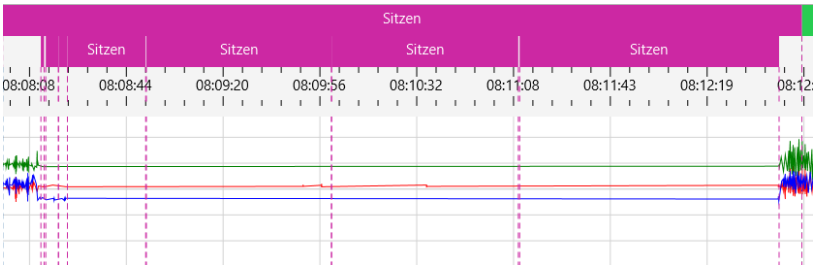


Abbildung 2.12: Label - Sitzen

In Abbildung (2.12) sieht man, dass ein durchgehendes Sitzen gut vom Modell klassifiziert wird. Falls es allerdings unruhige Sitzpassagen gibt (etwa eine kurze Neupositionierung), wird dies nicht mehr als durchgehendes Sitzen erkannt. Die Abgrenzungen zur nächsten Aktivität sind augenscheinlich dennoch gut getroffen.

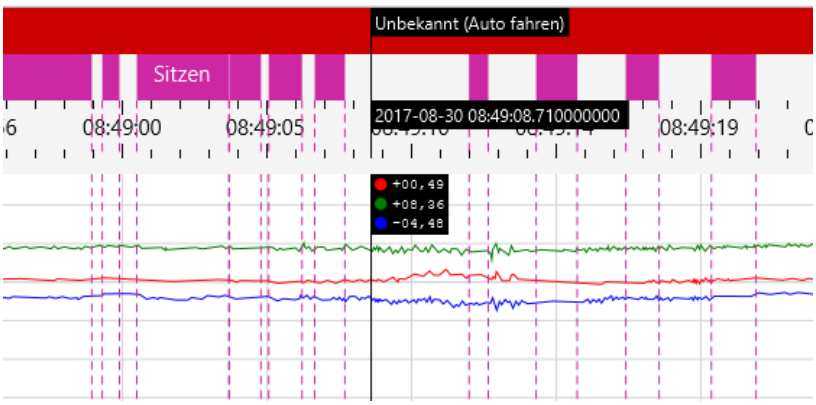


Abbildung 2.13: Label - Autofahren

Beim Autofahren (2.13) sieht man etwas stärkere Störungen in den Daten. Das Modell kann die Grundaktivität zwischenzeitlich zwar richtig bestimmen, erkennt

aber bei erhöhten Störungen nicht mehr die richtige Aktivität. Es sei jedoch zu berücksichtigen, dass die Aktivität Autofahren als solches nicht angelernt wurde.

Abbildung 2.14: Label - Gehen

Beim Gehen (2.14) sieht man sehr wenige konstante Label die länger sind als eine halbe Sekunde. Auch wenn die Aktivität Gehen eine Erkennungsrate von 95% besitzt (2.11), so sind zwischendurch immer vereinzelte Punkte, die nicht als Gehen gekennzeichnet werden. Hierdurch wirkt die Darstellung der Label sehr inkonstant.

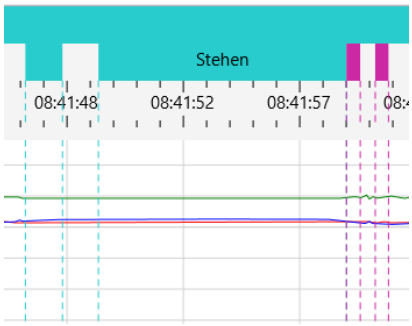


Abbildung 2.15: Label - Stehen

Auch beim Stehen (2.15) sieht man, dass die Label stark durch die Einzelpunktbetrachtung beeinflusst werden. Zwar werden Großteile der Aktivität richtig erkannt, allerdings werden kleine Änderungen direkt anders klassifiziert. In diesem Fall wird zwischenzeitlich Sitzen(lila) statt Stehen (türkis) erkannt.

Eine Lösung die man in diesem Zusammenhang in Betracht ziehen sollte ist das Unterteilen der Ausgangsdaten in statische Fenstergrößen und eine Zuordnung der Label per Mehrheitsentscheid. Somit werden vereinzelte Label (die eventuell nur Sekunde/100 lang sind) vermieden. Sollte sich für die Implementierung des KNN entschieden werden, müsste man sich anschließend auf eine geeignete Fenstergröße einigen. Auch wenn die Erkennungsrate hierdurch negativ beeinflusst werden sollte, so ist die Darstellung für den Endanwender deutlich sinnvoller und kann als Referenzwert zur eigenen Interpretation dienen.

3 Fazit

K-Nearest-Neighbor kann fast in jedem Einsatzgebiet verwenden, da er durch die verschiedenen Distanzmetriken, die Anzahl der Nachbarn und die Gewichtung an das jeweilige Gebiet angepasst werden kann. Der Algorithmus hat eine schnelle Anlernzeit, benötigt allerdings beim Klassifizieren von Daten viel Zeit. Entsprechend dem Anwendungsfall sollte darauf geachtet werden, ob eine derartige Zeitspanne tolerierbar ist. Aus eigenen Erfahrungen beträgt bei einem durchschnittlichen Datensatz in Mamks (etwa 2 Stunden) die Labelzeit in etwa 45 Sekunden. Das trainierte Modell hat ca. eine Größe von 100Mbyte und bewegt sich noch in einer humanen Größenordnung, um eventuell auch in Zukunft auch auf einem mobilen Gerät zum Einsatz zu kommen. Die Ergebnisse der angelernten Modelle haben eine sehr hohe Erkennungsrate. Vor allem mit einer gewichteten Distanzmetrik in Kombination mit der euklidischen Distanz konnten sehr gute Ergebnisse auf den vorliegenden Echtdaten erzielt werden. Grundsätzlich halte ich eine weitere Verwendung innerhalb der Projektgruppe als sinnvoll. Dennoch sollte beachtet werden, dass die Label entsprechend nachbearbeitet werden, um eine für den Anwender sinnvolle Visualisierung bereitzustellen.
Die in dieser Arbeit verwendeten Scripte zur Erstellung von Auswertungen und Generierung der Label in Python können im Bitbucket eingesehen und verwendet werden. [1]

[1] `http://amt.w2kroot.uni-oldenburg.de:7990/projects/PG_2017_MAMKSFZ/repos/repository/browse/Seminararbeit/Scharnowski`

Literatur

[1] Arcgis. *How inverse distance weighted interpolation works*. URL: https://pro.arcgis.com/de/pro-app/help/analysis/geostatistical-analyst/how-inverse-distance-weighted-interpolation-works.htm.

[2] Margherita Barile. *Taxicab Metric*. 2015. URL: http://mathworld.wolfram.com/TaxicabMetric.html.

[3] Alexander Bogomolny. *Distance Between Strings*. 2017. URL: https://www.cut-the-knot.org/do_you_know/Strings.shtml.

[4] College of Natural Sciences. *Nearest neighbor classifiers: Chapter e-6*. 2015. URL: http://www.cs.colostate.edu/~cs545/fall15/lib/exe/fetch.php?media=wiki:15_distance_based.pdf.

[5] Vasantha Kalyani David, Janusz Kacprzyk und Sundaramoorthy Rajasekaran, Hrsg. *Pattern Recognition using Neural and Functional Networks*. Bd. 160. Studies in Computational Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. ISBN: 978-3-540-85129-5. DOI: 10.1007/978-3-540-85130-1. URL: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10253349.

[6] Google. *TensorFlow - An open-source machine learning framework for everyone: Version 1.5*. URL: https://www.tensorflow.org/.

[7] improved outcomes software. *Manhattan*. 2016. URL: http://www.improvedoutcomes.com/docs/WebSiteDocs/Clustering/Clustering_Parameters/Manhattan_Distance_Metric.htm.

[8] Jaderberg Max, Mnih Vlad und Wojte Czarnecki. *REINFORCEMENT LEARNING WITH UNSUPERVISED AUXILIARY TASKS*. 2016. URL: https://arxiv.org/pdf/1611.05397.pdf.

[9] Olivia Klose. *Machine Learning (2) - Supervised versus Unsupervised Learning*. 2015. URL: http://oliviaklose.azurewebsites.net/machine-learning-2-supervised-versus-unsupervised-learning/.

[10] Sebastian Raschka. *Python machine learning: Unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics*. Community experience distilled. Birmingham und Mumbai: Packt Publishing open source, 2016. ISBN: 978-1-78355-513-0.

[11] Sayad Saed. *K Nearest Neighbors - Classification*. 2014. URL: http://www.saedsayad.com/k_nearest_neighbors.htm.

[12] scikit learn. *Scikit Learn - Machine Learning in Python: Version 0.19.1*. URL: http://scikit-learn.org/stable/.

[13] Shashi Shekhar, Hui Xiong und Xun Zhou, Hrsg. *Encyclopedia of GIS*. Second edition. Cham: Springer, 2016. ISBN: 978-3-319-17885-1. URL: http://dx.doi.org/10.1007/978-3-319-17885-1.

[14] Sigmoidal. *How does AlphaGo work? Power of Reinforcement Learning*. 2017. URL: https://sigmoidal.io/alphago-how-it-uses-reinforcement-learning-to-beat-go-masters/.

[15] Eric W. Weisstein. *"Distance": MathWorld*. 2018. URL: http://mathworld.wolfram.com/Distance.html.

B.4. Boosting Neuronal Networks

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Abteilung Assistenzsysteme und Medizintechnik

Projektgruppe Mobilitäts-Assessments
mit körpernahen Sensoren für zuhause
SoSe17 - WiSe17/18

Seminararbeit: Boosting Neuronal Networks

Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autoren: Marius Leyh

12. März 2018

Zusammenfassung

In dieser Seminararbeit wird ein Verfahren zum Boosting von Neuronalen Netzen gezeigt. Um die gesamte Thematik besser verstehen zu können, wird im Vorfeld das Thema Boosting genauer behandelt. Speziell wird in dieser Seminararbeit der Algorithmus Adaboost erklärt. Dieser Algorithmus ist einer der weitest verbreiteten Algorithmen und wird weltweit in verschiedenen Bereichen eingesetzt. Da die Ursprungsform des AdaBoost sich nur mit einer Klassifizierung über zwei Labels befasst, stellen wir im Folgenden noch die Erweiterungen AdaBoost.M1 und AdaBoost.M2 vor. Diese Algorithmen wurden entwickelt um auch Datensätze mit zwei und mehr Labels zu klassifizieren. Im Anschluss wird die Brücke zu neuronalen Netzen geschlagen und ein Verfahren vorgestellt, mit dem beide Themen miteinander verbunden werden können. In diesem Verfahren werden mehrere neuronale Netze antrainiert und nacheinander angereiht. Die anschlißenden Ergebnisse zeigen eine Verbesserung gegenüber dem urspünglichen einzelnen neuronalen Netz.

Inhaltsverzeichnis

1 Boosting

Um das Boosting von Neuronalen Netzen zu verstehen, hilft es sich vorab mit der Idee des Boostings allgemein zu befassen. Boosting kann helfen die Genauigkeit eines Klassifikators erhöhen. Um dies zu erreichen werden mehrere Klassifikatoren entworfen, welche mit einer Gewichtung entsprechend des Trefferquote versehen werden. Um einen genaueren Einblick zu erhalten, wird hier die Methode Adaboost vorgestellt. Diese Methode wurde erstmals im Jahr 1995 von Yoav Freund und Robert Schapire vorgestellt und wird in vielen Bereichen verwendet.

1.1 Beispiel

Wir stellen uns ein Programm zur Vorhersage eines Pferderennens vor [Szc05]. Dieses soll eine Aussage darüber treffen, welche Wetten statistisch vielversprechend sind. Hierfür soll eine Analyse mittels Boosting angefertigt werden. Wie bereits beschrieben werden dafür verschiedene Regeln erstellt, die verschieden gewichtet werden. Eine einzelne Regel ist dafür oft zu unflexibel oder zu kompliziert. Einfachere Regeln wären beispielsweise: Tippe auf das Pferd, das in der letzten Zeit oft gewonnen hat; Tippe auf das Pferd, das die besten Quoten hat; Tippe auf das Pferd, das die meisten Siege hat. Diese Regeln werden gewichtet und zusammengefasst. Man spricht daher auf von schwachen und starken Klassifikatoren. Eine einzelne Regel ist in ihrer Aussage beschränkt und daher schwach. Fasst man jedoch viele Schwache zusammen, erhält man einen starken Klassifikator.

2 AdaBoost

Die ursprünglich veröffentlichte Methode des AdaBoost (Adaptive Boosting) [Ada] eignet sich nur für Datensätze im zweidimensionalen Raum. Eine Erweiterung der Methode wird weiter unten beschrieben. Um einen Starken Klassifikator mittels AdaBoost zu erhalten, werden die im folgenden detaillierter beschriebenen Schritte der Reihe nach ausgeführt.

Algorithm 1 Pseudocode AdaBoost [Sch]

Input: sequence of m examples $\langle (x_1, y_1), ..., (x_m, y_m) \rangle$ where $x_i \in -1, +2$

Initialize $D_1(i) = 1/m$ for $i = 1, ..., m$.

Do for $t = 1, 2, ..., T$

1. Train **WeakLearner** using distribution D_t.
2. Get weak hypothesis $h_t : \chi \to -1, +1$.
3. Aim: select h_t with low weighted error:
$$\epsilon_t = Pr_{i\ D_t}[h_t(x_i) \neq y_i].$$
4. Choose $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$.
5. Update, for $i = 1, ..., m$:
$$D_{t+1}(i) = \frac{D_t(i) \exp(+\alpha_t y_i h_t(x_i)}{Z_t}$$
where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the hypothesis: $H(x) = \text{sign}\Big(\sum_{t=1}^{T} \alpha_t h_t(x) \Big)$

2.1 Gewichtung bestimmen

Im ersten Durchgang der AdaBoost Methode wird die Gewichtung initial bestimmt. Dabei bekommt jedes einzelne Element eines Datensatzes eine gleiche Gewichtung (1/n ; n = Anzahl der Elemente). In den folgenden Durchgängen wird die Gewichtung wie folgt bestimmt.

Wenn das Element im vorherigen Durchgang richtig klassifiziert wurde: [Nos]

$$w_{neu} = \frac{1}{2} * \frac{1}{1-e} * w_{alt}$$

Wenn das Element im vorherigen Durchgang falsch klassifiziert wurde: [Nos]

$$w_{neu} = \frac{1}{2} * \frac{1}{e} * w_{alt}$$

Durch den Aufbau der Formeln wird die Gewichtung gleich verteilt zwischen richtig klassifizierten und falsch klassifizierten Elementen. Da beim AdaBoost immer der Klassifizierer mit der geringsten Fehlerquote gewählt wird, bekommen die falsch klassifizierten Elemente einen höheren Wert zugewiesen.

2.2 Fehlerqoute bestimmen

Die Fehlerquote entscheidet, welcher schwacher Klassifikator (Weak-Learner) für den starken Klassifikator (Strong-Learner) ausgewählt wird. Dafür wird für jeden Weak-Learner eine eigene Fehlerquote errechnet. Diese ergibt sich aus der prozentualen Anzahl aller falsch klassifizierten Daten. Danach wird der Weak-Learner mit der geringsten prozentualen Fehlerquote gewählt.

2.3 VotingPower bestimmen

Damit ein Strong-Learner genutzt werden kann, wird für jeden gewählten Weak-Learner eine VotingPower bestimmt. Diese gibt an welchen Einfluss der Weak-Learner auf das Ergebnis bezüglich eines Elements hat. Die VotingPower wird wie folgt berechnet (die Variable e entspricht hierbei der Fehlerqoute / 100): [Nos]

$$\alpha = \frac{1}{2} ln((1 - e)/e)$$

2.4 Abbruchmöglichkeiten

Eine AdaBoost Schleife kann nach jedem Durchgang abgebrochen werden. Ein Abbruch dieser macht jedoch erst nach dem dritten Durchlauf Sinn. Da die VotingPower sonst Ihre Bedeutung verliert. Demnach wird vor dem Durchlaufen des Algorithmus eine maximale Anzahl an Durchläufen festgelegt. Diese sollte in jedem Fall kleiner sein als die Anzahl der möglichen Weak-Learner, da sonst alle Weak-Learner benutzt werden.

2.5 Erweiterungen

Da die Grundstruktur von AdaBoost nur mit zweidimensionalen Daten umgehen kann, gibt es mehrere Ansätze diese Methode für mehrdimensionale Datenstrukturen nutzbar zu machen. In den ersten Anfängen AdaBoost zu erweitern, wurden die Datensätze in mehrere zweidimensionale Strukturen aufgeteilt und später wieder zusammengeführt. Da diese Methode umständlich und fehleranfällig ist, wurden mit der Zeit Alternativen entwickelt. In dieser Seminararbeit werden die bekannten Erweiterungen AdaBoost.M1 und AdaBoost.M2 vorgestellt.

2.5.1 AdaBoost.M1

Die Erweiterung AdaBoost.M1 widmet sich dem Mehrklassenproblem. Oftmals sind in einem Datensatz mehr als nur zwei Label die unterschieden werden müssen. Durch die Veränderung der Abbildung h_t (siehe Algorithmus 2) ist AdaBoost.M1 in der Lage für einen bestimmten Eintrag im Datensatz mehrere mögliche Label zu wählen. Allerdings ist für den Algorithmus eine Fehlerqoute des Weaklearners unter 50% wichtig. Ist diese nicht gegeben, wäre bspw. ein Münzwurf ähnlich genau. Daher wird die Schleife abgebrochen sollte sich kein Weaklearner mehr finden der diesem Kriterium entspricht. Eine Wahl des richtigen Labels wird später über den Zusammenschluss aller gewählter Weaklearner entschieden [Cla].

Algorithm 2 Pseudocode AdaBoost.M1 [FS96]

Input: sequence of m examples $\langle (x_1, y_1), ..., (x_m, y_m) \rangle$ with labels $y_i \in Y = \{1, ..., k\}$

weak learning algorithm **WeakLearn**

integer T specifying number of iterations

Initialize $D_1(i) = 1/B$ for all i.

Do for $t = 1, 2, ..., T$

1. Call **WeakLearn**, providing it with mislabel distribution D_t.
2. Get back a hypothesis $h_t : X \to Y$.
3. Calculate the error of h_t: $\epsilon = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$. If $\epsilon_t > 1/2$, then set $T = t - 1$ and abort loop.
4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
5. Update Distribution D_t: $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$
   where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

Output the hypothesis: $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t}$.

2.5.2 AdaBoost.M2

Ebenso wie AdaBoost.M1 wurde auch AdaBoost.M2 dazu entwickelt das Problem mit mehrere Labels zu lösen. Aufbauend auf AdaBoost.M1 ist die für AdaBoost.M2 gravierende Änderung die Weiterentwicklung der Abbildung h_t (siehe Algorithmus 3. Diese bietet ebenso wie AdaBoost.M1 die Möglichkeit mehrere Label auszuwählen. Jedoch wird hierbei für jedes ausgewählte Label auch eine Wahrscheinlichkeit der richtigen Klassifizierung durch den Weaklearner mit angegeben. Im Endgültigen Zusammenschluss aller Weaklearner beeinflusst diese Wahrscheinlichkeit die Wahl eines Labels [Szc05].

Algorithm 3 Pseudocode AdaBoost.M2 [FS96]

Input: sequence of m examples $\langle (x_1, y_1), ..., (x_m, y_m) \rangle$ with labels $y_i \in Y = \{1, ..., k\}$

weak learning algorithm **WeakLearn**

integer T specifying number of iterations

Let $B = \{(i, y) : i \in \{1, ..., m\}, y \neq y_i\}$

Initialize $D_1(i, y) = 1/|B|$ for $(i, y) \in B$.

Do for $t = 1, 2, ..., T$

1. Call **WeakLearn**, providing it with mislabel distribution D_t.
2. Get back a hypothesis $h_t : X \times Y \to [0, 1]$.
3. Calculate the pseudo-loss of h_t: $\epsilon = \frac{1}{2} \sum_{(i,y) \in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$.
4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
5. Update D_t: $D_{t+1}(i, y) = \frac{D_t(i)}{Z_t} \cdot \beta_t^{(1/2)(1+h_t(x_i,y_i)+h_t(x_i,y_i))}$
   where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

Output the hypothesis: $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \big(\log \frac{1}{\beta_t} \big) h_t(x, y)$.

3 Boosting Neuronal Networks

Um Neuronale Netze mit der Methode des Boostings zu verbinden, gibt es bereits einige verschiedene Ansätze. In dieser Seminararbeit beschränken wir uns jedoch auf den Ansatz von Harris Drucker, Robert Schapire und Patrice Simard [DSS]. Hierbei wurden Datensätze mit handgeschriebenen Postleitzahlen des United State Postal Service (USPS) verwendet. Weitere Verfahren können in folgenden Publikationen gefunden werden: [GW99], [MSY$^+$], [SBa], [SBb].

3.1 Das Verfahren

Der gesamte Datensatz an handgeschriebenen Postleitzahlen wird für das Verfahren in drei Teile aufgeteilt: Trainingsdaten für das erste Netz, Validierungsdaten und Testdaten. Im Anschluss kann das erste Netz mit den Trainingsdaten angelernt werden und auf den Validierungsdaten gelernt werden. Das Netz wird dabei stetig verbessert, sodass die Fehlerquote ein vorgegebenes Minimum erreichen muss.
Ist das erste Netz fertig trainiert, muss ein Datensatz für das zweite neuronale Netz erstellt werden. Dafür werden aus dem gesamten Datensatz geringfügig verformte Daten erstellt und vom ersten trainierten Netz klassifiziert. Der zweite Trainingsdatensatz soll nachdem Erstellen genauso groß sein wie er erste Datensatz und eine 50/50 Quote von richtig und falsch klassifizierten Daten des ersten Netzes aufweisen. Dafür wird in einer Schleife der verformte Datensatz klassifiziert. Diese Schleife wird abgebrochen, wenn entweder ein Bild richtig oder falsch klassifiziert wurde. Welcher Fall dabei die Schleife zum Abbruch bringt, wird bei jedem Durchlauf neu entschieden. Nun kann das zweite Netz auf den eben erstellten Daten angelernt und

auf den ursprünglichen Validierungsdaten validiert werden. Durch die schwierigeren Trainigsdaten wird hierbei auch oft eine höhere Fehlerquoute erreicht.
Für das dritte und letzte Netz werden wieder verformte Daten erstellt. Diese werden von beiden Netzen klassifiziert und gelangen nur bei Uneinigkeit der Netze in den dritten Trainingsdatensatz. Danach kann das Netz auf diesen Daten angelernt und auf den Validierungsdaten validiert werden.
Um Ergebnisse über die Gesamtgenauigkeit zu bekommen, wird im Anschluss der Testdatensatz über das Konstrukt klassifiziert. Dabei werden zunächst die ersten beiden Netze zur Klassifizierung benutzt. Sollten diese sich bei der Erkennung einig sein, wird das entsprechende Label zugewiesen. Sind die Netze sich nicht einig, wird das dritte Netz zum Klassifizieren benutzt [DSS].

3.2 Deformation von Daten

Für das oben beschriebene Verfahren werden sehr große Datensätze vorausgesetzt. So werden für das zweite Trainingsset Daten benötigt die im gleichen Verhältnis richtig oder falsch klassifiziert wurden. Bei einer geringen Fehlerrate des ersten Netzes wird die Menge der benötigten Daten immer großer. Um diesen Problem entgegen zu wirken, wird in der Publikation [DSS] von Harris Drucker, Robert Schapire und Patrice Simard ein Verfahren zur Deformation der Bilddaten vorgestellt. Die verarbeiteten Bilder bestehen dazu aus einem Array aus 16x16 oder 20x20 Pixeln. Jeder Pixel eines Bildes hat dabei eine unterschiedliche Intensität zwischen weiß und schwarz. Die folgende Formel stellt hierfür einige Werkzeuge zur Deformation eines Pixles oder des gesamten Bildes dar [DSS].

$$\triangle F_{ij}(x,y) = \left[\frac{\partial F_{ij}(x,y)}{\partial x}\,\frac{\partial F_{ij}(x,y)}{\partial y}\right)$$

$$x\left\{k_1\begin{bmatrix}1\\0\end{bmatrix} + k_2\begin{bmatrix}0\\1\end{bmatrix} + k_3\begin{bmatrix}+y\\x\end{bmatrix} + k_4\begin{bmatrix}y\\x\end{bmatrix} + k_5\begin{bmatrix}-x\\y\end{bmatrix} + k_6\begin{bmatrix}x\\y\end{bmatrix} + k_7\begin{bmatrix}\frac{\partial F_{ij}(x,y)}{\partial x}\\ \frac{\partial F_{ij}(x,y)}{\partial y}\end{bmatrix}\right\}$$

Durch verändern der Werte für k_i können folgende Effekte in Kraft treten:

- k_1 = X-Translation
- k_2 = Y-Translation
- $k_{3,4,5,6}$ = Rotation
- $k_{3,4,5,6}$ = Diagonale Defomation
- $k_{3,4,5,6}$ = Achsen Deformation
- $k_{3,4,5,6}$ = Größenänderung
- k_7 = Intesität

Die Bilder des gesamten Datensätzes werden nun so oft verändert und klassifiziert, bis ein neuer Trainingssatz ensteht. Gleiches geschieht ebenfalls für den dritten Trainingssatz.

3.3 Rejection Rate

Die Rejection Rate ist ein weiteres Kriterium, welches in der gleichen Publikation [DSS] vorgestellt wird. Sie beschreibt den prozentualen Anteil der Daten, die aussortiert werden müssen um eine Fehlerquote von 1% zu erreichen. Diese wird in der Praxisanwendung benötigt, damit möglichst wenige Briefe an falsche Adressaten gesendet werden. Eine manuelle Sortierung aller ausgeworfenen Briefe, wird in der Publikation als besser und günstiger angesehen, als das falsche Versenden von Briefen, welche danach wieder zurückgesendet werden müssen. Sollten die größten Outputs eines Neuronalen Netzes zu nah bei einander liegen, wird dieses Bild verworfen. Einen genauer Wert für die Gleichheit der Outputs wird anhand der Fehlerrate bestimmt. Er wird so gewählt, dass die neue Fehlerrate nur 1% beträgt.

Error Rate	Rejection Rate
4,7%	8,9%

Tabelle 3.1: Ergebnisse vor dem Einsatz den Boosting [DSS]

# des Netzes	Error Rate
1	4,9%
2	5,8%
3	16,9%

Tabelle 3.2: Validierungsergbenisse der einzelnen neuronalen Netze [DSS]

3.4 Ergebnisse

Vor dem Projekt von Harris Drucker, Robert Schapire und Patrice Simard [DSS] gab es bereits ein Neuronales Netz, welches bei USPS im Einsatz war. Dieses brachte inklusive Einsatz von duoble backpropagation folgende Ergebnisse (3.1).

Nach dem Validieren der einzelnen neuronalen Netze auf den Testdaten ergaben sich folgende Fehlerqouten (siehe Tabelle 3.2). Hierfür wurde das Verfahren der Rejection Rate noch nicht angewandt.

In Tabelle 3.3 finden sich die Error Rates und Rejection Rates der Netze, die nacheinander zusammengefügt wurden. Die Rejection Rate gibt hierbei, wie bereis oben in 3.3 beschrieben, eine Quote der auszusortierenden Daten an, damit eine Fehlerqoute von 1,0% erreicht wird.

Weiterhin wurde das vorgestellte Verfahren an Daten des National Institute of Standards an Technology (NIST) angewandt (Siehe Tablle 3.4). Der bereitgestellte Datensatz beinhaltete Bilder mit 220.000 Zahlen, 45.000 Kleinbuchstaben sowie 45.000 Großbuchstaben. da die Error Rate bereits nach dem zweiten trainierten Netz sehr gering war. Hat man sich dazu entschlossen kein drittes Netz anzutrainieren.

Anzahl der Netze	Error Rate	Rejection Rate
nur Netz 1	4,9%	11,5%
Netz 1 und 2	3,9%	7,9%
Netz 1,2,3	3,6%	6,6%

Tabelle 3.3: Ergebnisse nach dem Zusammenfügen der neuronalen Netze [DSS]

	USPS Zahlen	NIST Zahlen	NIST Großbuchstaben	NIST Kleinbuchstaben
Error Rate einzelnes Netz	5,0	1,4	4,0	9,8
Error Rate Boosting Methode	3,6	0,8	2,4	8,1
Rejection Rate einzelnes Netz	9,6	1,0	9,2	29,0
Rejection Rate Boosting Methode	6,6	*	3,1	21,0

* Keine Rejection Rate, da die Error Rate bereits kleiner als 1% ist.

Tabelle 3.4: Ergebnissübersicht aller Datensätze [DSS]

4 Fazit

Das Boosting eines Konstrukts aus neuronalen Netzen kann sich, wie hier beschrieben, postiv auf die Erkennungsrate eines Klassifizieres auswirken. Boosting bietet durch die schlanke Methodik eine einfache und schnelle Erweiterung bereits bestehender Verfahren. Wie in den Ergebnissen gezeigt, kann hierbei die Fehlerquote um 1,4% gesenkt werden. Im gezeigten Beispiel würde somit nur jeder 28. Brief falsch erkannt werden. Mit nur einem neuronalen Netz liegt die Quote bei jedem 20. Brief. Interessanter ist jedoch die Verbesserung der Rejection Rate. In einem praktischen Beispiel wie diesem muss ein Klassifizierungs Algorithmus nicht die höchten Erkennungsraten liefern, sondern dem Unternehmen einen Mehrwert bieten. Sicherlich decken sich die beiden Punkte oftmals. Mit einer geringeren Rejection Rate wird aber auch der Arbeitsaufwand, der benötigt wird um die Aussortierten Briefe manuell zu sortieren, verringert. Dadurch ergibt sich für ein Unternehmen ein weiterer Mehrwert dieses Systems.

List of Algorithms

Tabellenverzeichnis

Literaturverzeichnis

[Ada] *AdaBoost*. https://en.wikipedia.org/wiki/AdaBoost

[Cla] *Classifying with AdaBoost.M1: The Training Error Threshold Myth: Leaes, Antonio; Fernandes, Paulo; Lopes, Lucelene; Assuncao, Joaquim.* https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS17/paper/download/15498/14966

[DSS] Drucker, Harris ; Schapire, Robert ; Simard, Patrice: *Improving Performance in Neural Networks using a Boosting Algorithm.* https://pdfs.semanticscholar.org/77b5/185dafb9e5b884a677a32713e54c253a4e0b.pdf

[FS96] Freund, Yoav ; Schapire, Robert: *Experiments with a New Boosting Algorithm.* http://web.eecs.utk.edu/~leparker/Courses/CS425-528-fall12/Handouts/AdaBoost.M1.pdf. Version: 1996

[GW99] Ghaderi, R. ; Windeattm T: *AdaBoost and neural networks.* https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es1999-12.pdf. Version: 1999

[MSY+] Moghimi, Mohammad ; Saberian, Mohammad ; Yang, Jian ; Li, Li-Jia ; Vasconcelos, Nuno ; Belongie, Serge: *Boosted Convolutional Neural Networks.* https://vision.cornell.edu/se3/wp-content/uploads/2016/08/boosted-convolutional-neural-1.pdf

[Nos] Noss, Jessica: *6.034 Recitation 10: Boosting (Adaboost).* https://www.youtube.com/watch?v=gmok1h8wG-Q

[SBa] Schwenk, Holger ; Bengio, Yoshua: *Boosting Neural Networks*. http://www.iro.umontreal.ca/~lisa/pointeurs/ada-nc.pdf

[SBb] Schwenk, Holger ; Bengio, Yoshua: *Training Methods for Adaptive Boosting of Neural Networks*. https://papers.nips.cc/paper/1335-training-methods-for-adaptive-boosting-of-neural-networks.pdf

[Sch] Schapire, Robert: *Explaining AdaBoost*. http://rob.schapire.net/papers/explaining-adaboost.pdf

[Szc05] Szczot, Magdalena: *Boosting schwacher Lerner*. http://www.informatik.uni-ulm.de/ni/Lehre/SS05/HauptseminarMustererkennung/ausarbeitungen/Szczot.pdf. Version: 2005

B.5. Die Diskriminanzanalyse

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Abteilung Assistenzsysteme und Medizintechnik

Projektgruppe Mobilitäts-Assessments
mit körpernahen Sensoren für zuhause
SoSe17 - WiSe17/18

Seminararbeit:
Die Diskriminanzanalyse

Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autoren: Alexander Thomas

14. Februar 2018

Inhaltsverzeichnis

1 Einleitung

Die Erkennung alltäglicher Bewegungen von beliebigen Personen mit Hilfe von Sensoren ist Gegenstand jüngster Untersuchungen und in verschiedenen Szenarien relevant. Diese reichen vom Gesundheitswesen bis zum Echtzeit-Tracking. Alltägliche Bewegungen sind z.B. Aktivitäten wie Gehen, Laufen, Stehen oder Sitzen.[7]

Um Bewegungen erkennbar machen zu können, müssen die charakteristischen Eigenschaften jeder Bewegung bekannt sein. So ist z.B. Gehen im Gegensatz zu Stehen eine dynamische Bewegung. Diese Sensordaten werden dann mit der vorher kenntlich gemachten Bewegung für die Trainingsphase eines Klassifizierungsalgorithmus verwendet. Das Ergebnis der Trainingsphase ist ein Modell, welches verwendet werden kann, um neue unbekannte Daten zu erkennen.

Wie bereits erwähnt gibt es viele Szenarien, wofür diese Daten verwendet werden können. So könnte im Bereich des Gesundheitswesens eine Aufzeichnung der Bewegungen, die eine Person an einem Tag durchgeführt hat, einem Arzt vorgelegt werden und so zu einer besseren Diagnose helfen. Um zu diesem Endergebnis zu kommen, müssen aber vorher die Daten mithilfe von Klassifizierungsalgorithmen eingelesen und ausgewertet werden.[7]

In dieser Seminararbeit soll deshalb zunächst eine Einführung in die Multivariaten Verfahren bzw. in die Allgemeine Diskrimanzanalyse gegeben werden und anschließend auf die Lineare sowie Quadratische Diskrimianzanalyse bezug genommen werden. Im Anschluss soll eine Überleitung zu den bisherigen Ergebnissen an zuvor gesammelten Bewegungsdaten stattfinden. Zum Schluss sollen die Ergebnisse bewertet und ein Fazit gezogen werden, sowie die weiteren Aussichten der Quadratischen Diskriminanzanalyse, bezogen auf dieses Projekt, besprochen werden.

2 Multivariate Verfahren

Die Multivariate Statistik ist eine Untergruppe der Statistik, die die gleichzeitige Beobachtung und Analyse von mehr als einer Ergebnisvariablen vornimmt. Die Anwendung der multivariaten Statistik ist die multivariate Analyse. Sie beschäftigt sich ebenso mit multivariaten Wahrscheinlichkeitsverteilungen, und zwar insofern dass sie betrachtet:
- Wie diese verwendet werden können, um die Verteilungen der beobachteten Daten darzustellen.
- Wie sie als Teil der statistischen Inferenz verwendet werden können, insbesondere wenn mehrere verschiedene Größen für die gleiche Analyse von Interesse sind.[4, 6]

Bei der Einteilung der multivariaten Verfahren gibt es verschiedene Ansätze. Ein möglicher Ansatz ist die Unterscheidung danach, ob die Verfahren Strukturen geben (z.B. Regressionsanalyse) oder diese nur untersuchen (z.B. Diskriminanzanalyse). Dabei überprüfen die strukturprüfenden Verfahren die Zusammenhänge zwischen Variablen, wenn über den Zusammenhang vor Anwendung der Analyse schon Hypothesen existieren. Bei den strukturegbenden Verfahren liegen noch keine Hypothesen vor.[5]

Strukturen-prüfende Verfahren haben als Ziel die Überprüfung von Zusammenhängen zwischen Variablen zu erkennen. Der Anwender besitzt eine Vorstellung über die Zusammenhänge zwischen Variablen und möchte diese mit den gegebenen Verfahren überprüfen. Verfahren, die diesem Bereich der multivariaten Datenanalyse zugeordnet werden können, sind in der nachfolgenden Tabelle 2.1 einzusehen.[10]

		UNABHÄNGIGE VARIABLE	
		metrisches Skalenniveau	nominales Skalenniveau
ABHÄNGIGE VARIABLE	metrisches Skalennivau	Regressions-analyse	Varianz-analyse
	nominales Skalenniveau	Diskriminanz-analyse, Logistische Regression	Kontingenz-analyse

Tabelle 2.1: Beispiele für Strukturen-prüfende Verfahren

Quelle: Prof. Dr. Klaus Backhaus [10, S. XX]

Strukturen-entdeckende Verfahren, sollen Zusammenhänge zwischen Variablen oder zwischen Objekten erkennen. Der Anwender besitzt zu Beginn der Analyse noch keine Vorstellungen darüber, welche Beziehungszusammenhänge in einem Datensatz existieren. Verfahren, die mögliche Beziehungszusammenhänge aufdecken können, sind die Faktorenanalyse, die Clusteranalyse und die Multidimensionale Skalierung.[10]

Eine weitere Möglichkeit der Einteilung besteht darin, die beobachteten Merkmale zu betrachten:
Sind alle Merkmale, die in die Berechnung mit einfließen, gleichberechtigt (Dependenzanalyse) oder gibt es Merkmale, die wichtiger sind als andere (Interdependenzanalyse). Außerdem kann man die Verfahren danach unterscheiden, ob sie Unterschiede oder Zusammenhänge analysieren[5, 6]

Die Nachfolgende Tabelle 2.2 listet noch einmal verschiedene multivariate Analyseverfahren auf und gibt jeweils ein Anwendungsbeispiel für das jeweilige Verfahren an.

Verfahren	Beispiel
Regressionsanalyse	Abhängigkeit der Absatzmenge eines Produktes von Preis, Werbeausgaben und Einkommen.
Varianzanalyse	Wirkung alternativer Verpackungsgestaltungen auf die Absatzmenge eines Produktes.
Logistische Regression	Ermittlung des Herzinfarktrisikos von Patienten in Abhängigkeit ihres Alters und ihres Cholesterin-Spiegels.
Diskriminanzanalyse	Unterscheidung der Wähler der verschiedenen Parteien hinsichtlich soziodemografischer und psychografischer Merkmale.
Kontingenzanalyse	Zusammenhang zwischen Rauchen und Lungenerkrankung.
Faktorenanalyse	Verdichtung einer Vielzahl von Eigenschaftsbeurteilungen auf zugrundeliegende Beurteilungsdimensionen.
Clusteranalyse	Bildung von Persönlichkeitstypen auf Basis der psychografischen Merkmale von Personen.
Kausalanalyse	Abhängigkeit der Käufertreue von der subjektiven Produktqualität und Servicequalität eines Anbieters.
Multidimensionale Skalierung	Positionierung von konkurrierenden Produktmarken im Wahrnehmungsraum der Konsumenten.
Conjoint Measurement	Ableitung der Nutzenbeiträge alternativer Materialien, Formen oder Farben von Produkten.

Tabelle 2.2: Überblick multivariater Analyseverfahren

Quelle: Prof. Dr. Klaus Backhaus [10, S. XXVII]

3 Diskriminanzanalyse Allgemein

Die Diskriminanzanalyse ist als strukturprüfendes Verfahren ein multivariates Verfahren zur Analyse von Gruppenunterschieden. Sie untersucht also die Unterschiedlichkeit zweier oder mehrerer Gruppen hinsichtlich mehrerer Variablen. Nachdem für eine Menge von Elementen die Zusammenhänge zwischen der Gruppenzugehörigkeit der Elemente und ihren Merkmalen analysiert wurden, lässt sich eine Prognose der Gruppenzugehörigkeit von neuen Elementen vornehmen. Derartige Anwendungen finden sich z.B. bei der Kreditwürdigkeitsprüfung (Einstufung von Kreditkunden einer Bank in Risikoklassen) oder bei der Personalbeurteilung (Einstufung von Außendienstmitarbeitern nach erwartetem Verkaufserfolg). [5, 10]

Durch die Diskriminanzanalyse wird anhand verschiedener Konto-Daten (z.B. Verhältnis von Soll zu Haben) die Diskriminanzfunktion berechnet. Es werden "gute" von "schlechten" Konten getrennt und andererseits der Diskriminanzpunkt bestimmt. Nun kann für jedes neue auszuwertende Konto die Diskriminanzfunktion berechnet und das Konto durch Berechnung des Abstands vom Diskriminanzpunkt in die jeweilige Gruppe eingeordnet werden. [5]

3.1 Vorgehen Diskriminanzanalyse

Die Durchführung einer Diskriminanzanalyse lässt sich in sechs Teilschritte zerlegen, wie das folgende Ablaufdiagramm in Abbildung 3.1 darstellt:

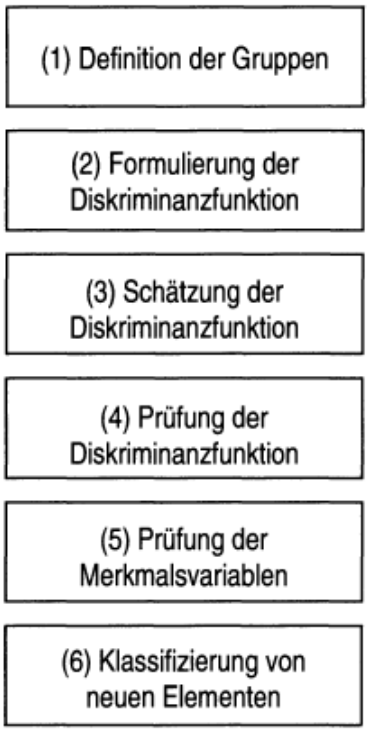


Abbildung 3.1: Ablauf Diskriminanzanalyse

Quelle: Prof. Dr. Klaus Backhaus [10, S. 149]

1. Definition der Gruppen

Die Diskriminanzanalyse beginnt mit der Definition der Gruppen. Diese kann sich unmittelbar aus dem Anwendungsproblem ergeben (z.B. Gruppierung von Käufern nach Produktmarken). Sie kann aber auch das Resultat einer vorangegangenen Analyse sein. Mit der Definition der Gruppen ist auch die Festlegung der Anzahl der Gruppen, die die Diskriminanzanalyse berücksichtigen soll, verbunden.[10]

2. Formulierung der Diskriminanzfunktion

Im nächsten Schritt ist eine Diskriminanzfunktion oder auch Trennfunktion zu formulieren und zu schätzen, die dann eine optimale Trennung zwischen den Gruppen

und eine Prüfung der diskriminatorischen Bedeutung der Merkmalsvariablen ermöglicht, welches für den späteren Verlauf von Bedeutung ist.[10] Die allgemeine (kanonische) Diskriminanzfunktion hat folgende Form:

$$Y = b_0 + b_1 X_1 + b_2 X_2 + ... + b_J X_J$$

Y = Diskriminanzvariable
X_j = Merkmalsvariable j (j=1, 2, ..., J)
b_j = Diskriminanzkoeffizient für Merkmalsvariable j
bo = Konstantes Glied

Die Parameter b_0 und b_j (j=1, 2, ..., J) sind auf Basis von Daten für die Merkmalsvariablen zu schätzen. Für jedes Element i(i=1, ..., I_g) einer Gruppe g (g=1, ... ,G) mit den Merkmalswerten X_{jgi} (j=1, ..., J) liefert die Diskriminanzfunktion einen Diskriminanzwert Y_{gi}.[10]

Jede Gruppe g lässt sich durch ihren mittleren Diskriminanzwert, der als Centroid (Schwerpunkt) bezeichnet wird, beschreiben:[10]

$$Y_g = \frac{1}{I_g} \sum_{i=1}^{I_g} Y_{gi}$$

Die Unterschiedlichkeit zweier Gruppen $g = A, B$ lässt sich damit durch die Differenz messen:[10]

$$|Y_A - Y_B|$$

3. Schätzung der Diskriminanzfunktion

Die Schätzung der Diskriminanzfunktion bzw. der unbekannten Koeffizienten b_j in der Diskriminanzfunktion erfolgt so, dass sie im möglichen Fall optimal zwischen den gegebenen Gruppen trennt. [10]

Als Referenz für die Unterschiedlichkeit von Gruppen wurde bereits die Distanz zwischen den Schwerpunkten eingeführt. Ein besseres Maß der Unterschiedlichkeit

(Diskriminanz) erhält man aber, wenn auch die Streuung der Gruppen berücksichtigt wird:

$$\Gamma = \frac{Streuung\ zwischen\ den\ Gruppen}{Streuung\ in\ den\ Gruppen}$$

Genauer betrachtet ergibt sich:[10]

$$\Gamma = \frac{\sum_{g=1}^{G} I_g(Y_g - Y)^2}{\sum_{g=1}^{G}\sum_{i=1}^{I_g}(Y_{gi} - Y_g)^2} = \frac{SS_b}{SS_w}$$

Die Streuung zwischen den Gruppen wird durch die quadrierten Abweichungen der Gruppencentroide vom Gesamtmittel gemessen und kann so für beliebig viele Gruppen genutzt werden. Um unterschiedliche Gruppengrößen zu berücksichtigen, werden die Abweichungen jeweils mit der Gruppengröße I_g multipliziert.
- Die Streuung in den Gruppen wird durch die quadrierten Abweichungen der Gruppenelemente vom jeweiligen Gruppencentroid gemessen.
- Die Streuung zwischen den Gruppen wird durch SS_b (Sum of Squares between) dargestellt
- Die Streuung in den Gruppen wird durch SS_w (Sum of Squares within) symbolisiert.[10]

Falls mehrer als zwei Gruppen vorliegen, können mehr als eine Diskriminanzfunktion ermittelt werden. Bei G Gruppen lassen sich maximal $G-1$ Diskriminanzfunktionen bilden. Die Anzahl der Diskriminanzfunktionen kann jedoch nicht größer sein, als die Anzahl J der Merkmalsvariablen, sodass die maximale Anzahl von Diskriminanzfunktionen durch $Min\{G-I, J\}$ erreicht ist.[10]

4. Prüfung der Diskriminanzfunktion

Die Trennkraft einer Diskriminanzfunktion lässt die Unterschiedlichkeit der Gruppen messen. Eine Möglichkeit zur Prüfung der Diskriminanzfunktion ist, die Untersuchungsobjekte mit deren tatsächlicher Gruppenzugehörigkeit zu vergleichen.[10]

Um die Klassifikationsfähigkeit einer Diskriminanzfunktion richtig beurteilen zu können, muss deren Trefferquote mit derjenigen Trefferquote verglichen werden,

die man bei einer rein zufälligen Zuordnung der Elemente erreichen würde.[10]

Eine bereinigte Trefferquote erhält man, indem man die verfügbare Stichprobe zufällig in zwei Unterstichproben aufteilt, eine Lernstichprobe und eine Kontrollstichprobe. Die Lernstichprobe wird zur Schätzung der Diskriminanzfunktion verwendet. Diese Methode ist allerdings nur dann sinnvoll, wenn eine ausreichend große Stichprobe zur Verfügung steht, da mit einer kleiner werdenden Größe der Lernstichprobe die Zuverlässigkeit der geschätzten Diskriminanzkoeffizienten abnimmt.[10]

5. Prüfung der Mermalsvariablen

Wichtig könnte zudem sein, die Merkmalsvariablen in der Diskriminanzfunktion beurteilen zu können. Zunächst, um die Unterschiedlichkeit der Gruppen zu erklären, und außerdem, um unwichtige Variablen aus der Diskriminanzfunktion zu entfernen. Hier kann der allgemein übliche F-Test verwendet werden. Das Ergebnis entspricht dann einer einfachen Varianzanalyse zwischen Gruppierungs- und Merkmalsvariable.[10]

Sind mehrere Diskriminanzfunktionen vorhanden, so gibt es für jede Merkmalsvariable mehrere Diskriminanzkoeffizienten. Um hier die diskriminatorische Signifikanz einer Merkmalsvariablen bezüglich aller ihrer Diskriminanzfunktionen zu beurteilen, sind die mit den Eigenwertanteilen gewichteten absoluten Werte der Koeffizienten einer Merkmalsvariablen zu addieren.[10]

6. Klassifizierung von neuen Elementen

Für die Klassifikation von neuen Elementen sind grundsätzlich drei verschiedene Konzepte vorhanden: Klassifizierung mittels Distanzen, mittels Klassizierungsfunktionen und über Wahrscheinlichkeitsberechnungen.[10]

Hier geht man folgendermaßen vor: ein Objekt a wird anhand seiner gemessenen Merkmalsausprägungen $a_1, a_2, ..., a_p$ in die Gruppe klassifiziert, zu deren Gruppenmittelpunkt es den kleinsten Abstand hat. Ist nur eine Diskriminanzfunktion vor-

handen, so wird ein Objekt derjenigen Gruppe zugeordnet, für die die euklidische Distanz minimal ist:[10]

$$|y(a) - y_j| = \sqrt{y(a)^2 - y_j^2}$$

Die von R.A. Fisher entwickelten Klassifizierungsfunktionen bilden eine Methode, um die Klassifizierung direkt auf Basis der Merkmalswerte (ohne Verwendung von Diskriminanzfunktionen) durchzuführen. Die Klassifizierungsfunktionen sind jedoch nur dann nutzbar, wenn gleiche Streuung in den Gruppen angenommen werden kann, also wenn die Kovarianzmatrizen der Gruppen annähernd identisch sind. Diese Methode wird auch als lineare Diskriminanzfunktion bezeichnet und in Abschnitt 4.1 näher behandelt. [10]

Gemäß dem Distanzkonzept wird ein Element i in diejenige Gruppe g eingeordnet, der es am nächsten liegt. Hier werden vorwiegende die quadrierten Distanzen verwendet:[10]

$$D_{ig}^2 = (Y_i - Y_g)^2 \qquad (g = 1, ..., G)$$

Bei mehreren Diskriminanzfunktionen wird die quadrierte euklidische Distanz im K -dimensionalen Diskriminanzraum zwischen dem Element i und dem Centroid der Gruppe g herangezogen:[10]

$$D_{ig}^2 = \sum_{k=1}^{K} (Y_{ki} - Y_{kg})^2 \qquad (g = 1, ..., G)$$

Das Wahrscheinlichkeitskonzept, baut auf dem Distanzkonzept auf. Es ermöglicht, die Berücksichtigung von (ungleichen) A-priori-Wahrscheinlichkeiten. Auch ermöglicht es die Berücksichtigung von (ungleichen) „Kosten“ der Fehlklassifikation. Ohne diese Erweiterungen würde es zu den gleichen Ergebnissen, wie das Distanzkonzept führen. Es wird nach folgendem Prinzip durchgeführt: Ordne ein Element i derjenigen Gruppe g zu, für die die Wahrscheinlichkeit $P(g|Y_i)$ maximal ist. [10]

Dabei bezeichnet $P(g|Y_i)$ die Wahrscheinlichkeit für die Zugehörigkeit von Element

i mit Diskriminanzwert Y_i zu Gruppe g (g=I, ..., G). [10]

Die Klassifizierungswahrscheinlichkeiten werden als A-posteriori-Wahrscheinlichkeiten bezeichnet. Zu ihrer Berechnung wird das Bayes-Theorem angewendet:[10]

$$P(g|Y_i) = \frac{P(Y_i|g)P_i(g)}{\sum_{g=1}^{G} P(Y_i|g)P_i(g)}$$

$P(g|Y_i)$ = A-posteriori-Wahrscheinlichkeit
$P(Y_i|g)$ = Bedingte Wahrscheinlichkeit
$P_i(g)$ = A-priori-Wahrscheinlichkeit

Die bedingte Wahrscheinlichkeit $P(Y_i|g)$ gibt an, wie wahrscheinlich ein Diskriminanzwert Y_i für das Element i wäre, wenn dieses zu Gruppe g gehören würde. Sie lässt sich durch Transformation der Distanz D_{ig} ermitteln.[10]

In Tabelle 3.1 werden noch einmal die verschiedenen Klassifizierungskonzepte aufgezeigt und untereinander verglichen.

	Klassifiz.-Funktionen	Distanz-Konzept	Wahrsch.-Konzept
Unterschiedliche A-priori-Wahrscheinlichkeiten	ja	nein	ja
Unterschiedliche Kosten der Fehlklassifikation	nein	nein	ja
Berücksichtigung ungleicher Streuungen in den Gruppen	nein	ja	ja
Unterdrückung irrelevanter Diskriminanz-funktionen	nein	ja	ja

Tabelle 3.1: Vergleich der Klassifizierungskonzepte

Quelle: Prof. Dr. Klaus Backhaus [10, S. 185]

3.2 Prüfung der Klassifikation

Zur Prüfung der Klassifikation haben sich mehrere Verfahren etabliert, die hier schnell aufgegriffen und erklärt werden sollen.

Resubstitution

Die Schätzung und Prüfung der Daten werden mit dem Ausgangsdatensatz durchgeführt, jedoch ohne dass aus diesem Daten entfernt oder hinzugefügt werden. Die Fehlerrate, die man bei diesem Test erhält, ist die gesuchte Fehlerrate. Allerdings neigt die Resubstitution zur optimistischen Unterschätzung der Fehlerrate, je kleiner die Stichprobe ist.[5]

Vorgehensweise:

1. Bestimme die Zuordnungsregeln
2. Wende die Zuordnungsregeln auf alle Elemente der Stichprobe an und bestimme die Anzahl der falsch zugeordneten Datensätze

Train-and-Test

Der Ausgangsdatensatz wird aufgeteilt in:

– Trainingsdatensatz - dieser enthält die Daten fur die Schätzung

– Testdatensatz - hierin sind 20-30% der Ausgangsdaten enthalten

Die gesuchte Fehlerrate ergibt sich hier aus der Fehlerrate des Testdatensatzes. Sie wird im Mittel richtig geschätzt.[5] Nachteilig ist, dass man zwei Stichproben benötigt. Dadurch wird der Umfang des Trainingsdatensatzes kleiner. Dies ist relevant, da bei der Diskriminanzanalyse gilt: Je mehr Daten vorhanden sind, desto besser können die Zuordnungsregeln bestimmt werden.[5]

Vorgehensweise:

1. Bestimme die Zuordnungsregeln aufgrund des Trainingsdatensatzes
2. Wende die Zuordnungsregeln auf alle Elemente der Teststichprobe an und be-

stimme die Anzahl der falsch zugeordneten Datensätze

Cross-Validation

Hierbei werden die Ausgangsdaten in m (gleichgroße) Stichproben aufgeteilt. Dabei gelten bei jedem Durchgang

- $(m-1)$ als Trainingsdatensatz und

- 1 als Testdatensatz

Die Fehlerrate ergibt sich hier aus dem Durchschnitt aller Testdaten-Fehlerraten.[5]

Leaving-One-Out

Bei dieser Test-Methode nimmt man jeweils einen Datensatz aus den Ausgangsdaten und verwendet diesen als Test-Datensatz. Die gesuchte Fehlerrate findet man durch Durchschnittsbildung der Fehlerraten der einzelnen Stichproben.

Vorgehensweise:

1. Nimm den i-ten Fall als Teststichprobe, die restlichen $n-1$ Fälle bilden die Trainingsstichprobe

2. Bestimme die Zuordnungsregeln mit der Trainingsstichprobe

3. Wende die Zuordnungsregeln auf den einen Fall der Teststichprobe an; stelle fest, ob er richtig zugeordnet wurde

4. Führe die Schritte 1-3 für $i = (1, 2, ..., n)$ durch und zähle die Anzahl der falsch zugeordneten Fälle. [5]

Bootstrap

Hierbei erzeugt man m Testdatensätze, die die gleiche Anzahl an Daten enthalten wie der Ausgangsdatensatz. Jeder Testdatensatz wird hier unter Verwendung folgender Regeln erzeugt:

– Entferne zufällig einzelne Elemente (ca. $\frac{1}{e} \equiv 37\%$)

– Füge zufällig bereits vorhandene Elemente hinzu

Mit diesen Datensätzen wird nun jeweils eine Diskriminanzanalyse durchgeführt. Die Fehlerrate ergibt sich aus dem Durchschnitt aller m Fehlerraten. [5]

4 Lineare und Quadratische Diskriminanzanalyse

4.1 Lineare Diskriminanzanalyse

Die lineare Diskriminanzanalyse ergibt sich für den Spezialfall von klassenweise identischen Kovarianzmatrizen $\Sigma_r = \Sigma$ mit r = 1, ..., k. [12, 9]

Bei der linearen Diskriminanzanalyse werden die folgenden Annahmen zugrunde gelegt:[3, 8]
(L1) Die Verteilungen innerhalb der Klassen sind Normalverteilungen, wobei unterschiedliche Erwartungswerte μ_i vorausgesetzt werden, aber die Kovarianzmatrizen Σ als identisch für alle Klassen angenommen werden.
(L2) Die Fehlklassifikationskosten sind gleich.
(L3) Die a-priori-Wahrscheinlichkeiten können unterschiedlich sein.

Die Lineare Diskriminanzfunktion sieht wie folgt aus: [12]

$$d_r(x) = -\frac{1}{2}(x - \mu_r)^T \Sigma^{-1}(x - \mu_r) + \log(p(r))$$

Für gleiche a-priori-Wahrscheinlichkeiten $p(1) = ... = p(k)$ wird das Individuum i derjenigen Klasse zugeordnet, deren quadratische Mahalanobis Distanz minimal ist. Im Gegensatz zur QDA müssen nur wechselseitige Differenzen zwischen den Diskriminanzfunktionen der Klassen geschätzt werden. Das heißt, die Anzahl der zu schätzenden Parameter ist deutlich niedriger. [12]

4.2 Quadratische Diskriminanzanalyse

Wird die Annahme der LDA (L1) durch die folgende Annahme (Q1) ersetzt, spricht man von quadratischer Diskriminanzanalyse:[3, 8]
(L1) Die Verteilungen innerhalb der Klassen sind Normalverteilungen, wobei unterschiedliche Erwartungswerte μ i vorausgesetzt werden, aber die Kovarianzmatrizen Σ als identisch für alle Klassen angenommen werden.
(Q1) Für die Verteilungen innerhalb der Klassen werden Normalverteilungen mit unterschiedlichen Erwartungswerten μ und unterschiedlichen Kovarianzmatrizen Σ_i angenommen.

In die logarithmierte Form der Bayes-Regel eingesetzt, erhält man folgende Diskriminanzfunktion:

$$d_r(x) = -\frac{1}{2}(x-\mu_r)^T\Sigma_r^{-1}(x-\mu_r) - \frac{1}{2}\log(|\Sigma_r|) + \log(p(r))$$

In der Praxis sind die Parameter der Normalverteilung meist unbekannt, sie müssen deshalb aus der Lernstichprobe geschätzt werden. Klassischerweise werden folgende unverzerrte Schätzer verwendet:[12]
geschätzte a-priori-Wahrscheinlichkeit der Klasse $\mathrm{p(r)} = \mathrm{n_r/n}$
geschätzter Mittelpunkt der Klasse $\mu_r = \mathrm{x_r}$
geschätzte Kovarianzmatrix der Klasse r: $\Sigma_r = \mathrm{S_k}$

Der Vorteil der quadratischen Diskriminanzanalyse ist, dass weniger Annahmen als in der linearen oder diagonalen linearen Diskriminanzanalyse vorausgesetzt werden. Es werden keinerlei Aussagen über die Kovarianzmatrizen Σ_r getroffen. Dies führt zu einer großen Anzahl zu schätzender Parameter für die verschiedenen Kovarianzmatrizen. Somit ist diese Methode nur für Datensätze mit vielen Beobachtungen in jeder Klasse geeignet.[12, 9]

Die Nachfolgende Tabelle 4.1 listet mögliche Anwendungsgebiete der QDA auf und zeigt, dass die Diskriminanzanalyse auch im freien Markt breit vertreten sein kann und für verschiedenste Gebiete nutzbar ist.

Problemstellung	**Gruppierung**	**Merkmalsvariablen**
Prüfung der Kreditwürdigkeit	Risikoklasse: – hoch – niedrig	Soziodemographische Merkmale (Alter, Einkommen, etc.) Anzahl weiterer Kredite, Beschäftigungsdauer, etc.
Auswahl von Außendienstmitarbeitern	Verkaufserfolg: – hoch – niedrig	Ausbildung, Alter, Perönlichkeitsmerkmale, körperliche Merkmale etc.
Analyse der Markenwahl beim Autokauf	Marke: – Mercedes – BMW – Audi, etc.	Wünsche zu Eigenschaften von Autos, z.B.: Aussehen, Straßenlage, Höchstgeschwindigkeit, Wirtschaftlichkeit etc.
Wähleranalyse	Partei – CDU – SPD – FDP – Grüne	Einstellung zu politischen Themen wie Abrüstung, Atomernergie, Tempolimit, Besteuerung, Wehrdienst, Mitbestimmung etc.
Diagnose von Atemnot bei Neugeborenen	Überleben: – ja – nein	Geburtsgewicht, Geschlecht, pH-Wert des Blutes, postmenstruales Alter der Mutter etc.
Erfolgsaussichten von neuen Produkten	Wirtschaftl. Erfolg: – Gewinn – Verlust	Neuigkeitsgrad des Produktes, Marktkenntnis des Unternehmens, Preis/Leistungs-Verhältnis, technolog. Know-how etc.
Analyse der Diffusion von Innovationen	Adoptergruppen: – Innovatoren – Imitatoren	Risikofreudigkeit, soziale Mobilität, Einkommen, Statusbewußtsein etc.

Tabelle 4.1: Mögliche Anwendungsgebiete der QDA

Quelle: Heß [5, S. 24]

5 Quadratische Diskriminanzanalyse in Bezug zu MAMKSFZ

In diesem Abschnitt werde ich die bisherigen Ergebnisse der Quadratischen Diskriminanzanalyse (QDA) aufzeigen und Erkentnisse aus den Resultaten ziehen. Die Ergebnisse sind einerseits mit Matlab 2016b, aber auch mit Python erstellt worden. Der Wechsel von Matlab auf Python erfolgte, da Python leichtgewichtiger und schneller als Matlab ist. Außerdem ist es für mich intuitiver, da ich bereits Erfahrung mit der Skriptsprache Python habe.

Für Python sind jedoch zusätzliche Pakete notwendig, die nachträglich installiert werden müssen, diese sind wie folgt aufgelistet: [11]
- pandas
- matplotlib
- sklearn
- datetime
- pickle

Die ersten Daten die von mir angelernt worden sind, waren die Assessment-Daten der alten Projektgruppe MAMKS. Die QDA hat den Vorteil, dass sie in Matlab relativ schnell angelernt werden konnte, sodass schnell Ergebnisse erzielt werden konnten. Andere Verfahren wie K-Nearest-Neighbors oder Boosted-Decision-Trees sind mit Matlab sehr langsam anzulernen, selbst auf dem Cluster der Univerität Oldenburg. Die Genauigkeiten der ersten Tests, die von Matlab ausgegeben wurden, lagen bei 89,91%. Bei weiteren Tests, die mit den Assessment-Daten und noch zusätzlich selbstgesammelten Daten vorgenommen wurden, sanken die Ergebnisse

jedoch auf nur noch rund 71% ab.

Die nachstehende Tabelle 5.1 stellt die Aktivitäten in Static, Dynamic und Transition dar und zeigt die Erkennungsrate (Acc) auf eben diesen Aktivitäten. Außerdem zeigt sie den Anteil der Aktivitäten an der Gesamtanzahl gemessen in Prozent. Matlab hat hier zusätzlich eine Cross-Validation von 5 genutzt.

Static (%)			Dynamic (%)			Transition (%)		
Aktivität	Acc	Anteil	Aktivität	Acc	Anteil	Aktivität	Acc	Anteil
SIT	95%	42,25	WALK	84%	21,91	STAND_SIT	65%	1,364
STAND	89%	30,56	TURN_AROUND_LEFT	54%	0,137	LIE_ON_BACK_LIE_ON_SIDE	59%	0,138
LIE_ON_BACK	80%	0,574	TURN_AROUND_RIGHT	45%	0,060	SIT_STAND	53%	1,268
LIE_ON_SIDE	78%	0,211	JUMP	42%	0,385	STAND_LIE	37%	0,170
			STAIR_CLIMB	28%	0,576	LIE_ON_SIDE_LIE_ON_BACK	35%	0,012
			BEND_OVER_OR_SQUAT	28%	0,092	LIE_SIT	20%	0,166
			WALK_BACKWARDS	0%	0,100	SIT_LIE	16%	0,002

Tabelle 5.1: Erste Zwischenergebnisse der QDA in Matlab

In der Tabelle 5.1 ist unschwer zu erkennen, dass die Aktivitäten „Sit“, „Stand“ und „Walk“ mit Abstand am häufgisten vertreten sind. Dies spiegelt sich auch in den Ergebnissen der QDA wieder. Denn diese kann, wie in Abschnitt 4.2 nachzulesen, am besten Elemente klassifiziern, die in großer Zahl vertreten sind. Dies bedeutet unweigerlich, dass Elemente die nicht ausreichend zur Verfügung stehen, schlecht von der QDA eingeordnet werden.

Als nächsten Schritt galt es die Senior-Home-Daten anzulernen und die Ergebnisse mit denen, der Assessment-Daten zu vergleichen. Hier sind drei verschiedene Datensätze entstanden, die dann mit Python angelernt worden sind:
- 1b_new: beinhaltet 12 Aktivitäten
- 2_new: beinhaltet 9 Aktivitäten
- 3_new: beinhaltet 9 Aktivitäten

Die Genauigkeiten der verschiedenen Datensätze sind in der Abbildung 5.1 einsehbar. Die Ergebnisse sind mit 79-80% ausgezeichnet und damit schlechter als die

Genauigkeiten bei den Datensätzen der alten Projektgruppe. Dies hängt mit großer Wahrscheinlichkeit an einerseits schlecht gelabelten Daten, sowie mit der größeren Anzahl an Aktivitäten zusammen.

```
========== RESTART: C:\Users\Alex\Desktop\Python QDA\
Einlesen Trainingsdaten beendet:  0:01:00.643035
Anlernen QDA/LDA beendet:  0:00:46.280976
Genauigkeit: 80.30644591871258 %
Genauigkeit: 75.09921889939835 %
Predict Scores beendet:  0:00:07.317709
>>>
========== RESTART: C:\Users\Alex\Desktop\Python QDA\
Einlesen Trainingsdaten beendet:  0:01:03.061690

Warning (from warnings module):
  File "C:\Users\Alex\AppData\Local\Programs\Python\Py
klearn\discriminant_analysis.py", line 442
    UserWarning)
UserWarning: The priors do not sum to 1. Renormalizing
Anlernen QDA/LDA beendet:  0:00:48.398303
Genauigkeit: 79.89433707991627 %
Genauigkeit: 74.35320926088941 %
Predict Scores beendet:  0:00:08.719194
>>>
========== RESTART: C:\Users\Alex\Desktop\Python QDA\
Einlesen Trainingsdaten beendet:  0:01:00.978884
Anlernen QDA/LDA beendet:  0:00:46.312037
Genauigkeit: 79.92869949517107 %
Genauigkeit: 74.33587648947932 %
Predict Scores beendet:  0:00:07.612476
```

Abbildung 5.1: Senior-Home-Studie Ergebnisse der QDA in Python

In der Abbildung 5.1 sind die Ergebnisse der drei Datensätze zu sehen, in der Reihenfolge wie sie oben aufgelistet worden sind. Die erste Genauigkeit ist das Ergebnis der QDA, sowie zusätzlich als Vergleich die Genauigkeit der Linearen Diskriminanzanalyse (LDA).

Bei der Anwendung des antrainierten Klassifizierers, kommt es, gegeben durch die schlechte Klassifizierung, zu Fehlerhaften Einordnungen von unbekannten Daten. Es wird bei den Vorhersagen von neuen Daten zu großer Wahrscheinlichkeit „Sit“, „Stand“ oder „Walk“ ausgegeben. Selbst dann, wenn man händisch Daten vorher-

sagen lassen möchte, die definitiv „Treppe steigen“, „Springen“, oder andere Aktivitäten darstellen.

Durch verschiedene Optimierungen, wie z.B. das Downsampling oder Upsampling der Daten, konnten die Ergebnisse leider nur geringfügig verbessert werden. Viel bessere Resultate erzielte jedoch der K-Nearest-Neighbor mit dem Downsampling.

Die Daten aus der alten Projektgruppe haben, anders als die Senior-Home-Daten, zeitlich vordefinierte Abläufe mit definierten Start- und Endzeitpunkten und geregelte Aktivitäten, die die Probanden durchführen mussten. Dies macht es einfacher diese Daten zu labeln. Die von uns aufgenommenen Daten, sind bei den Senioren zu Hause enstanden und sind in ihren Aktivitäten nicht eingeschränkt oder vordefiniert. Das bedeutet, dass beim Labeln Fehler passieren können, da manche Aktivitäten nicht einwandfrei zugeordnet werden konnten. Dies hat wiederrum Einfluss auf das Anlernen mit der QDA.

Nach den bisherigen Ergebnisse werden die Resultate der QDA immer „schlechter“, bzw. nehmen in der Gesamterkennungsrate ab. Das liegt vor allem auch daran, dass die QDA stark abhängig von der Anzahl an Beobachtungen in jeder Klasse ist, sowie von korrekt gelabelten Daten.

Als zusätzlicher Schritt wird deshalb versucht die Qualität der gelabelten Daten noch zu verbessern, um evtl. noch bessere Ergebnisse erhalten zu können. Jedoch liegt das Hauptaugenmerk unserer Arbeit auf dem K-Nearest-Neighbor und dem Decision-Tree Algorithmus, da diese mit Abstand die besten Ergebnisse erzielen. Sodass anzumerken ist, dass die Arbeit an der QDA eher in den Hintergrund gerückt wird.

6 Related Work

Bei meiner Suche nach Verwandten Arbeiten zu Bewegungsaufnahmen mit Sensoren und anschließender Klassifikation mit der Diskriminanzanalyse, habe ich nahezu nichts an Literatur finden können. Das kann unter anderem daran liegen, dass die Diskriminanzanalyse nicht sonderlich gut für diese Art Daten geeignet ist oder aber andere Verfahren deutlich bessere Ergebnisse präsentieren als die Diskriminanzanalyse.

Spitzenreiter sind dort z.B. der K-Nearest-Neighbor oder Decision Trees, welche so gut wie in jeder Arbeit vorkommen, die mit Bewegungsaufnahmen zu tun haben. So zeigen z.B. Albert Hein [1] und Allen Y. Yang [2], dass sich KNN und Decision-Trees hervorragend eignen, Bewegungsabläufe zu klassifizieren und solide Ergebnisse liefern.

Eine weitere Ausarbeitung von Milker [7] zeigt die Bewegungserkennung mit dem Bayes. Da die Diskriminanzfunktion auf Bayes aufbaut, ist ein Vergleich mit der QDA möglich. Jedoch zeigt sich hier, wie vermutet, ein eher schlechtes Ergebnis bei der Auswertung, denn die Erkennungsrate der Bewegungmuster liegt hier nur bei 70%, wohingegen der Decision-Tree-Algorithmus bei 98% Gesamtgenauigkeit liegt.[7]

Dies unterstreicht meine bisherigen Ergebnisse mit der Quadratischen Diskriminanzfunktion und macht ein weiteres Auseinandersetzen mit der QDA, bezogen auf die Zeitliche Begrenzung der Projektgruppe, eher nicht nötig.

7 Fazit

Diese Arbeit untersuchte die Bewegungserkennung mit Hilfe der Quadratischen Diskriminanzanalyse. Es wurde versucht die Diskriminanzfunktion in die Multivariaten Verfahren einzuordnen und den Ablauf des Algorithmus zu erklären. Anschließend wurden die bisherigen Ergebnisse vorgestellt und untereinander verglichen, sowie mögliche Erklärungen zu den Ergebnissen erörtert.

Die geringe Genauigkeit, die in dieser Untersuchung auftrat, lässt sich entweder auf schlecht gelabelte Daten zurückführen oder auf die zu wenig vorhandenen Aktivitäten, wie „Treppe steigen“ oder „Hinsetzen“. Leider ist ein großes Ungleichgewicht in den Daten vorhanden, was sich wahrscheinlich in den Ergebnissen des Klassifikators kenntlich macht. Auch andere Publikationen zeigen, dass die Diskriminanzanalyse leider nicht „gut“ genug ist, um eine weitere Auseinandersetzung mit diesem zu rechtfertigen.

Zusammenfassend lässt sich sagen, dass die Quadratische Diskriminanzanalyse ein spannender Algorithmus ist, der für viele verschiedene Sachverhalte nutzbar ist, jedoch für den weiteren Gebrauch der Projektgruppe MAMKSFZ nicht geeignet ist. Die bisher erzeugten Ergebnisse sind nicht relevant genug, um sich weiter mit diesem Algorithmus beschäftigen zu können. Dennoch hat es Spaß gemacht, sich mit diesem Algorithmus auseinanderzusetzen und seine verschiedenen Einsatzfelder kennen zu lernen.

Literatur

[1] Frank Krüger Albert Hein Prof. Dr.-Ing. Thomas Kirste. *Mit Sensordaten und Maschinellem Lernen Bewegungen erkennen.* https://jaxenter.de/sensordaten-und-erkannte-aktivitaeten-52645. Eingesehen am 14.02.2018.

[2] Shankar Sastry Ruzena Bajcsy Philip Kuryloski Roozbeh Jafari Allen Y. Yang Sameer Iyengar. *Distributed Segmentation and Classification of Human Actions Using a Wearable Motion Sensor Network.* https://pdfs.semanticscholar.org/439e/ee97dc7813e6d367ed144d51c5180cd1c556.pdf. Eingesehen am 14.02.2018.

[3] *Allgemeine Diskriminanzanalyse.* https://www.statistik.tu-dortmund.de/fileadmin/user_upload/Lehrstuehle/Datenanalyse/Wissensentdeckung/Wissensentdeckung-Li-5_2x2.pdf. Eingesehen am 01.02.2018.

[4] T. W. Anderson. *An Introduction to Multivariate Statistical Analysis (Wiley Series in Probability and Statistics).* 3. Aufl. Wiley-Interscience, 2003.

[5] Lena-Luisa Heß. *Diskriminanzanalyse.* http://members.tripod.com/lena_hess/data/skripte/Diplomarbeit.pdf/. Eingesehen am 07.02.2018.

[6] Johannes Müller Matthias Rudolf. *Multivariate Verfahren.* http://www3.hogrefe.de/fileadmin/redakteure/hogrefe_de/Psychlehrbuchplus/Multivariate_Verfahren/04_Diskriminanzanalyse/Bruecken_in_Arbeit.pdf. Eingesehen am 03.02.2018.

[7] Sven Milker. *Bewegungserkennung mit Smartphones mittels deren Sensoren.* https://west.uni-koblenz.de/sites/default/files/BewegungserkennungmitSmartp final.pdf. Eingesehen am 14.02.2018.

[8] U. Mortensen. *Klassifikations- und Diskriminanzanalyse*. `http://www.uwe-mortensen.de/DiskrimIndexTestTest.pdf`. Eingesehen am 12.02.2018.

[9] Michael Nothnagel. *Klassikationsverfahren der Diskriminanzanalyse*. `https://portal.ccg.uni-koeln.de/ccg/assets/templates/doc/statgen/publ/diploma/mnda.pdf/`. Eingesehen am 04.02.2018.

[10] Prof. Dr. Wulff Plinke Prof. Dr. Rolf Weiber (auth.) Prof. Dr. Klaus Backhaus Prof. Dr. Bernd Erichson. *Multivariate Analysemethoden: Eine anwendungsorientierte Einführung*. 8. Aufl. Springer Berlin Heidelberg, 2000.

[11] *Pythonlibs*. `https://www.lfd.uci.edu/~gohlke/pythonlibs`. Eingesehen am 2.12.2018.

[12] Vera Völkl. *Ein Regressionsmodell zur Untersuchung des Effektes von Datensatzcharakteristiken auf die relative Güte von Prädiktionsalgorithmen mit Anwendung auf 50 Microarray-Studien*. `https://epub.ub.uni-muenchen.de/21733/1/BA_Völkl.pdf`. Eingesehen am 01.02.2018.

B.6. Bagging Trees

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Abteilung Assistenzsysteme und Medizintechnik

Projektgruppe Mobilitäts-Assessments
mit körpernahen Sensoren für Zuhause
SoSe17 - WiSe17/18

Seminararbeit: Bagging Trees

Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autoren: Marlon Willms

14. Februar 2018

Inhaltsverzeichnis

1 Einleitung

Machine Learning (ML) und Künstliche Intelligenz (KI) haben in den letzten Jahren stark an medialer Aufmerksamkeit gewonnen. Dabei sind diese Forschungsfelder nicht neu, bereits 1959 wurde der Begriff des Machine Learnings durch Samuel, A. L. geprägt[7]. Durch die steigende Rechenkapazität sowie den stetig weiterwachsenden Datenmengen, gewinnen ML und KI jedoch weiter an Interesse und Sinnhaftigkeit. So findet sich auch in dieser Seminararbeit eine sinnvolle Verwendung. Im Rahmen der Projektarbeit „Mobilitäts-Assesments mit körpernahen Sensoren für Zuhause“ wird versucht, Mobilitätsveränderungen mit Hilfe verschiedener Sensoren zu erfassen und darzustellen. Um dies zu erreichen, tragen Probanden einen Gürtel, welcher Bewegungsdaten, bestehend aus Accelerometer-, Gyroskop- und Magnetometer-Werten, aufzeichnet. Damit die Veränderungen genauer erfasst werden können, ist es notwendig, aus den Gürtel-Daten Aktivitäten, wie zum Beispiel „Gehen“ oder „Sitzen“, herauszurechnen. Für diese Aufgabe können Machine -Learning-Algorithmen angewandt werden, welche solch eine Klassifikation vornehmen. Da eine Vielzahl unterschiedlicher Algorithmen existieren, ist zu klären welcher sich für diese Aufgabe eignet und wie dieser gegebenenfalls verbessert werden kann.

In dieser Arbeit wird der Fokus auf den „Bagging Trees“-Algorithmus gesetzt, da sich dieser Algorithmus in einer kleinen, eigens durchgeführten Teststudie als einer der besseren erwies. Dargelegt wird beispielsweise die Funktionsweise des Algorithmus. Da dieser jedoch auf einem Baumstruktur-Graphen basiert, werden hierzu vorab die benötigten Grundlagen vermittelt. Auch wird der generelle Ansatz von Bagging kurz dargestellt, um somit auf den „Bagging Trees“-Algorithmus zu schließen. Anschließend werden mit Hilfe eines Test-Datensatzes diverse Parametereinstellungen

stellung und Methoden des Bagging-Trees-Verfahren getestet, um eine Eignung für die Projektgruppe und deren Datensätze zu überprüfen.

2 Grundlagen

Da Entscheidungsbäume die Grundlage von Bagging-Trees und Random Forest bilden, wird folgend der Aufbau und die Funktionsweise erläutert. Auch werden mögliche Probleme der Entscheidungsbäume dargestellt. Ansätze zur Gegensteuerung werden ebenso benannt.

2.1 Entscheidungsbäume

Entscheidungsbäume sind auf Graphen basierende Bäume, welche mit Hilfe von definierten Regeln Entscheidungen treffen können. Ein großer Vorteil hierbei ist die, durch die Darstellungsform gegebene, relativ einfache Interpretierbarkeit [6]. Entscheidungsbäume gehören zum Supervised Machine Learning und besitzen diverse Formen.
Ein einfaches Beispiel ist in Abbildung 2.1 erkennbar. Ziel dieses Entscheidungsbaumes ist es, die „richtige“ Aktivität zu finden.
Einsatzgebiete finden Entscheidungsbäume sowohl im Regressionsbereich als auch bei der Klassifikation, welche aufgrund der Aufgabenstellung dieser Arbeit genauer betrachtet werden. Im Regressionsbereich wird mit Hilfe von Entscheidungsbäumen beispielsweise der geschätzte Wert eines Gebrauchtwagen ermittelt. Dementsprechend handelt es sich um nummerische Werte. Bei der Klassifikation wird wiederum eine Klasse, wie zum Beispiel die Aktivität „Stehen“, ausgegeben.

Beginnend mit dem Wurzelelement wird zuerst die Frage/ Regel gestellt, ob Arbeit zu tun ist oder nicht. Ist Arbeit zu tun, so wird sich dazu entschieden Zuhause zu bleiben (Stay in). Der Entscheidungsbaum beendet in diesem Fall. Ist keine Ar-

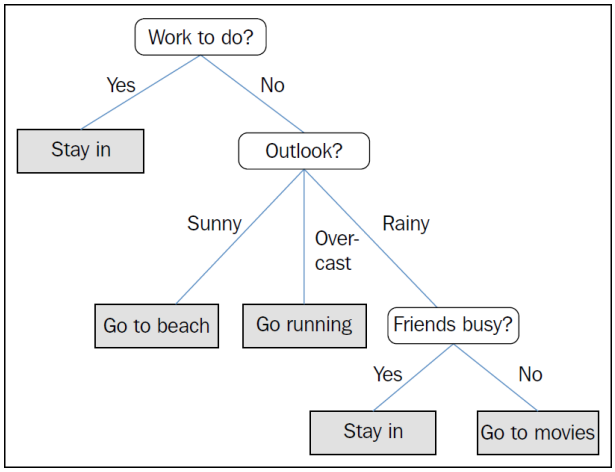


Abbildung 2.1: Beispiel eines Entscheidungsbaums [6]

beit zu tun, so wird als nächstes nach dem Zustand des Wetters gefragt. Hierbei gibt es die Zustände „Sonnig“, „Bewölkt“ oder „Regnerisch“, welche wiederum verschiedene Antworten liefern. Es wird dabei mit möglichst wenig Fragen versucht, herauszufinden, welche Aktivität ausgeführt werden soll. Wie dieser Aufbau entsteht, wird folgend versucht darzustellen. Dazu existieren mehrere Algorithmen. Zu nennen ist beispielsweise ID3, welcher zu c5.0 weiterentwickelt wurde oder aber auch der CaRT-Algorithmus, welcher auch für den „Bagging Trees“-Algorithmus Verwendung findet. Das grundsätzliche Vorgehen ist jedoch bei beiden Algorithmen ähnlich. Zur Erklärung dient das folgende Beispiel. Hierbei ist zunächst der dargestellte Datensatz gegeben (siehe Abbildung 2.2):

Farbe	Durchmesser	Label
Grün	3	Apfel
Gelb	3	Apfel
Rot	1	Apfel
Grün	1	Traube
Grün	3	Apfel

Abbildung 2.2: Beispieldatensatz

Aus dem Datensatz gehen in der ersten Spalte verschiedene Farben hervor, die zweite Spalte beinhaltet den jeweiligen Durchmesser. Die dritte Spalte beschreibt, um welche Frucht es sich bei der jeweiligen Zeile handelt. Diese Trainingsdaten sind somit eine Ansammlung von Beispielen. Die Algorithmen betrachten im ersten Schritt die erste Spalte, also alle Farben und versuchen zu ermitteln, wie viele Farben sie eindeutig einer Frucht zuordnen können. Dies ist bei den Farben Gelb und Rot der Fall (Apfel). Bei den anderen drei Einträgen ist es unklar. Somit ergibt sich eine Quote von 3/2. Als Graph dargestellt ergibt sich 2.3.

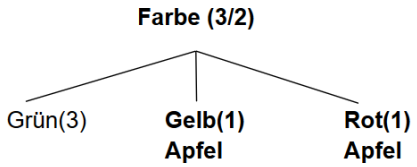


Abbildung 2.3: „Farbe“ als Entscheidungskriterium

Als nächstes wird die zweite Spalte „Durchmesser“ angeschaut. Dort gibt es die Werte „3“ und „1“. Ist der Wert „3“, so kann dieser eindeutig einem Apfel zugeordnet werden, welches drei Mal der Fall ist. Ist der Wert wiederum „1“, so ist es in den beiden Fällen nicht eindeutig welche Frucht daraus hervorgeht. Es ergibt sich daraus eine Quote von 2 zu 3 (siehe Abbildung 2.4).

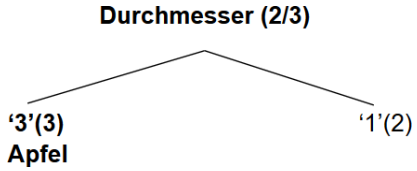


Abbildung 2.4: „Durchmesser“ als Entscheidungskriterium

Da es keine weiteren Spalten zur Betrachtung mehr gibt wird im nachfolgen Schritt nun die jeweilige Quote verglichen. Dabei stellt sich die Quote des Durchmessers als besser dar, da dort mehr Label direkt erkannt werden. Um eine noch bessere Partitionierung vorzunehmen, werden zur Berechnung weitere Faktoren, wie z.B. die Gewichtung hinzugezogen. Bei dem ID3-Algorithmus kommt dafür die Berechnung

des Information Gain zum Einsatz. Beim CaRT-Algorithmus wird der Gini-Index genutzt. Die Formel dafür lautet:

$$Gini(S) = 1 - \sum_{i=1}^{k} p_i^2$$

Abbildung 2.5: Gini-Index zur Berechnung der Quote

Dabei bezeichnet „S“ das Trainingset welches übergeben wird. „k“ steht für die Anzahl der verschiedenen Werte. „pi“ ist die Wahrscheinlichkeit, dass die Klasse „i“ zu „S“ gehört[8]. Zunächst wird der Gini-Index für den gesamten Datensatz ermittelt (Schritt 1), anschließend für die einzelnen Spalten (Schritt 2). Das Ergebnis wird je Spalte wiederum von dem zuerst berechneten Gesamt-Gini-Index abgezogen (Schritt 3). Die Spalte, welche im Vergleich zu den anderen Spalten das höchste Ergebnis erzielt, wird als Split-Kriterium ausgewählt.
Zum besseren Verständnis folgt erneut eine Berechnung anhand des Beispieldatensatzes:

Schritt 1: Berechnung des Gini-Index für das Trainingset (Label).
Gini(S) = 1 - (Häufigkeiten Apfel)^2 + (Häufigkeiten Traube)^2
Gini(S) = 1 - $(4/5)^2$ + $(1/5)^2$
Gini(S) = 1 - 0,64 + 0,04
Gini(S) = 0,32

Schritt 2a: Berechnung des Gini-Index für die Spalte Farbe

Instanzen für Grün:

Gini(Grün) = 1 - (Häufigkeiten Apfel)^2+ (Häufigkeiten Traube)^2

Gini(Grün) = 1 - $(2/3)^2$ + $(1/3)^2$

Gini(Grün) = 1 - 0,44 + 0,11

Gini(Grün) = 0,45

Instanzen für Rot:

Gini(Rot) = 1 - (Häufigkeiten Apfel)^2 + (Häufigkeiten Traube)^2

Gini(Rot) = 1 - $(1/1)^2$ + $(0/1)^2$

Gini(Rot) = 1 - 1 + 0

Gini(Rot) = 0

Instanzen für Gelb:

Gini(Gelb) = 1 - (Häufigkeiten Apfel)^2 + (Häufigkeiten Traube)^2

Gini(Gelb) = 1 - $(1/1)^2$ + $(0/1)^2$

Gini(Gelb) = 1 - 1 + 0

Gini(Gelb) = 0

Berechnung mit Gewichtung:

Häufigkeiten Grün * Gini(Grün) + Häufigkeiten Rot * Gini(Rot) + Häufigkeiten Gelb * Gini(Gelb)

Gini(Farbe) = 3/5 * 0,45 + 1/5 * 0 + 1/5 * 0 = 0,27

Schritt 2b: Berechnung des Gini-Index für die Spalte Durchmesser

Instanzen für „3“:

Gini(„3“) = 1 - (Häufigkeiten Apfel)^2 + (Häufigkeiten Traube)^2

Gini(„3“) = 1 - $(3/3)^2$ + $(0/3)^2$

Gini(„3“) = 1 - 1 + 0

Gini(„3“) = 0

Instanzen für „1“:

Gini(„1“) = 1 - (Häufigkeiten Apfel)^2 + (Häufigkeiten Traube)^2

Gini(„1“) = 1 - $(1/2)^2$ + $(1/2)^2$

Gini(„1“) = 1 - 0,25 + 0,25

Gini(„1“) = 0,5

Berechnung mit Gewichtung:

Häufigkeiten „3“ * Gini(„3“) + Häufigkeiten „1“ * Gini(„1“)

Gini(Durchmesser) = 3/5 * 0 + 2/5 * 0,5 = 0,2

Schritt 3: Vergleich der beiden Spalten (GiniGain)

Gini(S) - Gini(Farbe) = 0,32 - 0,27 = 0,05

Gini(S) - Gini(Durchmesser) = 0,32 - 0,2 = 0,12

Aus der Berechnung ergibt sich, dass durch Splitten mit Hilfe der Farbe eine Verbesserung von 0,05 möglich ist. Beim Durchmesser ist sogar eine Verbesserung der Unreinheit von 0,12 möglich, womit sich der Algorithmus zunächst für das Splitten via Durchmesser entscheiden würde. Anschließend würden alle verbleibenden Daten (Spalte drei und vier) rekursiv nach dem gleichen Prinzip betrachtet werden. Daraus ergibt sich der finale Entscheidungsbaum 2.6.

Ein Problem, welches bei Entscheidungsbäumen auftreten kann, ist das Overfitting. Wie aus der Beschreibung der Funktionsweise ersichtlich, versucht der Entschei-

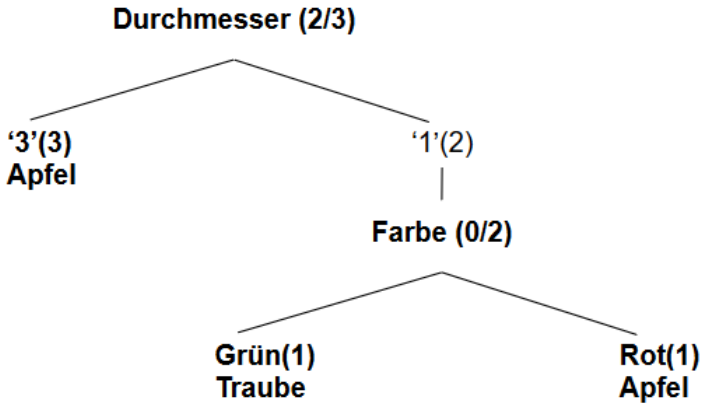


Abbildung 2.6: Finaler Entscheidungsbaum

dungsbaum rekursive solange Regeln beziehungsweise Entscheidungen zu treffen, bis er auch die letzten Restdaten genau gesplittet hat. Dies ist nicht immer Sinnvoll und zudem die Grundlage für Overfitting, was eine zu starke Anpassung an die Trainingsdaten beschreibt. Dies hat zur Folge, dass bei der Klassifizierung von anderen Datensätzen mit Hilfe des Entscheidungsbaumes ein ungenaueres Ergebnis entsteht. Dieses Problem ist in 2.7 abgebildet.

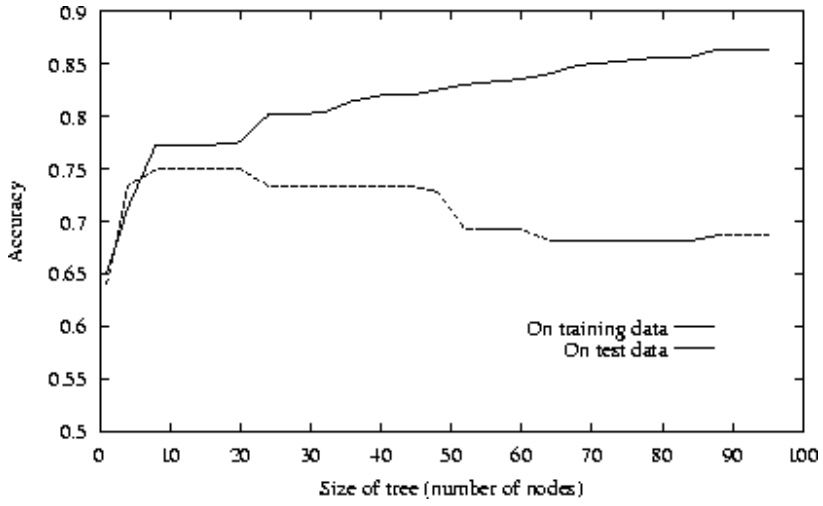


Abbildung 2.7: Overfitting[1]

Zu erkennen ist hierbei, dass mit steigender Anzahl der Knoten eines Baumes zunächst die Erkennungsrate, sowohl beim Klassifizieren von Trainingsdaten als auch von Test-Daten steigt. Ab einer gewissen Größe stagniert die Erkennungsrate auf den Test-Daten jedoch. Bei einem weiter wachsenden Baum fällt die Rate sogar. Um dieser Problematik entgegenzuwirken, kann das Pruning-Verfahren genutzt werden. Dabei wird ein Entscheidungsbaum zunächst komplett erstellt. Anschließend werden jedoch eher unwichtige Knoten wieder herausgenommen, um eine zu starke Anpassung an die Trainingsdaten zu verhindern. Dies hat ebenfalls den Vorteil, dass weniger Speicherplatzbedarf benötigt wird. Um der Problematik noch weiter entgegenzuwirken, ist beispielsweise der Einsatz von „Bagging“ denkbar, welcher im folgenden Kapitel erläutert wird.

2.2 Bagging und Bagging Trees

Der Begriff „Bagging“ ist eine Zusammensetzung der beiden Wörter „**B**ootstrap **agg**regat**ing**“ und wurde durch Leo Breiman geprägt[4]. Es bezeichnet eine Methode in der mehrere verschiedene Klassifikationsmodelle angelernt werden. Diese verschiedenen Modelle bilden anschließend einen zusammengesetzten Klassifikator der im Idealfall eine gesteigerte Vorhersagegenauigkeit besitzt. Werden für das Bagging lediglich Entscheidungsbäume angelernt, so kann dies als „Bagging Trees“ bezeichnet werden. Eingesetzt werden kann dieses Verfahren sowohl bei der Regression, als auch bei der Klassifikation welche in dieser Ausarbeitung genauer betrachtet wird. Es folgt eine Beschreibung des Ablaufes der Methode.

Zu Beginn können die zur Verfügung stehenden Daten in Test- und Trainingsdatensätze aufgeteilt werden. Anschließend werden die Trainingsdaten durch Ziehen von Bootstrap-Stichproben in einzelne, unabhängige Datensätze aufgeteilt. Dabei wird pro anzulernendes Modell eine Stichprobe benötigt. Die Bootstrap-Stichproben werden dabei nach einem definierten Verfahren gezogen. Stehen beispielsweise insge-

samt 30 Trainingssätze zur Verfügung, wird eine definierte Anzahl davon gezogen. Beispielsweise vier Sätze, womit jede Stichprobe immer vier Datensätze enthält. Beim Ziehen der Sätze kann es auch vorkommen, dass ein bereits gezogener Datensatz erneut gezogen wird. Nach dem Erstellen der Stichproben, oder auch „bags“ genannt, wird daraufhin zu jeder einzelnen Stichprobe ein Klassifikationsmodell angelernt. Das gesamte Verfahren wird in 2.8 skizziert.

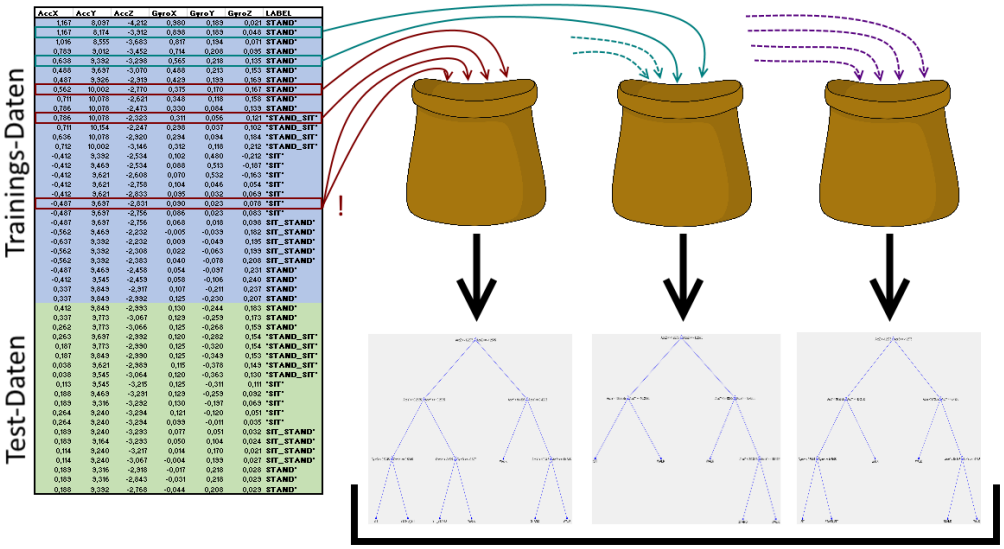


Abbildung 2.8: „bagging“-Methode: Erstellung

Die erlernten Modelle bilden nun im Zusammenschluss das fertige Klassifikationsmodell. Zur Klassifikation wird der zu klassifizierende Datensatz an jedes einzelne Teilmodel gesendet. Diese geben jeweils eine Klasse zurück, welche als „Vote“ bezeichnet werden kann. Per Mehrheitsentscheid wird somit final entschieden um welche Klasse es sich handelt. Skizziert ist dieser Vorgang in 2.9.

Wie viel Erfolg diese Methode einbringt hat Leo Breiman in seinem Paper dargestellt[4]. Hierbei verglich er die Klassifikationsergebnisse zwischen einem Entscheidungsbaum und einem Bagging-Modell, bestehend aus 50 Entscheidungsbäumen. Zur Validation wurde eine 10-fold-cross-validation angewandt. Durchgeführt wur-

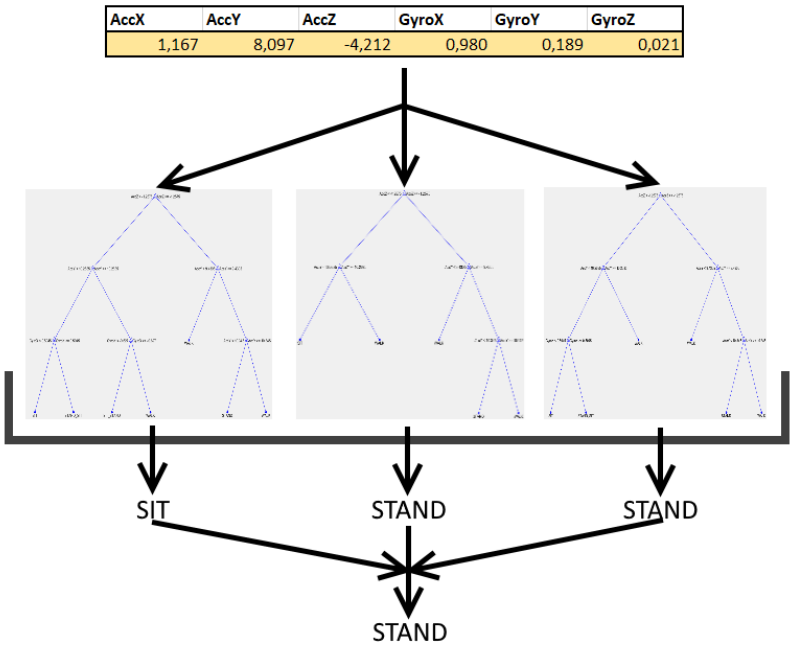


Abbildung 2.9: „bagging“-Methode: Klassifizierung

de dieser Vergleich auf verschiedenen Datensätzen mit unterschiedlicher Anzahl an Samples, Variablen und Klassen, wie in Abbildung 2.10 dargestellt.

Data Set	# Samples	# Variables	# Classes
waveform	300	21	3
heart	1395	16	2
breast cancer	699	9	2
ionosphere	351	34	2
diabetes	768	8	2
glass	214	9	6
soybean	683	35	19

Abbildung 2.10: Datensätze zum Vergleich [4]

Die erzielten Ergebnisse sind in Abbildung 2.11 aufgelistet. Es zeigt sich dabei eine sehr unterschiedliche Verbesserung. Diese reicht von sechs Prozent bei den Diabetes-Daten bis hin zu 43 Prozent bei den Herz-Datensätzen. Somit fand bei jedem Datensatz eine Verringerung der Falschklassifizierung statt.

Data Set	$\bar{e}_S$	$\bar{e}_B$	Decrease
waveform	29.1	19.3	34%
heart	4.9	2.8	43%
breast cancer	5.9	3.7	37%
ionosphere	11.2	7.9	29%
diabetes	25.3	23.9	6%
glass	30.4	23.6	22%
soybean	8.6	6.8	21%

Abbildung 2.11: Klassifikationsergebnisse im Vergleich[4]

Zu beachten ist, dass bei dieser Methode nicht nur Entscheidungsbäume angelernt werden können, sondern auch andere Modelle. Darunter fällt zum Beispiel ein neuronales Netzwerk[4]. Jedoch eignen sich nicht alle Modelle hierfür. Beispielsweise erhält ein „k-nearest neighbor“-Klassifikator durch diese Methode kein besseres Ergebnis. Dies ist davon abhängig, ob das Modell als „stabil“ oder „instabil“ gilt. Ist es „stabil“, so erhält man hierbei keine Verbesserung. Die Begriffe bezeichnen, ob durch eine Veränderung der Trainingsdaten durch Ziehen der Bootstrap-Stichprobe jeweils ein anderes Ergebnis antrainiert wird. Ist dies nicht der Fall, so hat diese Methode keinen Effekt. Ein Vorteil des „Baggings“ besteht darin, dass das Ziehen der Bootstrap-Stichproben sowie insbesondere das Anlernen der Teilmodelle verteilt berechnet werden kann. Steht die dafür notwendige Infrastruktur zur Verfügung, so kann dadurch die Anlernzeit stark verringert werden.

Um zu klären, wie viele einzelne Teilmodelle für eine Verbesserung der Klassifikation notwendig sind, also wie viele Stichproben gezogen und Modelle antrainiert werden müssen, führte Leo Breiman eine sonst identische Berechnung mit Hilfe von 10, 25, 50 und 100 Teilmodellen durch (Siehe Abbildung 2.12). Zu sehen ist, dass es keine nennenswerte Verbesserung ab ca. 50 Teilmodellen mehr gibt. Unterschiede zwischen zehn und 25 Teilmodellen sind jedoch durchaus erkennbar. Eine ähnliche Untersuchung wurde auch im Rahmen der Projektgruppe durchgeführt und ist unter Kapitel 3 einsehbar.

No. Bootstrap Replicates	Misclassification Rate
10	21.8
25	19.4
50	19.3
100	19.3

Abbildung 2.12: Vergleich unterschiedlicher Anzahl an Teilmodellen[4]

Die Bagging-Methode bietet zur Bestimmung der Vorhersagegenauigkeit, beziehungsweise der Falschklassifizierung, eine eigene Möglichkeit. Dabei wird beim Ziehen der Bootstrap-Stichproben bei jedem Bag darauf geachtet, welche Elemente nicht in dem Bag landen. Diese können nach dem Anlernen der Teilmodelle zur Validierung genutzt werden. Diese Vorgehensweise wird „out-of-bag“ oder kurz „oob“ genannt und erzielt im Direktvergleich mit anderen Validierungsmethoden wie zum Beispiel der Cross-Validation gute Ergebnisse [2].
Eine Variante von „Bagging Trees“ sind die „Random-Forest“. Diese werden im folgendem Kapitel weiter dargestellt.

2.3 Random Forest

Eine Variante der „Bagging Trees“ ist die „Random-Forest“-Methode. Sie wurde ebenfalls von Leo Breiman vorgestellt und baut dabei auf den Bagging-Trees-Algorithmus auf, spezialisiert diesen jedoch durch ein paar weitere Regeln und Methoden [5].
Die erste Regel besagt dabei, dass die Entscheidungsbäume nicht per Pruning-Verfahren gestutzt werden. Zweitens wird der Einsatz von „Random Subspacing“ genutzt. Bei der „Random Subspacing“-Methode wird für jeden Entscheidungsbaum zunächst, wie beim Bagging-Trees-Algorithmus eine Bootstrap-Stichprobe gezogen. Jedoch wird beim Antrainieren, genauer beim Berechnen des besten Split-Parameters, nur eine per Zufall ausgewählte Menge an Attributen, bzw. Merkmalen

zur Verfügung gestellt. Die Anzahl der Attribute kann vorab festgelegt werden. So werden beispielsweise von fünf möglichen Attributen nur zwei Attribute für die Berechnung des Splits zur Verfügung gestellt. Nach dem Berechnen des Splits wird für den nächsten Knoten bzw. Split erneut per Zufall eine Auswahl an Attributen zusammengestellt. Dies verstärkt den Effekt, unterschiedliche Teilmodelle anhand der gleichen Trainingsdaten zu generieren. Zu beachten ist dabei, dass durch die Vorgabe des Prunings-"Verbots" gegebenenfalls großer Speicherbedarf für die Modelle benötigt wird. Dies spiegelte sich auch in den eigenen Tests unter Kapitel 3 wider.

3 Nutzen in der Projektgruppe

Ziel dieser Seminararbeit ist es unter anderem zu prüfen, in wie weit sich die vorgestellten Methoden für die Klassifikationen der Projektgruppen-Datensätze eignen. Um diese Frage besser beantworten zu können, wird im Folgendem versucht eine Klassifizierung auf ähnlichen Daten auszuführen, die Ergebnisse zu betrachten und zudem mit der Veränderung eines Parameters dessen Auswirkungen zu betrachten. Die Projektgruppe baut auf der Arbeit der vorherigen Projektgruppe auf, welche „klinische“-Tests an Senioren durchgeführt haben, um verschiedene Bewegungsdaten zu erhalten. Dies wurde mit dem selben Sensorgürtel aufgenommen, der auch in der jetzigen Projektgruppe Einsatz findet. Die daraus gewonnen Daten gleichen sich somit in gewisser Weise und eignen sich somit als Daten-Grundlage für die folgenden Tests. Der Datensatz umfasst circa 4,9 Millionen Zeilen mit jeweils sechs Merkmalen. Diese Merkmale bestehen aus Erfassung der Beschleunigung, jeweils auf der x, y und z-Achse, sowie die gyroskopischen Daten auf den selben Achsen. Die siebte Spalte enthält zudem die Labelbezeichnungen (siehe Abbildung 3.1).

Name	Type	Range
AccX	double	-39.479 .. 28.223
AccY	double	-21.243 .. 92.32
AccZ	double	-45.273 .. 70.442
GyroX	double	-29.539 .. 12.066
GyroY	double	-5.968 .. 5.881
GyroZ	double	-2.942 .. 3.376
Label	cell	18 unique

Abbildung 3.1: Aufbau des Datensatzes

Insgesamt existieren 18 verschiedene Aktivitäten in den Labels. Diese lassen sich unterteilen in dynamische Aktivitäten, statische Aktivitäten und Transitionen, wel-

che eine Art Zwischenstufe zwischen zwei Aktivitäten darstellen (siehe Abbildung 3.2).

Label	Art
BEND_OVER_OR_SQUAT	Dynamisch
JUMP	Dynamisch
LIE_ON_BACK	Statisch
LIE_ON_BACK_LIE_ON_SIDE	Transition
LIE_ON_SIDE	Statisch
LIE_ON_SIDE_LIE_ON_BACK	Transition
LIE_SIT	Transition
SIT	Statisch
SIT_LIE	Transition
SIT_STAND	Transition
STAIR_CLIMB	Dynamisch
STAND	Statisch
STAND_LIE	Transition
STAND_SIT	Transition
TURN_AROUND_LEFT	Dynamisch
TURN_AROUND_RIGHT	Dynamisch
WALK	Dynamisch
WALK_BACKWARDS	Dynamisch

Abbildung 3.2: Label des Datensatzes

Aufbauend auf diesen Datensatz wurde mehrfach ein Bagging-Trees-Algorithmus antrainiert. Dies wurde in dem Programm „Matlab“ in der Version 2016R durchgeführt. Um vergleichbare Klassifizierungsergebnisse zu erhalten, wurde auf die „out-of-bag“-Validierung verzichtet und stattdessen eine 5-Folds-Validierung auf allen Tests durchgeführt. Diese ermöglicht das direktere Vergleichen; auch von Modellen, die nicht auf dem Bagging-Verfahren aufbauen wie z.B. KNN. Um den Effekt des Baggings zu bestimmen, wurde auch ein einzelner Entscheidungsbaum antrainiert. Dieser wurde auf 100 Splits gestutzt. Er erhielt eine Vorhersagegenauigkeit von 89,2%. Für das Bagging wurde zunächst das Training mit 30 Bags durchgeführt, da Matlab dies als Default-Wert vorschlägt. Das Ergebnis erreicht eine Erkennungsrate von 96,33%. Im direkten Vergleich erreicht das Bagging somit eine beachtliche Steigerung von circa sieben Prozent. Auffällig hierbei war, dass nicht nur deutlich mehr Rechenkapazität benötigt wurde, sondern auch viel mehr Speicherplatz. Beim

Entscheidungsbaum mit 100 Splits liegt dieser im Kilobyte-Bereich. Beim Bagging-Modell hingegen wurden 19 Gigabyte benötigt. Dies ist damit zu erklären, dass zum Einem nicht einer, sondern 30 Entscheidungsbäume enthalten sind, zum Anderen wurde beim Antrainieren kein Pruning genutzt. Somit wachsen die Bäume komplett aus und benötigen dadurch sehr viel Speicherplatz. Ein Versuch, einen der Bagging-Entscheidungsbäume in Matlab zu Visualisieren ist in Abbildung 3.3 (unterer Part der Bildes), im Vergleich zum 100-Splits-Entscheidungsbaum (oberer Part) dargestellt.

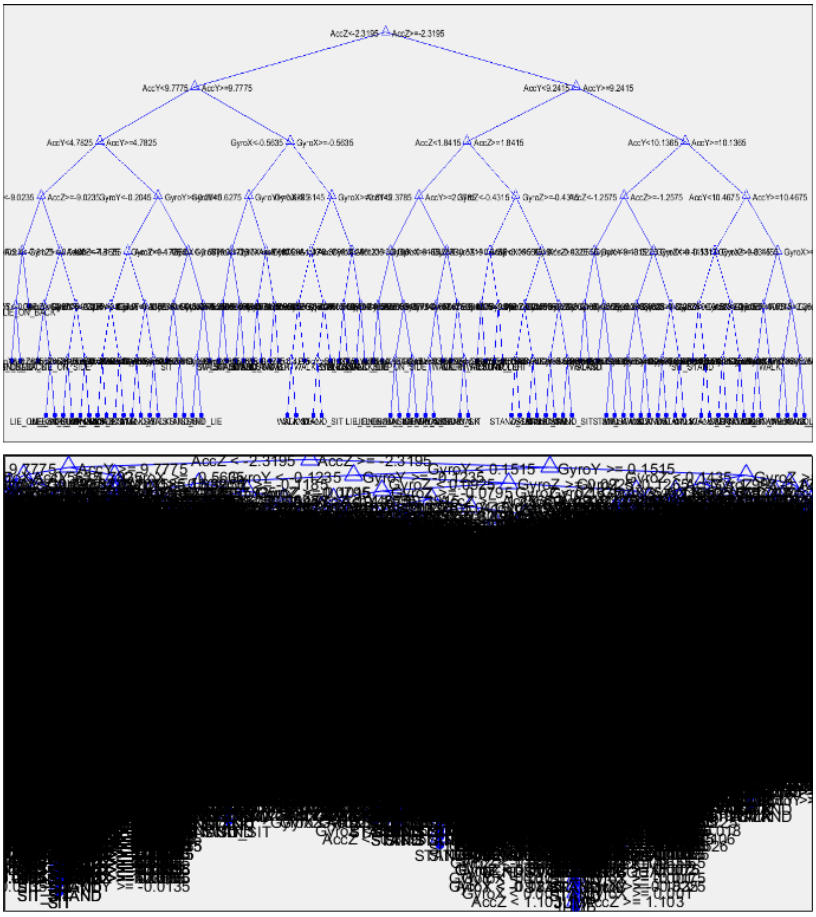
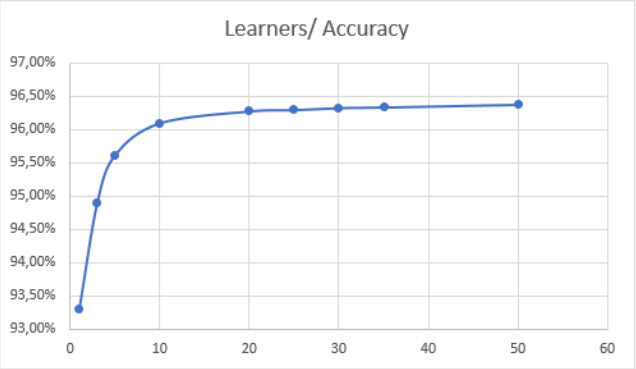


Abbildung 3.3: Darstellungsversuch eines Bagging-Entscheidungsbaumes (unten) und einem auf 100-Splits gestutzten Baumes (oben)

Nach dem Antrainieren des ersten Bagging-Modelles wurde mit der Anzahl der Bags experimentiert. Dieser Parameter wurde gewählt, da er als einer der relevantesten gilt [3]
Das Ergebnis ist in Abbildung 3.4 dargestellt.



Accuracy	Bags	GB
93,30%	1	0,8
94,90%	3	2
95,62%	5	3
96,10%	10	9
96,28%	20	13
96,30%	25	16
96,33%	30	19
96,34%	35	23
96,38%	50	33

Abbildung 3.4: Ergebnisse des Trainings

Auf der rechten Seite sind die Ergebnisse in tabellarischer Form inklusive Speicherplatzbedarf in Gigabyte dargestellt. Auf der linken Seite ist in Form eines Liniendiagramms die Abhängigkeit zwischen der Anzahl der genutzten Bags/ Learners und der Genauigkeit zu finden. Im Bezug auf die Genauigkeit ist der größte Anstieg zwischen einem und zehn Bags zu finden. Ab 20 Bags steigt die Genauigkeit nur marginal weiter an. Diese Ergebnisse decken sich in etwa mit denen von Leo Breiman (siehe Abbildung 2.12). Aus den Ergebnissen geht hervor, dass in diesem Fall nicht mehr als 20 bis 30 Bags benötigt werden, um eine hohe Genauigkeit zu erhalten. Aus der Tabelle wird auch der hohe Speicherplatzbedarf ersichtlich. So werden bei 50 Bags 33 Gigabyte Speicher benötigt. Je nach Einsatzgebiet kann dieser Bedarf zu Problemen führen. Beispielweise beim Einsatz auf mobilen Endgeräten, mit begrenztem Speicherplatz. Eine Lösung hierfür könnte wiederum sein,

die Klassifikation auf einem ausgelagertem Server zu übernehmen und das Mobilgerät lediglich als Datenstream-Sender zu nutzen. Eine weitere Alternative besteht darin, durch Pruning der einzelnen Entscheidungsbäume die Größe zu minimieren. Um dieser Möglichkeit nachzugehen, wurden ebenfalls Tests durchgeführt. Hierfür wurden die Entscheidungsbäume jeweils auf 100 Splits gestutzt. Dieser Test wurde für eine Varianz von Bags ausgeführt. Das Ergebnis ist in Abbildung 3.5 tabellarisch dargestellt.

Accuracy	**Bags**	MB
88,49%	**1**	202
89,30%	**3**	206
89,79%	**5**	209
89,74%	**10**	217

Abbildung 3.5: Ergebnisse des Trainings beim Einsatz von Pruning

Zu erkennen ist eine deutliche Verschlechterung der Genauigkeit. Jedoch wird selbst bei der höchsten Anzahl an Bags sichtlich weniger Speicherplatz benötigt, gegenüber dem Bagging-Model mit nur einem Bag. Dadurch eignet sich dieses Verfahren mehr für Endgeräte mit beschränktem Speicherplatz. Wegen mangelnder Genauigkeit ist dann jedoch ein anderer Algorithmus besser geeignet, wie z.B. KNN welcher eventuell weniger Speicherplatz benötigt und dennoch ein etwas besseres Ergebnis auf diesem Datensatz erzielt.

Um zu analysieren welche Unterschiede bei unterschiedlicher Anzahl an Bags genauer existieren, ist ein Vergleich der jeweiligen Konfusionsmatrizen sinnvoll. Betrachtet wird folgend die Konfusionsmatrix die aus dem Antrainieren der drei Bags ohne Pruning mit einer Genauigkeit von 94,90% hervorging (siehe Abbildung 3.6) und die Konfusionsmatrix aus dem Antrainieren mit zehn Bags ebenfalls ohne Pruning mit einer Genauigkeit von 96,10% (siehe Abbildung 3.7).

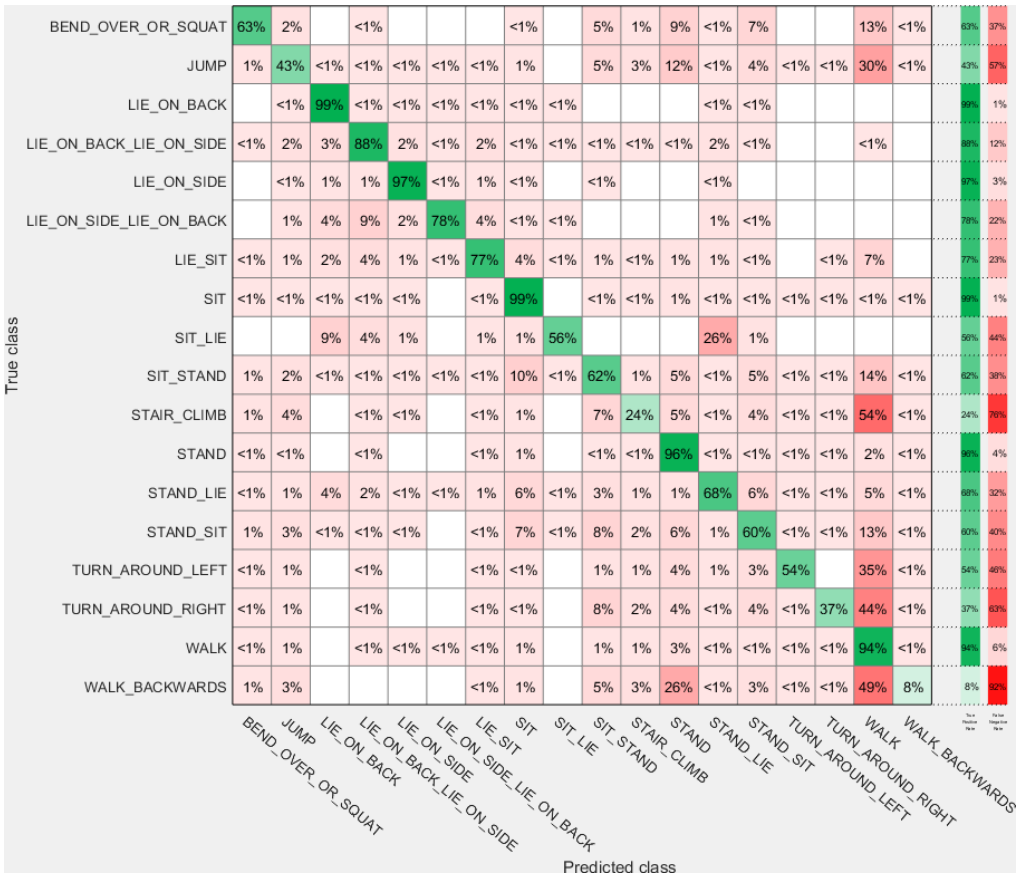


Abbildung 3.6: Konfusionsmatix als Ergebnis der drei Bags ohne Pruning

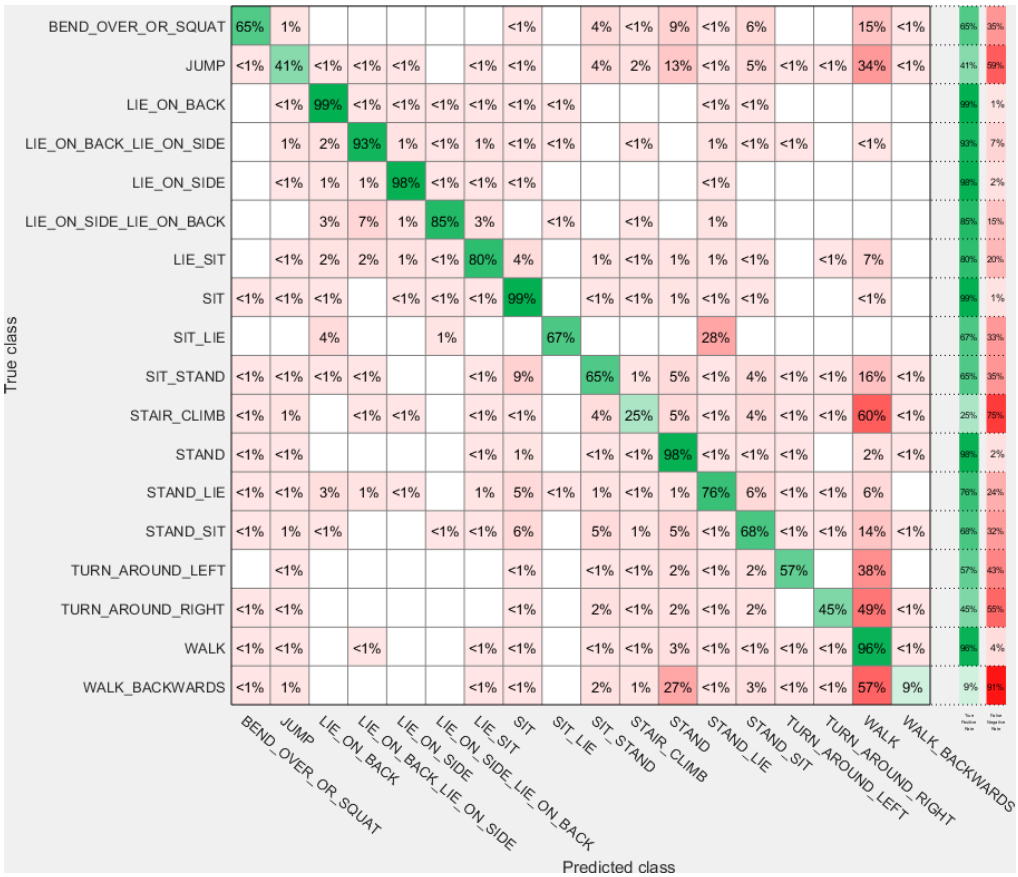


Abbildung 3.7: Konfusionsmatix als Ergebnis der zehn Bags ohne Pruning

Die Matrizen besitzen auf der linken Seite die originalen Aktivitäten, welche aus dem Datensatz hervorgehen und zu erkennen sind. Auf der unteren Seite befinden sich wiederum jene Aktivitäten, welche durch den Klassifikator „erkannt“ wurden. Dabei sind ebenfalls falsch erkannte Aktivitäten dabei.
Generell ist ersichtlich, dass es im Vergleich zwischen den drei und den zehn Bags, mit einer Ausnahme, bei allen Aktivitäten zu einer verbesserten Erkennungsrate kam. Auch existieren mehr weiße Felder was wiederum bedeutet, dass einige falsch zugeordneten Aktivitäten komplett verschwunden sind. Die einzige Ausnahme hinsichtlich der Verbesserung ist das „JUMP“. Diese Aktivitätenerkennung fällt von 43% auf 41%. Dies ist der häufigeren Falschklassifizierung als „STAND“ und „WALK“ geschuldet . Die anderen Falschklassifizierungen nehmen sonst auch bei „JUMP“ ab. Die größten Verbesserungen mit sieben bis neun Prozent gehen bei den Aktivitäten „LIE_ON_SIDE_LIE_ON_BACK„ , „SIT_LIE“, „STAND_LIE“, „STAND_SIT“ und „TURN_AROUND_RIGHT“ hervor. Auffällig ist dabei insbesondere das es sich dabei mit Ausnahme der zuletzt aufgezählten Aktivität nur um Transitionen handelt. Diese scheinen durch steigende Anzahl an Bags somit besonders an Genauigkeit gewinnen zu können.

4 Zusammenfassung

In dieser Seminararbeit wurde versucht, die Funktionsweise des Bagging-Trees-Algorithmus aufzuzeigen und eine Eignungsprüfung für die Projektgruppe durchzuführen. Zu diesem Zweck wurden zunächst die Entscheidungsbäume, welche die Grundlage des Algorithmus bilden, thematisiert und anhand von Beispielrechnungen vertieft. Bei den Entscheidungsbäumen zeigten sich jedoch Probleme, wie z.B. das Overfitting, auf. Die Vorstellung der Bagging- und Bagging-Trees-Methoden zeigten Ansätze zur Lösung des Problems, wodurch eine erhöhte Genauigkeit bei der Erkennung von Klassen ermöglicht wurde. In der eigenen kleinen Studie, in der verschiedene Parametereinstellungen und Vorgehensweisen getestet wurden, wurde deutlich, dass dieser Algorithmus mit den Gürtel-Sensor-Daten der vorherigen Projektgruppe gute Ergebnisse erzielt, wodurch ein Einsatz für die derzeitige Projektgruppe denkbar ist. Erkennbar war weiterhin, dass durch eine Erhöhung der Bag-Anzahl insbesondere die Erkennungsrate bei den Transitionen zunahm. Probleme zeigten sich jedoch zum Einen in dem hohen Rechenleistungsbedarf zum Antrainieren des Models, insbesondere aber auch bei dem hohen Speicherplatzbedarf des Klassifikators. Dies könnte insbesondere auf mobilen Endgeräten mit geringen Speicherplatz zu Problemen führen.

Literatur

[1] Diane Cook. *Overfitting in Decision Tree Learning*. URL: http://www.eecs.wsu.edu/~cook/dm/lectures/l4/node17.html.

[2] Gareth James u. a. *An introduction to statistical learning: With applications in R*. Corrected at 8th printing. Springer texts in statistics. New York u. a.: Springer, 2017. ISBN: 978-1-4614-7138-7.

[3] Jason Brownlee. *Bagging and Random Forest Ensemble Algorithms for Machine Learning*. 22.04.2016. URL: https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/.

[4] Leo Breiman. *Bagging Predictors*. 1996. URL: https://link.springer.com/article/10.1023/A:1018054314350.

[5] Leo Breiman. *RANDOM FORESTS*. 2001. URL: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf.

[6] Sebastian Raschka. *Python machine learning: Unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics*. Community experience distilled. Birmingham und Mumbai: Packt Publishing open source, 2016. ISBN: 978-1-78355-513-0.

[7] A. L. Samuel. „Some Studies in Machine Learning Using the Game of Checkers“. In: *IBM Journal of Research and Development* 3.3 (1959), S. 210–229. ISSN: 0018-8646. DOI: 10.1147/rd.33.0210.

[8] Yin Zhao. *An example of calculating gini gain in CART*.

B.7. Algorithmische Lösungen zur Unterstützung der Labelnachbearbeitung

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Abteilung Assistenzsysteme und Medizintechnik

Projektgruppe Mobilitäts-Assessments
mit körpernahen Sensoren für zuhause
SoSe17 - WiSe17/18

Seminararbeit: Algorithmische Lösungen zur Unterstützung der Labelnachbearbeitung

Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autoren: Maren Steinkamp

2. Februar 2018

Zusammenfassung

In dieser Seminararbeit wird das Problem betrachtet, dass der Prozess des manuellen Labelns viel Zeit und Aufwand bedeutet. Um diese Zeit und diesen Aufwand zu verringern, werden drei verschiedene algorithmische Lösungen betrachtet: Der Change Detection Algorithmus, das semiüberwachte Labeling und das Transfer Labeling. Der Change Detection Algorithmus bestimmt Punkte in den Daten, an denen eine Aktivität zu einer anderen Aktivität wechselt. Das semiüberwachte Labeling klassifiziert die Daten mit einer geringen Menge an gelabelten Daten als Wissenbasis selbstständig vor. Die labelnde Person muss nur bei einer Falschentscheidung des Algorithmus eingreifen. Durch jedes Eingreifen wird die Wissensbasis vergrößert. Das Transfer Labeling greift an dem Punkt ein, wenn das System zum Daten aufnehemen geändert werden soll. Es überträgt die Label von einem System auf das andere und verhindert somit den Labelprozess für das neue System. Alle drei Algorithmenarten eignen sich auf ihre Art dazu, den Labelprozess zu verkürzen.

Inhaltsverzeichnis

1 Einleitung

Im Rahmen unserer Projektgruppe Mobilitäts-Assessments mit körpernahen Sensoren für Zuhause benutzen wir Algorithmen aus dem Bereich des maschinellen Lernens, um die Daten des Sensorgürtels der zugehörigen Aktivität zuzuordnen. Diese Algorithmen benötigen aussagekräftige Beispieldaten, in denen mit Labeln die zugehörige Aktivität ausgedrückt wird. Um diese Beispieldaten zu generieren haben wir Probanden zu Hause besucht und mit Hilfe einer Android App die Aktivitäten, die diese im Verlaufe unseres Besuch absolvierten, zeitgenau dokumentiert. Bei diesem Verfahren des Labelns treten jedoch leicht Fehler auf. Zum einen sind die Uhrzeiten der benutzten Tablets nicht auf die Millisekunde genau mit den Uhrzeiten der Sensorgürtel synchronisiert. Eine weitere Fehlerquelle ist die Reaktionszeit, die eine Person braucht, um nach dem Feststellen einer Aktivität des Probanden auf dem Tablet das Richtige anzugeben. Wir können vorher nicht wissen, wann und welche Aktivität der Proband als nächstes ausführt. Also können wir nicht gleichzeitig mit dem Start der Aktivität auch den Start im Tablet vermerken. Hinzu kommt, dass bei schnell aufeinander folgenden Aktivitäten durch die kurze Zeit, die zur Reaktion bleibt, leicht falsche Eintragungen entstehen können.
Durch diese Fehlerquellen sind die vom Tablet aufgenommenen Daten sehr ungenau und können lediglich als Referenz im Labelprozess genutzt werden. Dies führt dazu, dass der Prozess des Labelns einen großen Zeitaufwand verursacht.
In dieser Seminararbeit möchte ich algorithmische Lösungen vorstellen, die diesen Prozess beschleunigen. Dazu stelle ich drei Algorithmenarten vor. Jede dieser Algorithmenarten werde ich auf ihre Tauglichkeit für den Einsatz in unserer Projektgruppe hin untersuchen. Außerdem werde ich eine dieser Algorihtmenarten selbst austesten und die Ergebnisse dokumentieren.

2 Algorithmen

Im folgenden Kapitel werde ich drei verschiedene Arten von Algorithmen vorstellen, die sich meiner Meinung nach zur Unterstützung der Labelnachbereitung eignen. Als erstes werde ich zwei Varianten des Change Detection Algorithmus vorstellen: Die Class Separability und die Cumulative Sum. Es gibt noch weitere Varianten dieses Algorithmus. Da diese beiden Varianten allerdings in der Literatur bereits für die Aktivitätserkennung eingesetzt wurden, habe ich mich auf diese beschränkt. Des Weiteren erkläre ich wie semiüberwachtes Labeling und Transfer Labeling zur Labelnachbearbeitung beitragen können.

2.1 Change Detection

Im Umgang mit zeitabhängigen Daten ist eine automatisierte Analyse aufgrund großer Datenmengen nicht mehr wegzudenken. Dabei ist es in vielen Fällen besonders wichtig den Zeitpunkt festzustellen, an dem sich die Daten signifikant ändern. Meist sind die Daten vor und nach diesem Wechsel konstant. Dies ist die Basis für die Berechnungen, welche die Change Detection Algorithmen einsetzen. Change Detection Algorithmen finden ihren Einsatz in verschiedenen Bereichen.
Sie werden zum Beispiel eingesetzt, um abnormale Messwerte in Sensordaten zu ermitteln. Wenn beispielsweise in einem Wald Temperatur und Luftfeuchtigkeit gemessen werden, können abnormale Messergebnisse auf Naturkatastrophen wie Waldbrände hinweisen. Die Change Detection Algorithmen sollen solche Ereignisse möglichst schnell und mit möglichst wenigen Fehlalarmen erkennen[3].
Ein weiteres Beispiel ist die Segmentierung von Signalen. In diesen Fällen ist der

Change Detection Algorithmus meist der erste Schritt für die Zuordnung der Daten in Kategorien. Der Algorithmus teilt die Daten in Segmente, bei denen an den Anfangs- und Endpunkten signifikante Änderungen auftreten. Dies wird zum Beispiel in der Spracherkennung eingesetzt. Durch die Segmentierung sind einzelne Worte voneinander getrennt, sodass die Erkennung des jeweiligen Wortes leichter fällt. Auch in der Aktivitätserkennung ist es vorstellbar, dass eine Segmentierung als erster Schritt der Datenverarbeitung nützlich sein könnte[1].

2.1.1 Class Separability

Dieser Algorithmus setzt die Change Detection mit Hilfe der Class Separability um. Er kann für abhängige und unabhängige Daten verwendet werden.
Für die Berechnung von Wechselpunkten mit der Class Separability wird das Wissen über die Form und die statistischen Parameter der Distributionen vor und nach einem möglichen Wechselpunkt nicht benötigt. Für die Erkennung eines Wechselpunktes ist es wichtig, dass Unterschiede in statistischen Parametern erkennbar sind. Es ist dagegen unwichtig, vorher zu wissen, welche das sind.
Der Algorithmus sucht innerhalb eines Abschnitts im Datensatz den Punkt, an dem ein Wechselpunkt am wahrscheinlichsten ist. Dieser Schritt wird Hypothesenbildung genannt. Im weiteren Verlauf des Algorithmus wird überprüft, ob dieser hypothetische Wechselpunkt ein wirklicher Wechselpunkt ist. Das nennt sich Hypothesenvalidierung.
Im Laufe des Algorithmus wird ein Analysefenster über den Datensatz geschoben. Für jeden Fensterabschnitt wird geprüft, ob innerhalb dieses Bereichs ein Wechselpunkt vorliegt. Damit gerade in Randbereichen dieses Fensters genügend große Distributionen vor beziehungseise hinter einem möglichen Wechselpunkt zustande kommen, wird das Fenster an beiden Seiten um einen Bereich der Größe m vergrößert. Um den wahrscheinlichsten Wechselpunkt in jedem Fenster zu finden, werden

für jeden Datenpunkt Varianz (s_f^2) und Mittelwerte ($\bar{f}$) der Distributionen vor und nach diesem Punkt berechnet. Die Berechnungen sind in Formel 2.1 zu sehen. n_1 ist die Anzahl der Datenpunkte in der Distribution vor einem möglichen Wechselpunkt und n_2 ist die Anzahl der Datenpunkte in der Distribution danach.

$$\begin{aligned} \bar{f}_1(l) &= \frac{\sum_{i=1-m}^{l-1} f(X_i)}{n_1}, \qquad & s^2_{f_1}(l) &= \frac{\sum_{i=1-m}^{l-1} (f(X_i) - \bar{f}_1)^2}{n_1 - 1} \\ \bar{f}_2(l) &= \frac{\sum_{i=l}^{n+m} f(X_i)}{n_2}, \qquad & s^2_{f_2}(l) &= \frac{\sum_{i=l}^{n+m} (f(X_i) - \bar{f}_2)^2}{n_2 - 1} \end{aligned} \tag{2.1}$$

Wenn ein Wechselpunkt vorliegt, sollten sich Varianz und Mittelwert der beiden Distributionen signifikant unterscheiden. Um diese Unterschiede besser erkennen zu können, werden zwei statistische Eigenschaften definiert: Die Between-Class Scatter (s_b) und die Within-Class Scatter (s_w).

$$\begin{aligned} s_b(l) &= \frac{n_1 n_2}{(n_1 + n_2)} (\bar{f}_1(l) - \bar{f}_2(l))^2 \\ s_w(l) &= (n_1 - 1) s^2_{f_1}(l) + (n_2 - 1) s^2_{f_2}(l) \end{aligned} \tag{2.2}$$

Die Between-Class Scatter beschreibt den Grad der Verschiedenheit zwischen den beiden Distributionen und die Within-Class Scatter den Grad der Homogenität. Ein Wechselpunkt liegt demnach vor, wenn die Between-Class Scatter maximal und die Within-Class Scatter minimal ist.

Da der Algorithmus auch online einsetzbar sein soll, also auf Daten, die kontinuierlich gestreamt werden, ist es wichtig, die Komplexität der Berechnungen möglichst gering zu halten. Dafür müssen die oben gezeigten Berechnungen lediglich für den ersten Datenpunkt eines Fensters ausgeführt werden. Alle weiteren Berechnungen bauen rekursiv auf diesen Werten auf. Die Mittelwerte der Distributionen können sehr einfach rekursiv berechnet werden:

$$\begin{aligned} \bar{f}_1(l+1) &= \frac{m + l - 1}{m + l} f_1(l) + \frac{f(X_l)}{m + l} \\ \bar{f}_2(l+1) &= \frac{n + m - l + 1}{n + m - l} f_2(l) + \frac{f(X_l)}{n + m - l} \end{aligned} \tag{2.3}$$

Darauf aufbauend kann auch die Between-Class Scatter einfach rekursiv berechnet werden. Die Within-Class Scatter ist schwerer rekursiv berechenbar. In [3] wurde jedoch bewiesen, dass die Summe der Between-Class Scatter und der Within-Class Scatter konstant ist. Also hat es den gleichen Effekt, wenn nur die Between-Class Scatter maximiert wird. Der wahrscheinlichste Wechselpunkt befindet sich an dem Index, an dem die Between-Class Scatter innerhalb des Fenseters maximal ist.
Im nächsten Schritt wird dieser Punkt (e) validiert. Wenn e ein Wechselpunkt ist, weichen die Mittelwerte der Distributionen vor und nach e signifikant voneinander ab. Ob eine signifikante Abweichung vorliegt, wird mit einem statistischen Standardtest, dem t-Test, ermittelt. Die Differenz der Mittelwerte wird mit dem Standardfehler überprüft. Für die Berechnung des Standardfehlers wird die Varianz und der Mittelwert der beiden Distributionen benötigt.

$$\begin{aligned} t &= \frac{f_1(e) - f_2(e)}{s} & s &= \sqrt{\frac{2MSE}{n_h}} \\ MSE &= \frac{n_1 s_{f_1}^2(e) + n_2 s_{f_2}^2(e)}{n + 2m - 2} & n_h &= \frac{2}{\frac{1}{n_1} + \frac{1}{n_2}} \end{aligned} \tag{2.4}$$

Der errechnete Wert t wird mit Hilfe des t-Tests untersucht. Wenn t den Anforderungen des t-Tests entspricht, wird der Punkt e als Wechselpunkt akzeptiert[3].

2.1.2 CuSum

Als Grundlage des Cumulative Sum Algorithmus (CuSum) wird das log-likelihood Verhältnis des Datensatzes verwendet. Dieses ist wie folgt definiert:

$$s_i = ln \frac{p_{\theta_1}(y_i)}{p_{\theta_0}(yi)} \tag{2.5}$$

Dabei beschreibt p_{θ_1}, den Teil des Datensatzes nach einem möglichen Wechselpunkt und p_{θ_0} den Teil des Datensatzes vor einem möglichen Wechselpunkt. Ein Wechsel innerhalb eines Datensatzes zeigt sich durch einen Vorzeichenwechsel im log-likelihood Verhältnis.

CuSum berechnet für jeden Punkt im Datensatz S_k, die Summe aller log-likelihood Verhältnisse von Punkten innerhalb eines festgelegten Radius um den betrachteten Punkt. Um eine Entscheidung zu treffen wird aus allen berechneten Werten S_k das Minimum m_k gespeichert und von dem Wert S_k abgezogen. Ist die Differenz größer oder gleich eines empirisch ermittelten Grenzwertes, wird der Punkt k als Wechselpunkt akzeptiert. In Formeln ausgedrückt, wird beim CuSum also folgendes berechnet:

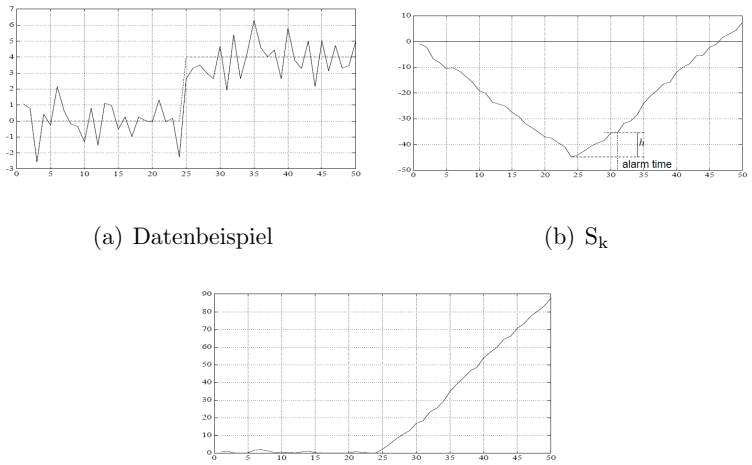
$$S_k = \sum_{i=1}^{k} s_i \qquad m_k = \min_{1 \leq i \leq k} S_j \qquad g_k = S_k - m_k \tag{2.6}$$

Wenn g_k kleiner einem empirisch ermittelten Grenzwert h ist, gehört k zur Distribution p_{θ_0}. Ist g_k größer oder gleich dem Grenzwert h, gehört k zur Distribution p_{θ_1}. Ein Wechselpunkt liegt also vor, wenn k zu einer anderen Distribution gehört als k-1. In Abbildung 2.1 ist der Verlauf der einzelnen Funktionen an einem Beispiel dargestellt. In Abbildung 2.1a sieht man den Verlauf des Datensatzes und den Verlauf des Medianwertes dieses Datensatzes. In Abbildung 2.1b ist der Verlauf der Funktion S_k für den gleichen Bereich wie in 2.1a dargestellt. In Abbildung 2.1c ist der Verlauf der Entscheidungsfunktion g_k zu sehen. Dabei fällt auf, dass an einem Wechselpunkt der Wert von g_k von einem konstanten Wert zu einem linear steigenden Wert wechselt[1].

In der Praxis ist diese Berechnung der CuSum allerdings nur verwendbar, wenn die beiden Distributionen vor und nach dem Wechselpunkt bekannt sind. Üblicherweise wird die Berechnung der log-likelihood durch andere Annäherungen ersetzt[6].

2.2 Semiüberwachtes Labeling

Der Ansatz des semiüberwachten Labelns soll den Labelprozess deutlich verkürzen. Hierfür muss nur ein kleiner Teil an gelabelten Daten bereits vorhanden sein. Diese



(a) Datenbeispiel (b) S_k

(c) g_k

Abbildung 2.1: CuSum Datenbeispiel[1]

Daten werden dafür benutzt, um Templates zu erstellen. In den Templates werden Startzeit, Endzeit, Sensordaten und die Bezeichnung des Labels gespeichert[2]. Mit dem Algorithmus Template Matching werden diese Templates erstellt und noch nicht gelabelte Daten auf ihre Zugehörigkeit zu einer der definierten Klassen überprüft[5].

Das Template Matching unterteilt sich also in zwei Abschnitte. Erst werden für jede Klasse ein oder mehrere Templates erstellt. Im zweiten Schritt wird jede neue Entität mit jedem Template verglichen, um das ähnlichste zu finden.

Ein Template ist eine Vorlage, aus der etwas neues geschaff werden kann. In diesem Fall dient es als Muster, um ähnliche Datenbereiche zu finden. Ein Template muss generell genug definiert sein, so dass auch Entitäten zugeordnet werden können, die leicht vom Template abweichen. Trotzdem muss es speziell genug sein, damit es einen Unterschied zu anderen Templates gibt. In diesem Ansatz wird mit zeitabhängigen Sensordaten gearbeitet. Der Verlauf der Sensorwerte innerhalb eines Labels hat eine bestimmte Form, welche sich in jeder Entität der gleichen Klasse wiederholt. Diese Form kann demnach als Template verwendet werden. Oft gibt es

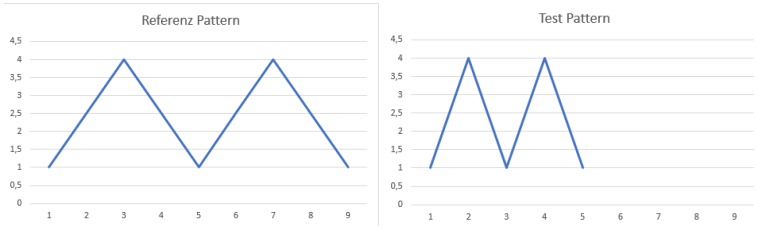
mehrere gelabelte Bereiche, die zu einer Klasse gehören. In diesem Fall kann der am besten erscheindende Bereich gewählt werden, mehrere Templates erstellt werden oder ein Durchschnitt aller Bereiche als Template gewählt werden.
Beim zweiten Schritt geht es darum die neuen Entitäten mit den Templates auf Unterschiede und Gleichheiten zu untersuchen. Sie sollen der Klasse zugeordnet werden, bei der sie die höchste Übereinstimmung mit dem zugehörigen Template haben. Diese Übereinstimmung kann auf verschiedene Arten bestimmt werden. Im Allgemeinen handelt es sich um Messmethoden für die Distanz. Im folgenden werden drei dieser Methoden genauer vorgestellt. Das Erste ist die euklidische Distanz. Dies ist die einfachste Methode, die verwendet werden kann, um die Übereinstimmung zwischen zwei Sequenzen von Punkten zu berechnen. X und Y sind je eine Sequenz von Punkten:

$$X = x_1, x_2, \ldots, x_i, \ldots, x_n \qquad Y = y_1, y_2, \ldots, y_i, \ldots, y_m$$
$$Dabei \quad gilt : 0 < i \leq m - n - 1 \quad und \quad n < m$$

Die euklidische Distanz wird wie folgt berechnet:

$$d(i) = \sqrt[2]{\sum_{k=1}^{n} (Y(i+k) - X(k))^2} \tag{2.7}$$

Diese Berechnung hat allerdings den Nachteil, dass Bereiche, deren Form dem Template ähnelt, aber deren Zeitraum anders ist, eine hohe Distanz haben. Dabei sollen auch diese Bereiche den Templates zugeordnet werden.
Eine Lösung für dieses Problem liefert das Dynamic Time Warping. Dieser Algorithmus sucht die beste Übereinstimmung mit einem Template unter der Betrachtung von Streckung und Beugung entlang der Zeitachse. Im ersten Schritt des Algorithmus wird eine mxn Matrix erstellt mit den Distanzen zwischen jedem Punktepaar y_i, x_i. In Abbildung 2.2a ist der Verlauf eines beispielhaften Templates (Y(t)) und in Abbildung 2.2b der Verlauf einer neuen Instanz zu sehen. Die daraus resultierende mxn Matrix ist in Tabelle 2.1 zu sehen. Ausgehend vom Startpunkt (1,1) wird ein Weg zum Punkt (m,n) gesucht, der die geringste summierte Distanz ergibt. In Ta-



(a) Referenz Y(t) (b) Test X(t)

Abbildung 2.2: Dynamic Time Warping: Beispiel

Tabelle 2.1: Dynamic Time Warping: mxn Matrix

Y(t)							
8	1	0	9	0	9	0	
7	2,5	2,25	2,25	2,25	2,25	2,25	
6	4	9	0	9	0	9	
5	2,5	2,25	2,25	2,25	2,25	2,25	
4	1	0	9	0	9	0	
3	2,5	2,25	2,25	2,25	2,25	2,25	
2	4	9	0	9	0	9	
1	2,5	2,25	2,25	2,25	2,25	2,25	
0	1	0	9	0	9	0	
		1	4	1	4	1	
		0	1	2	3	4	X(t)

belle 2.2 ist für das in Tabelle 2.1 begonnene Beispiel für jeden Punkt die niedrigste Distanz eingetragen, mit der dieser Punkt vom Start erreicht werden kann. Vom Endpunkt ausgehend wird nun der Weg ausgewählt, der die niedrigste Distanzsumme hervorgebracht hat. Diese Summe beschreibt die Ähnlichkeit der neuen Entität mit dem betrachteten Template. Die Klasse des ähnlichsten Templates wird der Entität zugeordnet.

Eine weitere Methode zur Bestimmung der Ähnlichkeit zwischen Template und neuer Entität ist das Rce. Diese Methode versucht Formähnlichkeiten in Signalsegmenten zu finden. Dies passiert durch das Verhältnis zwischen dem Korrelationkoeffizienten γ_{YX} und der euklidischen Distanz[5].

$$Rce = \frac{\gamma_Y X}{dist(Y, X)} \tag{2.8}$$

Tabelle 2.2: Dynamic Time Warping: Minimale Distanzen von Punkt (0,0)

Y(t)							
8	1	27	27	18	18	11,25	
7	2,5	27	18	18	9	11,25	
6	4	24,75	15,75	15,75	6,75	15,75	
5	2,5	15,75	15,75	6,75	6,75	9	
4	1	13,5	13,5	4,5	13,5	6,75	
3	2,5	13,5	4,5	4,5	6,75	9	
2	4	11,25	2,25	13,5	6,75	15,75	
1	2,5	2,25	4,5	6,75	9	11,25	
0	1	0	9	9	18	18	
		1	4	1	4	1	
		0	1	2	3	4	X(t)

Nachdem das Template Matching durchgeführt wurde, wird dem Nutzer des semi-überwachten Labelings die berechnete Klasse vorgeschlagen. Der Nutzer kann nun entscheiden, ob das Label richtig zugeordnet wurde. Wenn dies nicht der Fall ist, ändert er das Label und die entsprechenden Templates werden angepasst[2].

2.3 Transfer Labeling

Bei der Arbeit mit Sensordaten werden oft Machine Learning Algorithmen angewendet. Diese Algorithmen werden mit Sensordaten trainiert. Dadurch sind sie stark auf das benutzte System spezialisiert. Für jede Änderung des Systems (zum Beispiel das Hinzufügen eines Sensors oder eine neue Platzierung des Gesamtsystems) müssen die Algorithmen neu angepasst werden. Wenn dies nicht passiert, werden die Interpretationen der Daten deutlich schlechter. Für eine Anpassung der Algorithmen ist es notwendig, Sensordaten zu labeln, mit denen die Algorithmen ein neues Klassifizierungsmodell erlernen können. Das Labeln von Sensordaten ist sehr aufwendig und kostet viel Zeit. Der Ansatz des Transfer Labelings setzt an dieser Stelle an. Anstatt die Sensordaten manuell neu zu labeln, soll das Wissen vom alten System auf das neue/geänderte System übertragen werden. Dieser Ansatz basiert auf dem Prinzip des Transfer Learnings. Hierbei wird Wissen, welches aus einem Problem gelernt wurde, auf ein ähnliches Problem übertragen.

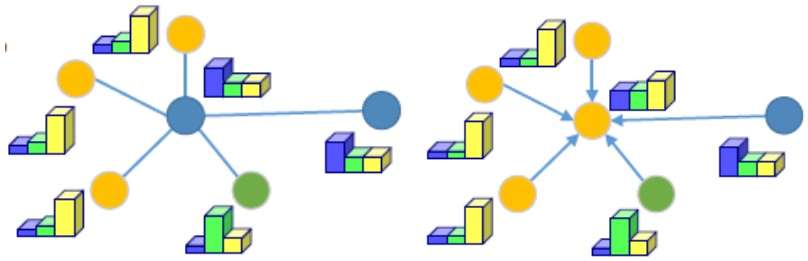
Für das Transfer Labeling ist also folgende Situation gegeben. Es existiert ein System, welches Sensordaten mit einer festen Anzahl an Sensoren aufnimmt, in Zukunft altes System genannt. Dieses alte System hat ein trainiertes Modell für die Erkennung von Aktivitäten. Diesem System wird ein neues System hinzugefügt, in Zukunft neues System genannt, welches noch kein trainiertes Modell enthält. Beide Systeme nehmen gleichzeitig Sensordaten auf. Das neue System soll nun seine Daten auf Grundlage des gemeinsam gesammelten Wissens beider Systeme labeln. Dafür soll es ein neues Klassifizierungsmodell erstellen. Als Resultat soll optimalerweise die Erkennungsrate des Gesamtsystems steigen. Es ist jedoch auch möglich das alte System durch das neue System zu ersetzen. Also ist es notwendig, dass das neue System die Daten ebenfalls möglichst genau labeln kann, damit es zu möglichst wenigen Erkennungsfehlern kommt.

Das neue System bekommt die Klassifizierungen des alten Systems übermittelt. Die beiden Systeme haben ihre Sensoren möglicherweise an unterschiedlichen Positionen am menschlichen Körper. Daher können Aktivitäten, die für eins der Systeme klar unterscheidbar sind, im anderen System zu Unsicherheiten führen. Deshalb wird vom alten System nicht einfach das gesetzte Label übertragen, sondern ein sogenanntes Semi-Label. Dieses ist ein 2-Tupel (a_i,W), bei dem das a_i das gesetzte Label des alten Systems ist und W der Wahrscheinlichkeitsvektor für die Wahrscheinlichkeiten, mit denen das alte System eines der anderen möglichen Label fälschlicherweise für a_i hält. Die Einträge des Wahrscheinlichkeitsvektors werden mit Hilfe der Likelihood-Deficiency (L_p) gebildet. Hier wird die Wahrscheinlichkeit berechnet, dass ein Sensor an Körperposition p die Aktivität a_1 mit der Aktivität a_2 verwechselt.

$$L_p(a_i, a_j) = \frac{n_p(a_i, a_j)}{\sum_{a_k \in A} n_p(a_i, a_k)} \tag{2.9}$$

$n_p(a_i, a_j)$ ist die Anzahl der Instanzen, bei denen der Sensor an Position p a_j als a_i erkannt hat. n_p wird mit Hilfe der Ambiguitätsrelation AR_p bestimmt. Hierbei werden Relationen als Elemente von AR_p betrachtet, wenn ein Sensor an Position

p, die Aktivität a_2 fälschlicherweise als Aktivität a_1 erkennt. Wenn eine Relation $((p, a_i), a_j)$ kein Teil von AR_p ist, bedeutet das, dass $L_p = 0$ ist.
Das neue System legt die erhaltenen Semi-Label über den gleichen Observationszeitraum und kombiniert seine eigenen Observationen damit. Hierfür wird das Verfahren Minimum Disagreement verwendet. Dieses Verfahren beruht auf der Annahme, dass Instanzen mit ähnlichen Sensorwerten auch zur gleichen Aktivität gehören müssen. Die Uneinigkeiten im Bezug auf die Label zwischen diesen Instanzen soll verringert werden. Dafür wird ein Ähnlichkeitsgraph gebildet, in dem die Knoten jeweils eine Instanz darstellen. Darin enthalten sind die Sensordaten dieser Instanz und die Semi-Label. Die Kanten sind gewichtet und die Gewichte sind die Ähnlichkeiten zwischen den Sensordaten der beiden verbundenen Knoten. Die Semi-Label jeder Instanz werden durch die Nachbarknoten angepasst. Nachbarn mit höherem Kantengewicht haben einen höheren Einfluss. Der Wahrscheinlichkeitsvektor des Semi-Labels wird zu kleinen Teilen in Richtung des Wahrscheinlichkeitsvektors der Nachbarn verschoben. In Abbildung 2.3 ist der Prozess zu erkennen. Durch diesen Prozess steigt die Genauigkeit der Semi-Label. Das wahrscheinlichste Label aus dem Semi-Label wird ausgewählt und zum Bilden eines neuen Klassifizierungsmodells im neuen System genutzt[7].



(a) Ähnlichkeitsgraph vor der Anpassung

(b) Ähnlichkeitsgraph nach der Anpassung

Abbildung 2.3: Minimum Disagreement[7]

3 Einsatzmöglichkeiten für unsere Projektgruppe

In diesem Kapitel möchte ich darauf eingehen, inwiefern sich die in Kapitel 2 beschriebenen Algorithmen für unsere Projektgruppe einsetzen lassen. Dazu werde ich jeweils auf die allgemeinen Einsatzmöglichkeiten im Bereich der Aktivitätserkennung eingehen und anschließend bewerten, inwiefern diese Algorithmen die Labelnachbearbeitung für uns als Projektgruppe beschleunigen können.

3.1 Change Detection

Der Change Detection Algorithmus, dessen Berechnung mit Hilfe der Class Separability durchgeführt werden, wurde im Paper [4] für die Korrektur von Labelgrenzen in Aktivitätsdaten eingesetzt. Hierfür wurden von mehreren Personen gelabelte Daten als Grundwahrheit angenommen. Als Testmenge wurden diese gelabelten Bereiche je einmal um 10 Prozent und einmal um 25 Prozent unter Verwendung der Normalverteilung mit $\mu = 0$ und $\sigma = 1$ und $\sigma = 1,5$ verschoben. Die Verschiebung ist nur gering, da sich das Paper auf kleine Fehler konzentriert. Auf den verschobenen Daten wurde der Change Detection Algorithmus angewandt und die Genauigkeit der korregierten Label überprüft. Das wurde mit Hilfe einer k-nächste-Nachbarn Klassifizierung durchgeführt. In Abbildung 3.1 ist zu erkennen, dass die korrigierten Daten eine signifikante Verbesserung der Genauigkeit im Vergleich zu den verschobenen Daten aufweisen. Jedoch kommen sie in den seltensten Fällen an die Genauigkeit der Grundwahrheit heran[4].

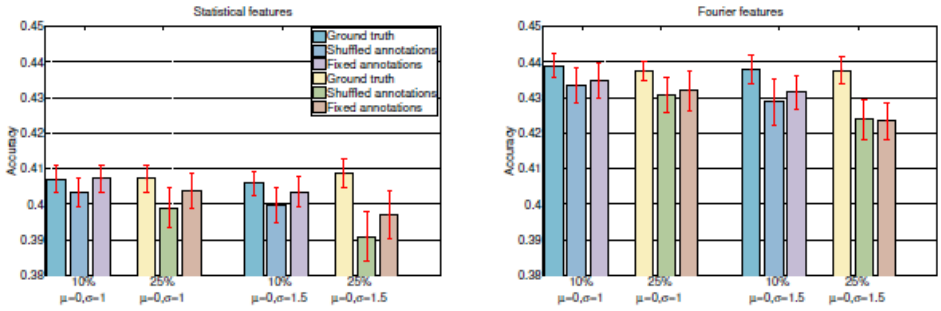


Abbildung 3.1: Ergebnisse von Class Separability[4]

Für unseren Anwendungsfall genügt es nicht kleine Fehler zu korregieren, da die mit dem Tablet gelabelten Daten höhere Abweichungen enthalten und nicht nur verschobene Grenzen. Aber der Anwendungsfall zeigt, dass dieser Algorithmus generell auf Aktivitätsdaten anwendbar ist.

Die Berechnung mit der kumulierten Summe wurde im Paper [6] für die Aktivitätserkennung eingesetzt. Hierfür wurden ähnlich wie in unserer Projektgruppe die Sensoren Accelerometer, Gyroskop und Magnetometer verwendet. Im Testszenario wurden mehrere Probanden aufgefordert einen festgelegten Bewegungsablauf auszuführen.

Auf den entstandenen Daten wurde der Change Detection Algorithmen ausgeführt. Es wurden die Magnitüde der Accelerometerdaten, die Magnitüde der Gyroskopdaten und je die Summe und das Produkt der vorhergenannten Daten verwendet. In Abbildung 3.2 sind die Ergebnisse des CuSum Algorithmus zu erkennen. Es wurde die Genauigkeit(Accuracy) und der Korrelationskoeffizient berechnet. Zusätzlich wird die Fensterlänge und der Grenzwert angegeben, bei denen die beste Genauigkeit erzielt wurde. Als Vergleichsdaten für die Berechnung der Genauigkeit und des Korrelationskoeffizienten diente ein Datensatz, bei dem die Wechselpunkte manuell festgelegt worden sind.

Es lässt sich erkennen, dass der CuSum-Algorithmus auf allen vier verwendeten Da-

Optimal value	MBCD-Acc.	MBCD-Ang.	MBCD-SUM.	MBCD-PROD.
Accuracy	$\mathbf{0.9576 \pm 0.0080}$	0.9414 ± 0.0164	0.9414 ± 0.0165	0.9434 ± 0.0154
Correlation coeff.	$\mathbf{0.9010 \pm 0.0153}$	0.8588 ± 0.0465	0.8583 ± 0.0469	0.8635 ± 0.0429
Window length	12.6167 ± 4.1869	15.2500 ± 6.4345	15.0500 ± 6.8720	14.5167 ± 6.6832
Threshold	$3.468e{-6} \pm 2.049e{-6}$	0.3588 ± 0.1731	0.3551 ± 0.1746	0.4346 ± 0.1690

Abbildung 3.2: Ergebnisse von CuSum[6]

tensätzen sehr gute Ergebnisse erzielt[6]. Daher halte ich den CuSum-Algorithmus für die Erkennung von Aktivitätsgrenzen für geeignet.
Im Allgemeinen hat die Erkennung von Wechselpunkten in einem Datensatz den Vorteil, dass beim Labelprozess nur noch die Aktivitäten für die Bereiche zwischen den Wechselpunkten festgelegt werden müssen. Der Bereich an sich muss bei sehr genauen Change Detection Algorithmen kaum noch angepasst werden. Dadurch dauert der Labelprozess deutlich kürzer, da das Anpassen der Grenzen die meiste Zeit verbraucht.

3.2 Semiüberwachtes Labeling

Das in Paper [5] verwendete Prinzip des Template Matching wurde in Paper [2] zur Erkennung von Aktivitäten verwendet. Hierfür wurden 22 primitive Aktivitäten betrachtet, die gruppiert wurden. Dadurch wurden insgesamt acht Sportaktivitäten betrachtet. Als Sensor wurde ein Accelerometer verwendet. Es wurden 48 Probanden dazu aufgefordert die acht Sportaktivitäten einen festgelegten Zeitraum über auszuführen, davon waren 19 Probanden übergewichtig und 29 normalgewichig.
Aus der Gruppe der normalgewichtigen Probanden wurden 50 Prozent der Daten für die Template Erstellung genutzt. Die gleichen Daten wurden auch zum Training von Klassifizierungsalgorithmen verwendet, um die Ergebnisse vergleichen zu können. Der Rest der Gruppe der Normalgewichtigen und der ganze Datensatz der Gruppe der Übergewichtigen wurde zum Testen verwendet. Die Genauigkeit der

Ergebnisse wird durch die Sensitivität angegeben, das ist die true-positiv Rate. Dabei lässt sich erkennen, dass es starke Unterschiede zwischen den betrachteten Aktivitäten gibt. In der Abbildung 3.3 sind die Sensitivitäten der erkannten Aktivitäten für jede Aktivität und für jeden Template Matching Ansatz angegeben. Der Algorithmus wird dabei je einmal auf den noch nicht genutzten Daten der Normalgewichtigen und einmal auf den Daten der Übergewichtigen ausgeführt. Auf beiden Datensätzen sind die Ergebnisse sehr ähnlich. Der Ansatz Rce schneidet am besten ab.

In Abbildung 3.4 ist der Vergleich zu den Ergebnissen der Klassifizierungsalgorith-

	Sensitivity%				
Activity (*)	Euclidean	DTW	DDTW	Correlation	*Rce*
Cycling (14)	82.2	87.7	83.2	48.2	88.4
Cross trainer (12)	19.8	15.5	19.5	5.0	7.6
Rowing (12)	38.3	43.4	54.0	10.6	52.3
Running (12)	86.0	70.5	62.5	94.0	73.8
Squatting (2)	73.9	8.7	47.8	82.6	91.3
Stepping (12)	10.5	14.8	39.9	86.0	68.1
Walking (14)	80.4	23.4	72.5	81.9	87.4
Weight lifting (4)	62.2	37.8	56.8	83.8	73.0

*Number of subjects per activity included in the training set.

(a) Normalgewichtige Probanden

	Sensitivity%				
Activity(*)	Euclidean	DTW	DDTW	Correlation	*Rce*
Cycling (14)	85.9	91.6	87.1	54.7	92.4
Cross trainer (12)	39.3	15.2	32.6	10.9	12.9
Rowing (12)	49.4	54.4	60.2	6.5	58.7
Running (12)	85.3	77.2	76.4	95.8	84.8
Squatting (2)	68.2	45.5	36.4	86.4	100.0
Stepping (12)	58.7	10.6	10.2	80.6	79.7
Walking (14)	80.5	24.3	70.4	88.0	91.2
Weight lifting (4)	57.7	75.7	79.3	63.1	64.0

*Number of subjects per activity included in the training set.

(b) Übergewichtige Probanden

Abbildung 3.3: Ergebnisse für Semiüberwachtes Labeling[5]

	Aggregated Sensitivity%		
Classifier	Normal	Overweight	Δ
Euclidean	66.1	72.6	6.5
DTW	43.4	47.5	4.1
DDTW	64.9	64.7	−0.2
Correlation	62.3	63.9	1.6
Rce	74.7	78.7	4.0
DT	81.9	80.6	−1.3
NB	79.6	79.7	0.1
LR	84.6	83.5	−1.1
ANN	86.7	85.9	−0.8

Abbildung 3.4: Ergebnisse für Semiüberwachtes Labeling im Vergleich zum Machine Learning[5]

men dargestellt. Hierbei fällt auf, dass die Klassifizierungsalgorithmen nur leicht

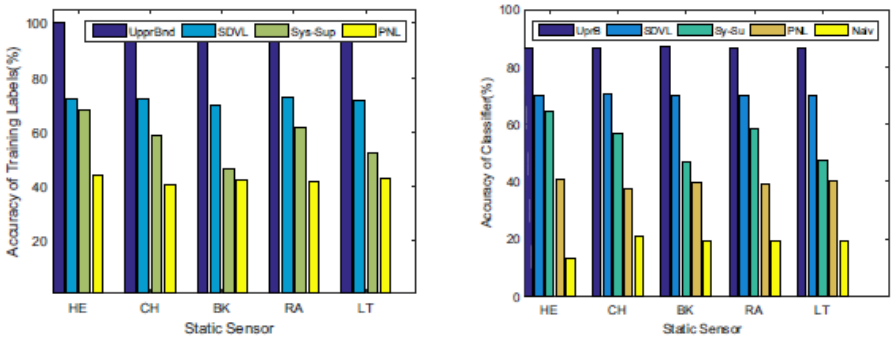
besser sind als der beste Template Matching Ansatz Rce. Das Template Matching hat im Vergleich auf beiden Datensätzen gleich gute oder sogar besser Ergebnisse auf den Daten der übergewichtigen Probanden. Es kann also besser generalisieren, als die Klassifizierungsalgorithmen, die in diesem Vergleich schlechtere Ergebnisse bei den Daten der Übergewichtigen als bei den Ergebnissen der Normalgewichtigen erzielten. Sie arbeiten also schlechter auf neuen Daten[5].
Durch die ständige Anpassung der Templates im semiüberwachten Labeling kann die Sensitivität noch gesteigert werden, aber auch wenn nur das Template Matching betrachtet wird, eignet sich diese Methode bereits für die Aktivitätserkennung.
Der Ansatz des semiüberwachten Labelings bedeutet deutlich weniger Aufwand bei der Labelnachbearbeitung, da in diesem Fall nur die vom Algorithmus falsch gelabelten Daten bearbeitet werden müssen. Alle anderen Daten müssen nicht betrachtet werden.

3.3 Transfer Labeling

Um das Transfer Labeling zu testen, wurde wie in Paper [7] beschrieben, ein Experiment mit neun Teilnehmern durchgeführt. Die Probanden haben 22 Aktivitäten ausgeführt. Dabei hatten sie Sensoren an Brust, Rücken, rechtem Arm, linker Hüfte und Kopf. Jeder der Sensoren wurde in je einem Durchgang als altes System verwendet. Die anderen Sensoren waren jeweils das neue System, welches sich an unterschiedlichen Positionen befand. Um herauszufinden, ob der Algorithmus gut generalisieren kann, wurde zusätzlich mit zwei öffentlich verfügbaren Datensätzen getestet (OPP und SDA). Auf Abbildung 3.5a ist die Genauigkeit der Labeldaten auf dem neuen System zu erkennen, also die direkt vom alten System übertragenen Label. In der Abbildung ist der durch das Experiment entstandene Datensatz dargestellt. Es wird jeder Sensor einmal als altes System verwendet. Jede dieser Varianten wird in der Abbildung dargestellt. Es werden verschiedene Ansätze des

Transfer Labelings verglichen. Dabei ist der SDVL Algorithmus (hellblau in der Abbildung) der in dieser Arbeit vorgestellte. Upper Bound (dunkelblau in der Abbildung) bezeichnet die Genauigkeit des alten Systems. Dabei fällt auf, dass der Algorithmus SDVL am besten abschneidet, er kommt jedoch nicht an die Upper Bound heran.

In Abbildung 3.5b ist die Genauigkeit der Klassifizierungen durch das neue System



(a) vom alten System übertragene Label (b) Klassifizierung durch das neue System

Abbildung 3.5: Ergebnisse von Transfer Labeling[7]

zu sehen. Der Aufbau der Grafik ist identisch zu Abbildung 3.5a. Hier fällt auf, dass wieder SDVL die besten Ergebnisse erzielt und das neue System durchschnittlich 15 Prozent schlechter ist als das alte System. Nach der Generierung einer eigenen Klassifizierung werden insgesamt bessere Ergebnisse erzielt als mit den direkt vom alten System übertragenen Labeln[7].

Ob sich der Einsatz dieses Algorithmus lohnt, muss abgewogen werden. Einerseits liefert er schlechtere Ergebnisse als ein Klassifizierungsalgorithmus, der mit manuell gelabelten Daten trainiert wurde. Andrerseits ist es eine sehr große Arbeitserleichterung, wenn die Daten nicht erneut gelabelt werden müssen.

Für unsere Projektgruppe ist der Algorithmus insgesamt uninteressant. Aber im weiteren Verlaufe des Gesamtprojektes könnte die Anwendung dieses Algorithmus sinnvoll sein, wenn ein Genauigkeitsverlust von etwa 15 Prozent akzeptabel ist.

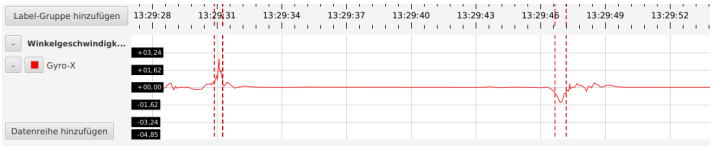
4 Umsetzung: Change Detection

Im Rahmen dieser Seminararbeit habe ich den Change Detection Algorithmus in die, in der Projektgruppe allgemein genutzte, Software MAmkS integriert. Dabei habe ich mich auf die Berechnung der Wechselpunkte mit der Class Separability konzentriert (vgl. Kapitel 2.1.1).
Dazu habe ich jede Datenreihe einzeln betrachtet und einzeln Wechselpunkte berechnet (Accelerometer-X/-Y/-Z, Gyroskop-X/-Y/-Z und Magnetometer-X/-Y/-Z). Es wird ein Sliding Window über die jeweilige Datenreihe geschoben. Das Sliding Window Verfahren war in der Anwendung MAmkS bereits integriert. Für jeden Punkt in jedem Fensterausschnitt habe ich die Between-Class Scatter berechnet. Für jeden Ausschnitt hat der Algorithmus den Punkt als möglichen Wechselpunkt gewählt, der die niedrigste Between-Class-Scatter hat. Im nächsten Schritt habe ich mit Hilfe des t-Tests validiert, ob diese Punkte wirklich Wechselpunkte sein können. Alle akzeptierten Wechselpunkte habe ich einer Liste hinzugefügt. Zwischen je zwei Wechelpunkten habe ich anschließend ein Label gelegt, um die Unterscheidung zwischen zwei Aktivitäten deutlich zu machen.
Bei der Umsetzung sind einige Probleme aufgetreten. Zum einen ergab der Algorithmus keine guten Ergebnisse für die Accelerometerdaten. Hier wurden oft höhere Peaks innerhalb der Gehbewegung als Wechselpunkte identifiziert. Deshalb habe ich versucht, die Daten vorzubearbeiten. Dafür habe ich verschiedenne Filter ausprobiert, um die Daten zu glätten. Einer davon war der Medianfilter. Dieser hat entweder zu wenig geglättet und das Problem blieb bestehen oder er hat zu stark geglättet und die Daten bestanden nur noch aus einer geraden Linie. Als nächstes habe ich den Gauß-Filter ausprobiert. Der hat die Kanten im Graph abgerundet. Das Problem mit der Erkennung der Peaks blieb jedoch bestehen. Die

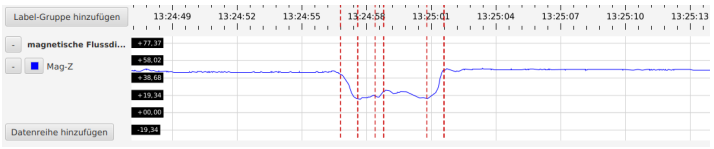
besten Ergebnisse ergab eine Vorverarbeitung, die die Höhe der Peaks berechnet. Eine weitere Möglichkeit wäre jedoch die Berechnung der Entropie. Das zweite Problem war, dass pro Fensterausschnitt nur ein Wechselpunkt gefunden wurde. Liegen die Wechselpunkte zu nah beieinander, wird nur einer davon identifiziert. Das Problem kann mit einer möglichst kleinen Fenstergröße oder mit sich überlappenden Fensterausschnitten gelöst werden.
Im Allgemeinen funktioniert der Algorithmus auf den Gyroskop- und Magnetometerdaten (s. Abbildung 4.1a und 4.1b) besser als auf den Accelerometerdaten. Das liegt vermutlich daran, dass diese Daten ruhiger verlaufen und nicht innerhalb einer Aktivität große Schwankungen aufweisen. Auf den Accelerometerdaten funktioniert die Wechselpunkterkennung nur soweit, dass zwischen dynamischen und statischen Aktivitäten Wechselpunkte gefunden werden können (s. Abbildung 4.1c). Innerhalb eines dynamischen Bereiches findet der Algorithmus meistens keine Wechselpunkte. Allgemein lässt sich allerdings feststellen, dass der Algorithmus in dieser Konfiguration keine Ergebnisse liefert, die als sichere Grenzen zwischen Aktivitäten genutzt werden können. Leider spezifiziert das Paper [4], in dem dieser Algorithmus für die Grenzenfindung zwischen Aktivitäten benutzt wurde, nicht genauer, inwiefern die Sensordaten vorbearbeitet wurden. Das ist denke ich ein Ansatz, mit dem man die Ergebnisse noch verbessern könnte. Auch wäre es möglich über eine andere Validierungsmethode für die Wechselpunkte nachzudenken.

4.1 Aktivitätserkennung

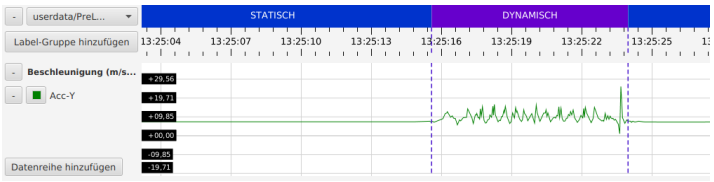
Da die Wechselpunkte zu ungenau für die genaue Festlegung von Grenzen zwischen Akivitäten ist, habe ich ausprobiert, sie als grobe Anhaltspunkte für die Erkennung von Aktivitäten zu verwenden. Zusätzlich habe ich mich mit der Biomechanik einiger Aktivitäten beschäftigt und die Bewegungen auf die Sensordaten übertragen. Die Gleichmäßigkeiten, die ich dabei gefunden habe, habe ich in mein Programm



(a) Beispiel: Wechselpunkte Gyroskop



(b) ÜBeispiel: Wechselpunkte Magnetometer



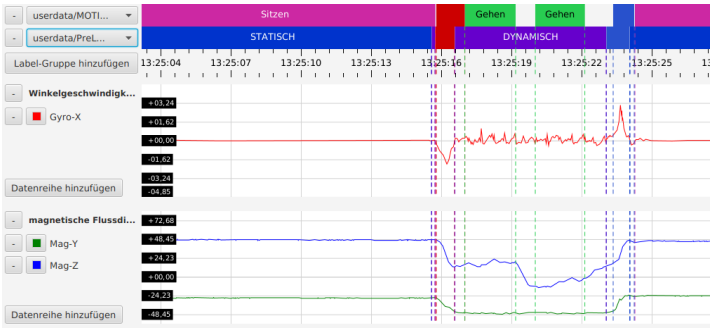
(c) Beispiel: Trennung statisch/dynamisch bei den Accelerometerdaten

Abbildung 4.1: Beispiele für gefundene Wechselpunkte

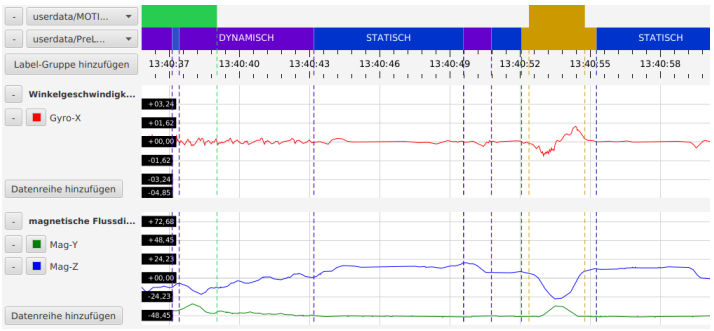
integriert, um automatisch diese Aktivitäten zu identifizieren. Ich habe mich mit den Aktivitäten Hinsetzen, Aufstehen und Hocken beschäftigt.

Beim Hinsetzen bewegt der Mensch sich nach hinten und unten. Das Becken dreht sich dabei nach hinten. Diese Bewegungen lassen sich auf die Magnetometer Y-Achse, die Magnetometer Z-Achse und die Gyroskop X-Achse übertragen. Die Gyroskop X-Achse zeichnet Rotationen um das Becken nach vorne und nach hinten auf, da die Sensoren etwa auf Beckenhöhe angebracht sind. Deshalb ist das Nachhintenbeugen beim Hinsetzen durch einen Peak auf der Gyroskop X-Achse gekennzeichnet. Die Werte sind im Normalfall bei 0, also dauert die Rotation an bis nach dem Peak wieder die Nulllinie erreicht ist. Die Magnetometer Y-Achse zeichnet Änderungen in vertikaler Richtung auf. Beim Hinsetzen nähert sich der Mensch dem Boden und damit dem Erdkern. Also wird die magnetische Flussdichte höher und somit steigt der Wert der Magnetometer Y-Achse. Die Magnetometer Z-Achse

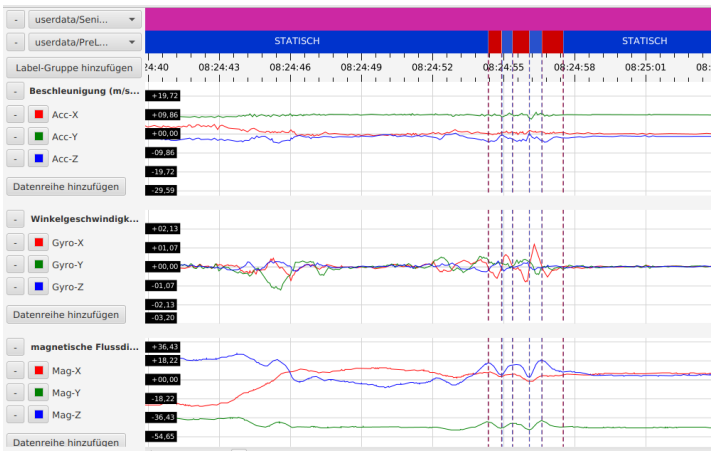
zeichnet Änderungen in die horizontale Richtung auf. Also steigt der Wert durch die Rückwärtsbewegung beim Hinsetzen ebenfalls an.
Beim Aufstehen gelten ähnliche Beobachtungen wie beim Hinsetzen. Diesmal sind die Änderungsrichtungen jedoch vertauscht. Der Mensch bewegt sich nach vorne oben. Also sinken die Werte von der Magnetometer Y- und Z-Achse. Die Drehung um die Gyroskop X-Achse geht nun nach vorne. Also haben die Werte einen Peak nach unten.
Das Hocken lässt sich in zwei Phasen unterteilen: Die Abwärtsbewegung und die Aufwärtsbewegung. Bei der Abwärtsbewegung bewegt der Mensch sich nach vorne unten. Demnach hat die Gyroskop X-Achse einen Peak nach unten, während die Magnetometer Y-Werte steigen. Die Magnetometer Z-Werte sinken durch die Vorwärtsbewegung beim Hinuntergehen. Die Aufwärtsbewegung geht nach hinten oben. Also haben die Gyroskop X-Werte einen Peak nach oben, die Magnetometer Y-Werte steigen und die Magnetometer Z-Werte sinken.
Um die Aktivitäten anhand der Wechselpunkte zu berechnen, werden also die Datenreihen Magnetometer-Y, Magnetometer-Z und Gyroskop-X betrachtet. Wenn auf allen drei Datenreihen in einem ähnlichen Bereich Wechselpunkte identifiziert werden, wird überprüft, ob die Entwicklungen der Werte zu den oben beschriebenen Aktivitätsmustern passen. Wenn dies der Fall ist, wird an diesen Stellen das entsprechende Label gesetzt.
Wie in Abbildungen 4.2a und 4.2b zu erkennen, erkennt der Algorithmus die Aktivitäten auf guten Daten sehr gut. Die obere Reihe sind in den Abbildungen jeweils die manuell gelabelten Daten und die untere Reihe sind die berechneten Label. Bei Daten, in denen die Aktivitäten unruhig sind, erkennt der Algorithmus auch an Stellen Aktivitäten, an denen eigentlich keine sein sollten. (s. Abbildung 4.2c)
Diese Art der Aktivitätserkennung eignet sich als ein erster Schritt im Labelprozess. Jedoch darf sich auf die Label nicht verlassen werden. Eine manuelle Nachprüfung bleibt auch für die drei erkannten Labelarten notwendig.



(a) Beispiel: Aufstehen und Hinsetzen



(b) Beispiel: Hocken



(c) Beispiel: Rauschen führt zu falsch erkannten Aktivitäten

Abbildung 4.2: Beispiele für erkannte Aktivitäten

5 Fazit

In dieser Seminararbeit wurden verschiedene algorithmische Lösungen betrachtet, die den großen Zeitaufwand des Labelns verringern sollen.
Im Allgemeinen lässt sich darüber aussagen, dass sich alle drei Arten von Algorithmen auf ihre Weise dafür eignen.
Die Change Detection Algorithmen reduzieren den Aufwand dadurch, dass sie es der labelnden Person abnehmen, die Grenzen zwischen verschiedenen Aktivitäten zu setzen.
Das Semiüberwachte Labeling hilft der labelnden Person in sofern, dass sie nur noch kontrollieren muss, ob der Algorithmus die richtigen Entscheidungen trifft. Wenn die richtigen Entscheidungen getroffen wurden, muss sie sich um den Anteil des Datensatzes nicht mehr kümmern. Nur im Falle einer falschen Entscheidung seitens des Algorithmus muss eingegriffen werden.
Das Konzept des Transfer Labelings ändert nichts an der Tatsache, dass ein Datensatz erstmal per Hand gelabelt werden muss. Es setzt da an, wenn das System, mit dem der Datensatz erfasst wird, geändert werden soll. In diesem Fall verringert es den Aufwand des Labelns, da für das neue System nicht noch einmal gelabelt werden muss.
In der praktischen Umsetzung des Change Detection Algorithmus konnten noch keine guten Ergebnisse erreicht werden. Mit weiteren Versuchen den Algorithmus und die Vorverabeitung des Datensatzes anzupassen, könnten vermutlich bessere Ergebnisse erreicht werden. Die Erfolge, die in der Literatur erreicht wurden, sprechen dafür, dass der Algorithmus an sich für unseren Anwendungsfall brauchbar ist.

Literatur

[1] Michèle Basseville und Igor V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[2] Alexander Diete, Timo Sztyler und Heiner Stuckenschmidt. „A smart data annotation tool for multi-sensor activity recognition“. Englisch. In: *2017 IEEE International Conference on Pervasive Computing and Communications Workshops : PERCOM Workshops*. Piscataway, NJ: IEEE Computer Soc., 2017, S. 111–116.

[3] Ankur Jain und Yuan-Fang Wang. *A New Framework for On-Line Change Detection*.

[4] Reuben Kirkham u. a. „Automatic Correction of Annotation Boundaries in Activity Datasets by Class Separation Maximization“. In: *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*. UbiComp '13 Adjunct. Zurich, Switzerland: ACM, 2013, S. 673–678.

[5] J. Margarito u. a. „User-Independent Recognition of Sports Activities From a Single Wrist-Worn Accelerometer: A Template-Matching-Based Approach“. In: *IEEE Transactions on Biomedical Engineering* 63.4 (2016), S. 788–796.

[6] Alberto Olivares u. a. „Detection of (In)activity Periods in Human Body Motion Using Inertial Sensors: A Comparative Study“. In: *Sensors* 12.12 (2012), 5791–5814.

[7] Seyed-Ali Rokni und Hassan Ghasemzadeh. „Synchronous Dynamic View Learning: A Framework for Autonomous Training of Activity Recognition Models using Wearable Sensors“. In: (Apr. 2017).

B.8. Realtime event processing von Sensordaten & Aktivitätserkennung

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Abteilung Assistenzsysteme und Medizintechnik

Projektgruppe Mobilitäts-Assessments
mit körpernahen Sensoren für zuhause
SoSe 17 - WiSe 17/18

Seminararbeit: Realtime event processing von Sensordaten & Aktivitätserkennung

Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers
Autoren: Patrick Warszewik

14. Februar 2018

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In dieser Seminararbeit beschäftige ich mich mit dem Thema Realtime event processing von Sensordaten & Aktivitätserkennung. Die Seminararbeit findet im Rahmen der Projektgruppe Mobilitätsassessments mit körernahen Sensoren statt, die von jedem Gruppenmitglied eine individuelle Leistung in Form dieser Einzelarbeit erfordert. Dabei wird im nächsten Kapitel der Hintergrund dieser Seminararbeit beschrieben die einen kleinen Einblick in die Arbeit unserer Projektgruppe darstellt und die Relevanz dieser Seminararbeit, die thematisch an die Projektgruppe anknüpft, verdeutlicht. Im Kapitel Rahmenbedingungen werden die Grenzen dieser Ausarbeitung beschrieben die den Umfang dieser Ausarbeitung im Rahmen zu halten. Die Kapitel Apache Kafka und Apache Storm beschreiben die Projekte der Apache Foundation, die ich zu meiner Lösungsfindung untersucht habe und deshalb jeweils ein separates Kapitel besitzen. Im Kapitel Lösungansatz stelle ich mein Konzept einer Beispiel Architektur vor und zeige welche Rolle Apache Kafka und Apache Storm darin haben werden. Am Ende stelle ich mein Fazit vor die zusammen mit einem Ausblick, den Abschluss dieser Seminararbeit darstellt.

2 Hintergrund

Der Hintergrund dieser Seminararbeit liegt im Rahmen dieser Projektgruppe die im April 2017 begann und noch bis April 2018 laufen wird. Dabei besteht die Kernaufgabe des Projektes, aus dem antrainieren eines Algorithmus zur Erkennung und Klassifikationen von Aktivitäten, aus Daten die von Sensorgürteln kommen. Die Sensorgürtel tragen derzeit ausgewählte Probanden der Versa Studie die Daten, in mehreren Dimensionen über die Aktivitäten ihres Alltages, aufzeichnen. Bisher werden die Daten des Sensorgürtels per Herstellersoftware auf einen lokalen Computer übertragen. Eine kabellose Übertragung ist derzeit nicht möglich, da der Sensorgürtel mit dieser Funktionalität nicht ausgestattet ist. Nach Aussage von Dr. rer. nat. Sebastian Fudickar gibt es derzeit Abschlussarbeiten, die sich mit der Entwicklung und Umsetzung einer drahtlosen Übertragung der Daten per Bluetooth beschäftigen. Somit kann die Annahme betroffen werden, dass der Sensorgürtel bereits eine Schnittstelle zur drahtlosen Kommunikation besitzt und nicht noch zusätzlich ein Konzept hinzu erdacht werden muss.

Um das noch Thema dieser Seminararbeit besser nachvollziehen zu können, soll eine User Story bzw. ein Anwendungsbeispiel folgendes Szenario beschreiben.

Die User Story lautet: *Wenn ich den Sensorgürtel benutze, möchte ich sofort die Aktivität sehen, welche ich gerade tätige.*

Daraus lassen habe ich folgende Funktionalität ableiten:

1. Die Daten sollen in Echtzeit Feedback darüber geben, welche Aktivität der Klassifikationsalgorithmus erkannt hat.

2. Das Ergebnis soll gespeichert werden, damit dies vom Benutzer oder anderen relevanten Personen einsehbar ist.

3. Die Daten werden über eine Drahtlose Kommunikation an den Klassifikationalgorithmus gestreamt, der die übergebenen Daten klassifiziert und das Endergebnis anschließend ausgibt.

Hierbei erläutere ich noch kurz das Thema der Echtzeitdatenverarbeitung bzw. des Realtime event processings. Mit dem realtime event processing ist es möglich sogenannte Datenströme oder Streams sequentiell zu verarbeiten während der Stream weiterläuft und man ihn damit nicht unterbricht. Dazu erwähne ich noch, warum dieses Thema eine solche Relevanz besitzt um sich damit in einer Seminararbeit zu beschäftigen.

Zum einen ist Seminarthema mit den immer öfter vorkommenden Begriffen Big Data und Internet of Things verbunden. Hinter Big Data steckt die Bezeichnung, mit immer größeren Datenmengen konfrontiert zu werden, die an Umfang nicht abnehmen sondern stetig wachsen und es darum geht, diese effizient zu verarbeiten. Unter Internet of Things steckt die Bedeutung, dass wir Daten aus unserem Alltag wahrnehmen und diese verarbeiten können. Dazu gehören auch Gegenstände die mittlerweile in der Lage sind Daten zu erfassen, in Form von Sensoren in Produktionsmaschinen oder modernen Fahrzeugen. Wichtig zu erwähnen ist auch, das die Erwartungshaltung der Nutzer von Interaktionssystemen so weit fortgeschritten ist, das Ergebnisse möglichst in Echtzeit ohne große Latenzen erfolgen sollen, da sonst der Benutzer sämtliches Interesse an der Benutzung des Systems verliert. Damit soll das realtime event processing abhilfe schaffen, indem große Datenmengen bereits während des Übertragungsvorgang verarbeiten werden, ohne dabei den Datenstrom erst auf einer Datenbank zu speichern da dies auf sehr große Datenmengen mit enormen Zeitverlust verbunden ist.

3 Rahmenbedingung

In diesem Kapitel werden die Rahmenbedingungen beschrieben welche ich mir zu dieser Seminararbeit gesetzt habe. Dazu habe ich mir den Fokus gesetzt, der dem Arbeitsaufwand dieser Seminararbeit gerecht wird und dieser in angemessener Zeit zu schaffen ist. Der Fokus liegt darin, das Gegenstand dieser Seminararbeit ein Konzept ist, das es zu Entwicklern galt, wie eine Architektur aussehen könnte, die die Gürteldaten in Echtzeit verarbeiten und das Ergebnis der Echtzeit-Klassifizierung als Output wieder zurückgibt. Dazu stehen einem viele Möglichkeiten und Vorgehensweisen einer Umsetzung zur Verfügung. Es gilt aber auch noch den Kostenfaktor zu erwähnen, der in dieser Arbeit natürlich gleich Null entsprechen muss. Damit ist gemeint, dass sämtliche Technologien oder Systeme auf Freeware oder Open Source Basis basieren müssen, da die Möglichkeit Geld in dieser Seminararbeit zu verwenden nicht gegeben ist. Zusätzlich sollen auch damit sämtliche Schritte dieser Arbeit möglichst einfach nachvollziehbar gestaltetet sein, damit die Lehrenden meine Vorgehensweise anhand der freien Technologien nachvollziehen können. Daher wurde vorweg schon das Kriterium von Open Source oder Freeware System und Werkzeugen gesetzt. Mit MATLAB wäre eine solche Umsetzung beispielsweise möglich. Da ich persönlich jedoch keinerlei Erfahrung im Umgang von MATLAB habe und das anlernen der Benutzung von MATLAB alleine schon ein Risikofaktor darstellen würde, nicht rechtzeitig mit der Seminararbeit fertig zu werden. Daher beschränken sich hier sämtliche Konzeptentwürfe auf Java Technologie in denen ich zum derzeitigen Kenntnisstand die meiste Erfahrung vorweisen kann. Als mögliche Plattformen nannten die Themensteller dieser Seminararbeit, die Apache Kafka und Apache Storm Plattformen. Beide Plattformen können in Kombination zu einer effizienten Lösung meiner Seminararbeit führen. Dazu bedarf es aber zunächst einer

Einarbeitung in beide Plattformen meinerseits, da ich vorher beide Plattformen noch nicht verwendet bzw. genutzt habe. Darum bestand eine Komponente dieser Arbeit auch sich mit beiden Plattformen vertraut zu machen und ihre Funktionsweise zu verstehen und diese anschließend auf meine Problemstellung anzuwenden. Als weiteren Punkt wird die Prämisse getroffen, dass der Sensorgürtel bereits in der Lage ist die Sensordaten über eine Drahtlose Kommunikation wie Bluetooth zu versenden und ich der der Lage bin, diese Daten zu empfangen.

4 Apache Kafka

In diesem Kapitel werden die Begriffe und Grundzüge des Apache Projekts Kafka[1] erläutert, auf denen mein späterer Lösungsansatz basiert. Dabei ist Kafka mit einem sogenannten commit-log Dienst für Datenströme zu vergleichen. Konkret geht es dabei die Aufzeichnung sämtlicher Daten die den Datenstrom passiert haben. Dazu kann die Aufzeichnung an beliebigen Stellen zurückverfolgen werden und den Datenstrom an dieser Stelle wieder aufrufen bzw. konsumieren lassen. In der Vergangenheit wurde viel an Kafka weiter entwickelt und seit November 2017 wurde die Version 1.0 veröffentlicht was die Software auch für den produktiv Betrieb einsetzbar macht. Vorteile mit denen Kafka geworben wird, bestehen in seiner Hochverfügbarkeit von Daten, in dem sämtliche commit-logs repliziert werden, um im Falle eines Verbindungsabbruches wieder an vorheriger Stelle anknüpfen zu können. Zusätzlich bietet Kafka geringe Schreib und Leselatenzen, indem die Daten aus dem Datenstrom sofort auf der Festplatte geschrieben werden und dadurch Umwege ausgeschlossen werden, welche die Latenzzeit vergrößern könnten. Dadurch erreicht Kafka eine Datendurchsatzrate von mind. 100.000 Daten pro Sekunde[2]. Weiterer Vorteil bietet Kafka in seiner Skalierbarkeit die es möglich macht die Größe des Clusters während des Betriebs nach belieben zu vergrößern oder zu verkleinern. Im Grunde funktioniert Kafka nach dem Publish-Subscribe Prinzip das an dem Entwurfsmuster Beobachter angelehnt ist. Darin wird ein Beobachter über neu erhaltene Daten informiert und kann auch diese beziehen. Das Herzstück der Kafka Architektur bildet dabei das Cluster. Sämtlich Nachrichten die vom Publicher veröffentlicht werden gehen an das Cluster und werden anschließend an alle Subscriber mittels einer TCP Verbindung entsandt. Im Kafka Kontext werden für Publisher

[1] https://kafka.apache.org/

[2] https://www.innoq.com/de/articles/2013/08/log-daten-verarbeiten-mit-kafka/

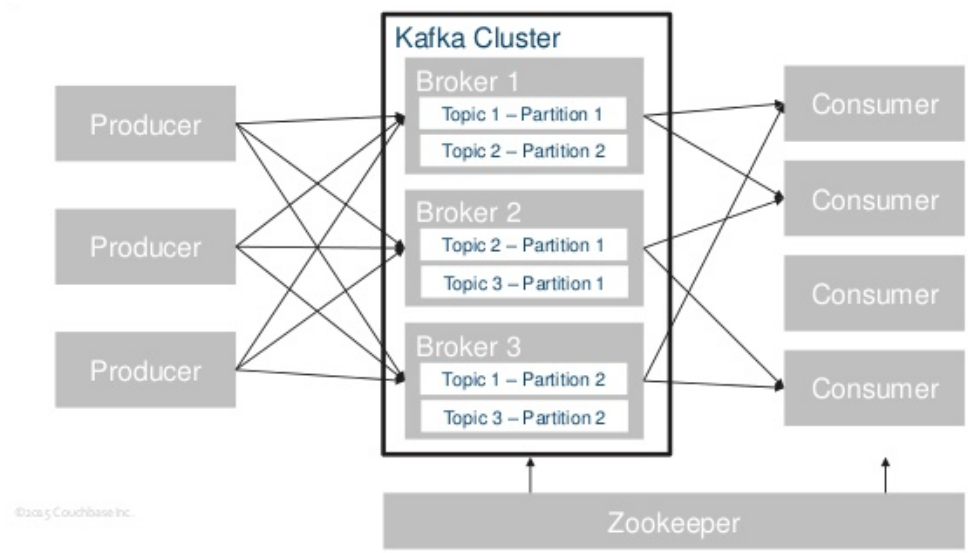


Abbildung 4.1: Kafka Architektur

und Subscriber die Begriffe Producer und Consumer verwendet. Die Abbildung 4.1[3] soll visuell darstellen, wie die einzelnen Kafka Komponenten miteinander interagieren diese im Bezug zueinander stehen.

Ganz links sind die Producer abgebildet, die Datenströme an das Cluster senden. In der Mitte ist das Cluster als das größte Rechteck dargestellt, das aus mehreren Topics, Partitionen und Brokern bestehen kann. Ein Topic entspricht in Kafka einem Stream der einen beliebigen Namen enthalten kann. Wenn ein Producer Daten an das Cluster senden will, muss er dabei den Namen des Topics als Parameter angeben, damit der Producer weiß an welches Topic er die Daten schicken muss. Ein Topic kann dabei aus mindestens einer oder mehreren Partionen bestehen. Die Anzahl der Partitionen ermöglichen hierbei parallele Zugriffe auf die Topics. Ein Consumer nimmt in diesem Kontext die Rolle des Subscribers ein, indem er einen

[3] https://www.slideshare.net/willgardella/real-time-messages-at-scale-with-apache-kafka-and-couchbase

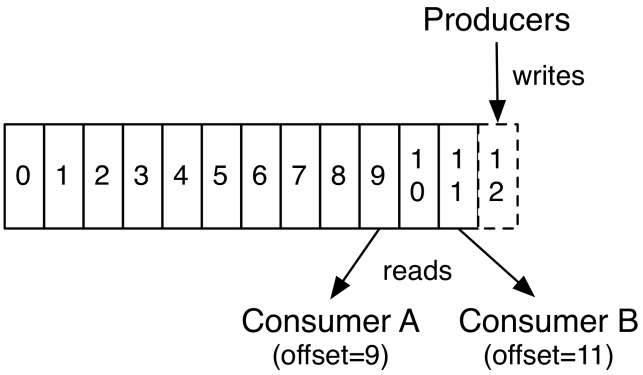


Abbildung 4.2: Anatomie eines Topics

Topic abonniert. Man benutzt hierfür das Wort abonnieren da der Consumer auf neue Daten des Topics passiv wartet und nicht während des Betriebs die Anwendung blockiert, wenn er ggf. keine Daten erhält. In der Abbildung sind auch innerhalb des Clusters sogenannte Broker zu erkennen. Die Broker stehen für die Server oder Knoten innerhalb eines Clusters die eine Partition eines Topics betrieben. Als unterste Instanz auf dem Bild sei noch abschließend Zookeeper zu erwähnen die für die Verwaltung zwischen den Brokern Producern und Consumer zuständig ist. Die Daten des Producers werden als sequentielle Key-Value Paare im Topics angelegt. Die Paare werden daraufhin mit einer eindeutigen ID dem sog. Offset nummeriert. An dieser Stelle wird ein weiterer Grund für Kafkas hoher Performanz erklärt, indem die Consumer sich selber den Offset eines Topics, welchen die zuletzt gelesen haben, merken. Die Abbildung 4.2[4] zeigt den Aufbau eines Topics anhand eines Producers der Daten schriebt und zwei Consumer zu selben anfangen den Topic anfangen an unterschiedlichen Stellen zu lesen.Natürlich gibt es noch mehr über Kafka zu berichten, um jedoch nicht den Fokus der Seminararbeit zu verlieren, sei das bisherige Wissen über Kafka mehr als ausreichend , um ein Verständnis für der Lösungsdarstellung zu haben.

[4]https://kafka.apache.org/documentation

5 Apache Storm

Bei Apache Storm[1] handelt es sich um eine Echtzeitdatenverarbeitungsplattform, die ursprünglich von Twitter aufgekauft worden ist. Mittlerweile befindet sich Storm im Portfolio von Apache, die zu einem Open Source Projekt veröffentlicht wurde. Seit September 2017 befindet sich die Plattform in seiner aktuellsten Version 1.0.5. Neben seiner Echtzeitdatenverarbeitung bietet Storm auch Performanz Vorteile mit einer geringen Latenzzeit,die es laut Apache, möglich macht etwa eine Million Nachrichten pro Sekunden pro Knoten zu verarbeiten[2]. Als weiteren Vorteil garantiert Storm im Fehlerfall keinen Datenverlust dadurch das dieselbe Nachricht nochmals verarbeitet wird. Genau wie bei Kafka bietet Storm eine Skalierbarkeit die es erlaubt im laufendem Betrieb, Knoten im Cluster hinzuzufügen und Storm dies erkennt und auf diese Ressourcen zurückgreift. Auch wie bei Kafka gibt es in der Storm Domäne ein paar Begriffe die eine kurzen Erklärung benötigen, um das Verständnis der späteren Lösung nachvollziehbarer zu gestalten. Im direkten Vergleich zu Kafka fällt dieser Begriffsumfang deutlich geringer aus. Im Grunde besteht eine Strom Komponente aus einer Topolgie. Eine Topolgie kann als gerichteter Graph beschrieben werden, der aus Spots und Bolts besteht. Ein Spout stellt eine Quelle dar, die einen Stream einließt und als Output den Stream an Bolts weiterleitet. Ein Bolt stellt eine Komponente die einzelne Funktionen am Stream ausführen und den Output als neues Streamobjekt an einen neuen Bolt weiterleiten. Solche Funktionen können beispielsweise Filterungen, Aggregation oder auch die Verbindung zu externen Datenquellen sein. Die Abbildung 5.1[3] soll einmal die Topolgie zum besseren Verständnis graphisch darstellen. An der Abbildung lässt sich erkennen, das eine

[1] http://storm.apache.org/index.html
[2] http://storm.apache.org/
[3] https://hortonworks.com/hadoop-tutorial/processing-trucking-iotdata-with-apache-storm/

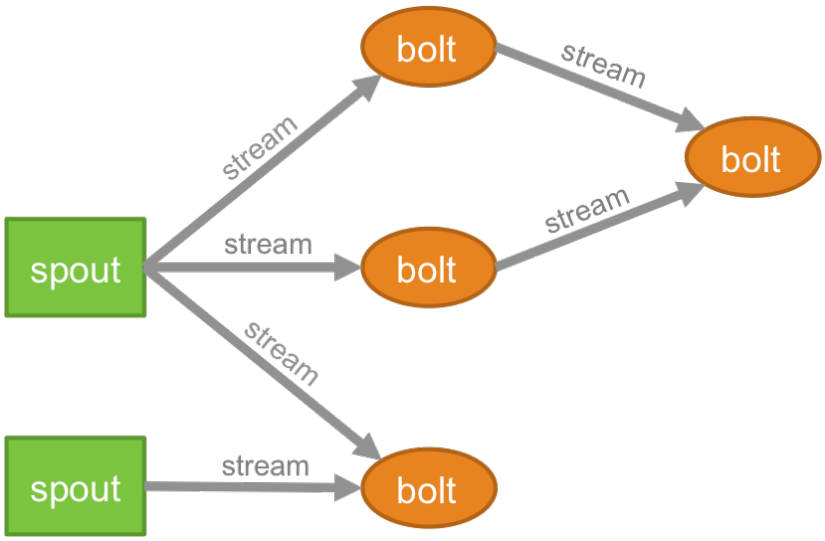


Abbildung 5.1: Eine Beispiel Storm Topologie

Topolgie nicht nur aus einem sondern mehren Spouts und Bolt bestehen kann. In solchen Kombinationen ist man auch in der Lage komplexe Stream Verarbeitungen durchzuführen. Natürlich sind das, auch wie bei Kafka, nicht die einzigen Komponenten zu Storm. Der Grad an Erläuterungen wurde so weit gesetzt das er lediglich zu späterer Lösungsvorstellung genüge da wir auch hier der Fokus der Seminararbeit verloren geht, wenn weitere Komponenten oder auf tieferen Ebenen die Storm Plattform erläutert werden würde.

6 Lösungsansatz

In diesem Kapitel wird nun der Lösungsansatz ausführlich beschrieben, mit dem die Daten des Sensorgürtels kabellos in Echtzeit verarbeitet werden, mithilfe Klassifikationsalgorithmuses. Die zentrale Idee ist hierbei, Kafka und Storm zu kombinieren bzw. Storm in Kafka zu integrieren. Die Arbeitsteilung würde dabei so aussehen, dass Kafka die Daten aus dem Sensorgürtel in das Cluster streamt und Storm die Daten aus dem Cluster ließt. Anschließend verarbeitet Storm die Daten und persistiert die Ergebnisse an verschiedenen externen Stellen wie Datenbanken oder Dateien. Die Frage ob nun Kafka Consumer oder Storm Spout den Stream aus dem Cluster ließt, lässt sich durch eine Integration effektiv lösen. Und zwar würde der Spout eine Quelle aus dem Kafka Cluster beziehen und die Rolle eines des Kafka Consumers einnehmen. Damit würde es zu einer Überschneidung beider Plattformen kommen, da beide Komponenten die selbe Aufgabe haben, indem die Daten aus dem Cluster abonniert bzw. gelesen werden. Diese Idee ist derweil nicht neu und man bekommt oft den Namen KafkaSpout[1] zu lesen wenn es um die Integration von Storm und Kafka geht. Der Producer erhält die Daten per Bluetooth aus dem Sensorgürtel und sendet diese in das Kafka Cluster an das entsprechende Topic im Millisekunden Takt. Der Producer braucht dazu nur zusätzlich den Namen des Topics zu wissen. Als einfachstes Schema könnte als Topic Name die jeweilige Probanden ID genommen werden, von dem die Sensordaten im Augenblick geliefert werden, da jede Probanden ID eindeutig ist. Die Daten die vom Producer gesendet werden, würden im Millisekunden Takt sämtliche Werte des Gürtel streamen. Die Werte bestehen aus mehreren Sensoren wie des Accelerometers, Gyrometers, Magnetometers, Thermometers zusätzlich kann auch der Akkustand

[1] http://storm.apache.org/releases/2.0.0-SNAPSHOT/storm-kafka.html

Datenreihen-Tabelle

Zeitpunkt	Acc-X [m/s^2]	Acc-Y [m/s^2]	Acc-Z [m/s^2]	Acc-Tmp [°C]	Gyro-X [rad/s]	Gyro-Y [rad/s]	Gyro-Z [rad/s]	Mag-X [µT]	Mag-Y [µT]	Mag-Z [µT]	Pressure-P [hPa]	Pressure-T [°C]	Power-Voltage [V]
2017-10-27 08:25:32.580...	0,358	-8,468	-2,723	-0,500	-0,330	-0,638	-0,130	9,884	66,648	4,066	1018,550	23,525	4,170
2017-10-27 08:25:32.590...	0,130	-8,092	-2,192	-0,500	-0,570	-0,628	-0,091	10,276	66,990	4,457	1018,550	23,525	4,170
2017-10-27 08:25:32.600...	-0,096	-8,694	-2,568	-0,500	-0,843	-0,596	-0,136	9,884	66,648	4,430	1018,550	23,525	4,170
2017-10-27 08:25:32.610...	-0,094	-9,297	-3,473	-0,500	-0,852	-0,675	-0,165	9,509	66,990	4,781	1018,550	23,525	4,170
2017-10-27 08:25:32.620...	0,057	-9,297	-4,001	-0,500	-0,898	-0,587	-0,183	9,500	66,648	5,139	1018,550	23,525	4,170
2017-10-27 08:25:32.630...	0,131	-8,619	-3,848	-0,500	-0,917	-0,396	-0,153	9,500	66,648	5,868	1018,550	23,525	4,170
2017-10-27 08:25:32.640...	0,433	-8,318	-3,399	-0,500	-0,940	-0,308	-0,115	9,500	66,648	6,232	1018,550	23,525	4,170
2017-10-27 08:25:32.650...	0,207	-8,770	-3,624	-0,500	-1,028	-0,336	-0,122	9,500	66,648	6,596	1018,550	23,525	4,170
2017-10-27 08:25:32.660...	-0,322	-9,221	-4,296	-0,500	-1,032	-0,313	-0,177	9,500	66,648	7,325	1018,550	23,525	4,170
2017-10-27 08:25:32.670...	-0,474	-8,694	-4,591	-0,500	-0,968	-0,210	-0,189	9,108	66,306	7,298	1018,550	23,525	4,170
2017-10-27 08:25:32.680...	-0,097	-7,941	-4,292	-0,500	-0,955	-0,094	-0,192	9,108	66,306	8,391	1018,550	23,525	4,170
2017-10-27 08:25:32.690...	0,130	-7,791	-4,594	-0,500	-1,070	-0,080	-0,226	9,108	66,306	9,120	1018,550	23,525	4,170
2017-10-27 08:25:32.700...	-0,097	-7,941	-4,742	-0,500	-1,292	-0,034	-0,238	8,725	66,306	9,829	1018,550	23,525	4,170
2017-10-27 08:25:32.710...	-0,021	-8,243	-4,670	-0,500	-1,491	0,156	-0,266	8,725	66,306	10,922	1018,550	23,525	4,170
2017-10-27 08:25:32.720...	0,359	-8,920	-4,603	-0,500	-1,625	0,324	-0,303	8,332	65,964	11,624	1018,550	23,525	4,170
2017-10-27 08:25:32.730...	0,436	-9,523	-4,007	-0,500	-1,717	0,296	-0,291	8,332	65,964	11,988	1018,420	23,525	4,170

Abbildung 6.1: Auszug der Sensordaten

des Gürtel abgefragt werden. Die Abbildung 6.1[2] zeigt den Auszug einer Tabelle aus der MAMKS Software in denen beispielhafte Gürteldaten aufgezeigt sind. So könnte der Inhalt der Daten aussehen, die der Kafka Producer an das Cluster bzw. das Topic streamt. Dabei entspricht jeder Tabelleneintrag, einem Eintrag im Topic einer Partition. Diese werden als Key-Value Paare mit einer eindeutigen Offset hinterlegt. Unser KafkaSpout würde sich dann einen Topic abonnieren und die Daten gestreamt bekommen die er beim letzten Mal aufgehört hat. Möglich wäre es natürlich auch den Stream an jeder anderen beliebigen Stelle zu lesen. Sollte der KafkaSpout zu ersten Mal das Topic abbonieren, würde man die Daten vom ersten Offset an erhalten. Der KafkaSpout leitet anschließend die Daten, welche er abonniert hat, an die nachfolgenden Bolts weiter. Nach erster Überlegung kann der Umfang an Bolt aus mindestens drei bestehen. Der erste Bolt würde zum Puffern der Daten benutzt werden. Je nach Parameter des Klassifikationsalgorithmuses, kann die Puffergröße, von Millisekunden bis Minuten, unterschiedlich sein. Dazu würde der PufferBolt einen Parameter erhalten mit dem es möglich wäre, den Pufferzeitraum beliebig anzupassen. Ein weiterer Bolt würde den Klassifikationsalgorithmus enthalten, der die gepufferten Daten erhält und diese als Funktion ausführt. Der Output Klassifikationsalgorithmus-Bolts, bildet dann gleichzeitig den

[2]eigener Screenshot aus der MAMKS Software

Parameter des nächstes Bolts und zwar des OutputBolts. Hierbei können bis zu drei verschiedene Bolts gewählt werden die vorher über den Split-Bolt weitergeleitet wurden wird. Entweder einzeln oder gleichzeitig könnten dadurch mehrere OutputBolts bedient werden die das Ergebnis direkt in einer Konsole ausgeben, oder eine XML Datei beschreiben in diesen die bisherigen Klassifikationsergebnisse aus festgehalten sind. Denkbar ist auch der Einsatz einer Datenbank in denen die Ergebnisse entsprechend gespeichert werden. An dieser Stelle würde sich auch der Einsatz einer Time-Series Datendatenbanken eignen. Time Series Datenbanken besitzen den Vorteil, dass bei Ihren Daten der Zeitstempel im Vordergrund steht und Daten in diese Datenbank geschrieben werden die oft nur angehängt und selten geändert werden. Da herkömmliche relationale und nicht relationale Datenbanken oft Probleme damit haben, da häufige Operationen an Performanz Problemen leiden, sollen die Time Series Databases Abhilfe schaffen. Als Mögliche Datenbanken käme da OpenTSDB in Frage. Seit der Version 1.1.0[3] bietet Storm auch Unterstützung durch Integration einer Schnittstelle, die für Bolts genutzt werden kann, um die Daten in einer OpenTSDB Datenbank zu schreiben. Die Skizze: Abbildung 6.2[4], soll exemplarisch den beschriebenen Lösungsvorschlag verdeutlichen, worin Gürtel eines Probanden Daten, über deine kabellose Verbindung an das Kafka Cluster sendet.

Um auch in der Skizze zu veranschaulichen welchen Bereich jeweils Kafka und Storm einnehmen, wurde der Bereich den Kafka einnimmt, gelb markiert und den Bereich von Storm in lila. Der KafkaSpout, lässt sich in der Mitte erkennen, wo beide Plattformen miteinander vereinigt sind bzw. zusammenarbeiten. Bei Bedarf kann diese Lösung auch entsprechend angepasst oder erweitert werden. Denkbar wäre auch das der Output-Bold durch eine völlig neue Topolgie oder an neues ein Kafka Cluster ergänzt wird um weitere Operationen an den Daten durchzuführen. Möglich ist auch die Option, dass der Producer gleichzeitig ein Consumer ist. Dadurch könnte

[3] https://github.com/apache/storm/tree/master/external/storm-opentsdb

[4] eigene Zeichnung von draw.io

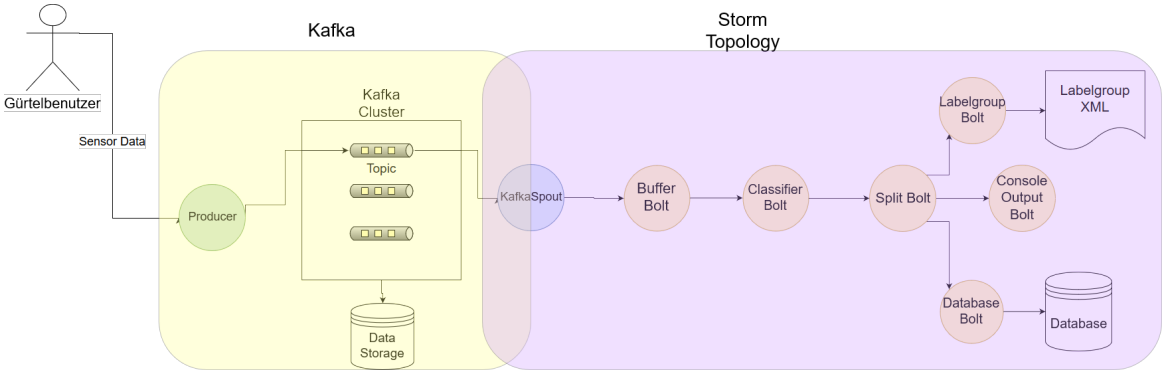


Abbildung 6.2: Visualisierung des Lösungsansatzes

in Form einer Anwendung die Möglichkeit geboten werden, den Stream zu starten, der zur selben Zeit den Output liefert, um den Benutzer auch Feedback darüber zu geben kann, welche Bewegung er gerade ausgeführt hat. Diese Anwendung könnte in Form einer klassischen Desktop Anwendung erfolgen. Allerdings liegt der Gedanke einer mobilen Lösung näher, um auch die Mobilität des Gürtelbenutzers nicht einzuschränken. Dies könnte in einer Erweiterungen unserer Activity Tracker App sein, die in dieser Projektgruppe implementiert wurde oder auch Stand-Alone App die auf deinem Tablet oder Smartphone läuft. Anhand dieser vielen Optionen sieht man welche Kombinationen diese Seminararbeit offenbart. Sollte es zu einer Implementierung dieses Systems kommen, bietet sich hiermit eine sehr mächtige Echtzeitdaten Verarbeitungsplattform an, welche einfach zu Erweitern wäre indem man die Größe des Clusters und der Topolgie nach den Bedürfnissen ändern könnte, dank Kafkas und Storms Skalierung.

7 Fazit und Ausblick

In den vorherigen Kapitel wurde ein Lösungssatz zum Realtime event processing an Sensordaten und Aktivitätserkennung vorgestellt. Dank der Apache Kafka und Storm Plattformen lassen sich in der Programmiersprache Java ein System zur Echtzeitdatenverarbeitung umsetzen. Dank ihrer Skalierungen lassen sich zum einen das Kafka Cluster und die Storm Topologie bei Bedarf erweitern oder Verkürzen ohne dabei Verluste in Daten oder Betriebszeit hinnehmen zu müssen. Damit dürfte es bei einer Umsetzung und Implementierung zu keinen großen Schwierigkeiten kommen. Aufgrund der bereits fortgeschrittenen Projektlaufzeit unserer derzeitigen Projektgruppe, die nur noch bis Ende März läuft, ist eine Realisierung dieses Lösungsansatzes als eher unwahrscheinlich anzusehen. Grund dafür wären die dafür nötige Zeit zur Umsetzung, die derzeit von uns vorrangig zum Refactoring bisheriger Implementierungen und Projektdokumentationen genötigt werden. Aufgrund der Projektziele hat diese Seminararbeit auch einen eher geringeren Nutzen für derzeitige Projektgruppe und bietet vielmehr den Ausblick eines Themas für die Zukunft. Eine weitere Überlegung kann auch eine Integration in die aktuelle MAMKS Software sein, wobei die Möglichkeiten zum Grad der Integration interessant sind. Denkbar wäre die Möglichkeit das Cluster und die Bolts über die MAMKS Software steuern zu können oder auch ein neues Modul zur Ausgabe der Echtzeit Verarbeitung. Diese Ideen könnten mit Rücksprache der Betreuern als Teil einer Abschlussarbeit angesehen werden, oder als Aufgabe der kommenden Projektgruppe, womit diese Seminararbeit gut als Einstieg in diese Thematik und Orientierung dienen kann.

B.9. Nachbearbeitung von Klassifikationsergebnissen

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Abteilung Assistenzsysteme und Medizintechnik

Projektgruppe Mobilitäts-Assessments
mit körpernahen Sensoren für zuhause
SoSe17 - WiSe17/18

Seminararbeit: Nachbearbeitung von Klassifikationsergebnissen

Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autor: Beatrice Coldewey

16. April 2018

Zusammenfassung

Im Kontext der Human Activity Recognition ermöglichen maschinelle Lernverfahren die Klassifizierung von Sensordaten. Eine Verbesserung der begrenzten Genauigkeit der im Zuge der Projektgruppe MAMKS und MAMKSFZ generierten Modelle soll durch eine Nachbearbeitung der Label erfolgen.
Die durchgeführte Literaturrecherche nach in diesem Kontext angewandten Verfahren gibt einen Überblick über einige Möglichkeiten zur Verbesserung der Klassifikationsergebnisse. Um entscheiden zu können, welches der Verfahren eine Verbesserung der Klassifikation ermöglichen könnte, wurden die Klassifikationen des mamks-Modells auf die auffälligsten Fehlklassifikationen hin untersucht. Abschließend werden Realisierungsmöglichkeiten wie ein Mehrheitsentscheid und regelbasierte Verfahren auf der Basis der aktuellen Arbeit der PGMAMKSFZ diskutiert.

Inhaltsverzeichnis

1 Einleitung

Die Projektgruppen „Mobilitäts-Assessments mit körpernahen Sensoren“ (PGMAMKS) und „Mobilitäts-Assessments mit körpernahen Sensoren für Zuhause“ (PGMAMKSFZ) haben unter Anderem zum Ziel, in den mittels Sensorgürtel aufgenommenen Bewegungsdaten einzelne Aktivitäten wie z. B. Gehen möglichst präzise zu erkennen. Zu diesem Zweck wird eine Auswahl der aufgenommenen Daten für die Anwendung von überwachten maschinellen Lernverfahren vorverarbeitet und mehrere Modelle mit verschiedenen Verfahren wie z. B. Convolutional Neuroal Network oder K-Nearest Neighbor mit verschiedenen Parametern auf den Daten antrainiert. Die Modelle können schließlich zur Klassifikation von Sensordaten verwendet werden, d. h. den Sensordaten werden automatisch Label mit Informationen bezüglich Aktivitäten wie z. B. Gehen, Hinsetzen oder Sitzen zugeordnet.

Die Evaluation der Ergebnisse erfolgt anhand von manuell erstellten Referenzlabeln. Die dabei ermittelte Klassifikationsgenauigkeit der Modelle liegt jedoch nicht bei 100 %. Neben den Bereichen, wo die Label richtig zugeordnet / nicht zugeordnet wurden (true positive / true negativ), gibt es auch Bereiche, denen nicht das richtige Label zugeordnet wurde (false positiv / false negativ). Eine präzise Erkennung ist jedoch von großer Bedeutung, da die Modelle im weiteren Verlauf unbekannte Datensätze klassifizieren sollen und die Label für weitere Auswertungen verwendet werden. Soll zukünftig z. B. analysiert werden, wie viele Stunden am Tag der Proband sitzt, ist dies nur sinnvoll möglich, wenn es sich bei den Bereichen, die durch das Modell als Sitzen gelabelt werden, auch wirklich um solche handelt. Ist eine Verbesserung der Klassifikationsgenauigkeit nicht mehr durch eine Anpassung der Trainingsdaten oder der maschinellen Lernverfahren möglich, kann eine Nachbearbeitung der Klassifikationsergebnisse durchgeführt werden.

Welche Fehler in den Datensätzen zu finden sind und wie diese ggf. durch eine Nachbearbeitung behoben werden können, wird nachfolgend diskutiert. Dazu werden in der Literatur genannte Verfahren recherchiert und die vorhandenen Datensätze bezüglich der größten Klassifikationsfehler untersucht. Abschließend werden Empfehlungen bezüglich der Anwendbarkeit verschiedener Verfahren auf den Anwendungsfall der Projektgruppe gegeben.

2 Related Work

Die Einsatzbereiche von maschinellen Lernverfahren sind ebenso vielfältig wie die Möglichkeiten bezüglich der Realisierung. Während in vielen Bereichen für den jeweiligen Verwendungszweck ein ausreichend gutes Ergebnis erreicht wird, werden die Klassifikationsergebnisse in einigen Fällen noch nachbearbeitet. Die in der Literatur angewandten und im Zuge der Projektgruppe interessanten Möglichkeiten einer Nachbearbeitung werden nachfolgend dargestellt.

Bereits im Ausblick der Abschlussdokumentation der PGMAMSK ist die Idee einer Nachbearbeitung der Klassifikationsergebnissen zufinden. Als gegebenes Beispiel soll überprüft werden, ob ein bestimmtes Label im Bezug auf seinen Kontext auch wirklich sinnvoll ist. So ist beispielsweise die Aktivität Springen mit einer Dauer von 0,1 Sekunden ebenso wie kurze Stehen-Abschnitte ohne Aufstehen und Hinsetzen während des Sitzens als unplausibel einzustufen [1].
Ebenso wurden bei der Anforderungserhebung in den Interviews der PGMAMKSFZ bereits Themen wie die Optimierung von Klassifikationsergebnissen mittels merged machine learning / Mehrheitsentscheidungen oder der Überprüfung der Sinnfrage bezüglich der Label mittels Einbau von Regeln angesprochen.

Bei der weiteren Recherche wurde Literatur nach Kombinationen aus Stichworten wie „postprocessing“, „improve classification output“, „improve precision“, „review“, „(Human) Activity Recognition“, „machine learning“, „sensor data“ und „accelerometer“ durchsucht.

Viele der gefundenen Textstücke mit dem thematischen Schwerpunkt auf der Aktivitäts-

erkennung bearbeiten lediglich verschiedene Erkennungs- und maschinelle Lernverfahren und ihre Performance im Vergleich zu anderen. Ebenso konnten unzählige Studien zur Verbesserung der einzelnen Lernverfahren selbst gefunden werden. In einer Vielzahl der gefundenen Literatur wird der Begriff Postprocessing auch im Zusammenhang mit der Verarbeitung und Vorbereitung der Sensorrohdaten für z. B. das maschinelle Lernen verwendet. Während in einigen Artikeln die Nachbearbeitung der Klassifikationsergebnisse lediglich als Möglichkeit zur Verbesserung ohne konkrete Realisierungsangaben genannt wurde, konnten einige Verfahren identifiziert werden, mit denen in Studien eine Verbesserung erzielt werden konnte:

Im Bereich der Klassifikation von Sensordaten mittels maschineller Lernverfahren ist das am häufigsten genannte Problem das Auftreten von kurzzeitigen, fehlerhaften Labeln, die z. B. aufgrund von Sensorrauschen in den verschiedenen Aktivitätsbereichen auftauchen. Da über die in den Daten enthaltenen Informationen in den beschriebenen Fällen bekannt ist, dass diese für eine gewisse Zeit andauern, ist es möglich, die Klassifikationsergebnisse um diese kurzzeitigen Fehlinterpretationen zu bereinigen und die Daten zu glätten. In [16] wird beispielsweise im Zuge der Klassifikation der Körperhaltungen Sitzen, Stehen und Gehen davon ausgegangen, dass die Betroffenen in einer Haltung für eine gewisse Zeit verweilen. Befinden sich in den Ergebnissen Haltungen, die nur sehr kurz andauern, werden diese mit der am häufigsten in der Zeiteinheit verwendeten Haltung ersetzt. In [9] konnten betroffene Frames ebenfalls mittels Mehrheitsentscheid über eine vordefinierte Nachbarschaft von zwei Sekunden (Optimum) korrigiert werden. Wurde keine Mehrheit erreicht, verblieb das Label unverändert. In [4] wurde mittels Mehrheitsentscheid über 0,25 Sekunden eine Verbesserung von 2-4 % erreicht. Für ein Toolkit zur Gestenerkennung wird in [6, 8] die Implementierung eines Label-Filters beschrieben. Dieser ermöglicht, dass Label, die in ihrer Dauer unter einer definierten Grenze liegen, durch ein anderes Label ersetzt werden, wenn dieses zuvor ausreichend lang erkannt wurde. Ist letzteres nicht der Fall, erhällt der Bereich das „default null rejection class label“. In [10] erfolgt die Glättung des Ergebnisses durch die Einschränkung, dass Übergänge in eine

andere Transition nur erlaubt sind, wenn die Wahrscheinlichkeit für dies eine minimale Dauer überschreitet. Die genannte Problematik der kurzzeitigen Fehlklassifikationen lässt sich auch in anderen Anwendungsbereichen finden. Bei der Flugbahnklassifizierung unterliegen die verschiedenen Modie ebenfalls einer Mindestdauer, sie dürfen durch den verwendeten Medianfilter über N benachbarte Sample aber auch nicht übersehen werden.[5]

Bezüglich der Verwendung von Hintergrundwissen für regelbasierte Verfahren zur Überprüfung der Sinnfrage einzelner Label und Labelabfolgen konnten keine übertragbaren Verfahren gefunden werden. Lediglich im Zuge der weiteren Auswertung der Klassifikationsergebnisse kommen beispielsweise in [15] Regeln zum Einsatz. Dabei werden EKG-Signale, welche mit Herzschlag-Typen gelabelt sind und Accelerometer-Daten mit Information über Aktivitäten dahingehend mittels bestimmter Regeln überprüft, ob es sich um einen echten Alarm bei einer erkannten Arrhythmie handelt oder ob sich ein Fehlalarm aus der aktuellen Aktivität ableiten lässt.
Eine Integration von Hintergrundwissen bezüglich des Zeitverhaltens einzelner Aktivitäten ist ggf. noch in [8, 7] erkennbar. In dem entwickeltem Toolkit zur Gestenerkennung wurde ein Timeout-Filter implementiert. Dieser verhindert, dass Gesten, von denen angenommen wird, dass sie innerhalb eines bestimmten Zeitabschnittes nicht mehrfach vorkommen, aufgrund der Art der Datenverarbeitung nicht mehrfach erkannt werden. Alternativ wird in [19] und in [12] Kontextwissen bereits beim Training und der Klassifikation integriert. Ersterer verwendet die ausgewählten Features oder Rohdaten der benachbarten N Frames zusätzlich zu denen des aktuellen Frames. Zweiterer beschreibt die Verwendung der Information über die letzte Aktivität als Feature für den aktuellen Frame.
Eine weitere Alternative zur Integration von Kontextwissen aber auch zur Glättung der Klassifikationsergebnisse soll mittels Hidden Markov Modellen möglich sein. Die in Abbildung 2.1 dargestellten Ergebnisse wurden in [11] mittels Convolutional Neural Networks, der Berechnung eines Hidden Markov Modell und der Reklassifizierung mittels

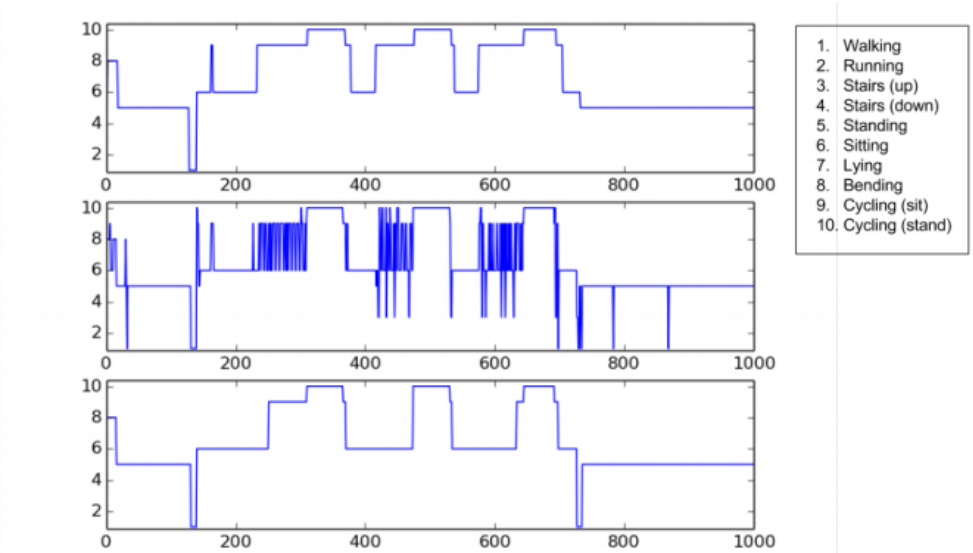


Abbildung 2.1: Klassifikationsverbesserung durch Nachbearbeitung mittels Vertibi: Der obere Graph repräsentiert die realen Aktivitäten, der Mittlere zeigt die Ergebnisse des CNN und der untere die des Viterbi. [11]

Viterbi-Algorithmus erzeugt. Dafür wird in einem Zwischenschritt die Wahrscheinlichkeit, zwischen bestimmten Aktivitäten zu wechseln, berechnet. In [13] wurde Aufbauend auf einen Decision Stumps Classifier ein Hidden Markov Modell auf der Basis der Ausgabewahrscheinlichkeiten des Klassifizierers antrainiert. In [20] wurde eine Verbesserung der Aktivitätserkennung von Google mittels Markov smoother implementiert.

Während zahlreiche Studien die Ergebnisse verschiedener maschineller Lernverfahren und Parametrisierungen vergleichen, um für den jeweiligen Anwendungsfall den besten Klassifikators zu ermitteln, ist ein in der Literatur häufig genannter, vielversprechender Bereich Ensemble Methoden. Bei dieser Methode werden mehrere Modelle (Basismodelle) generiert und ihre Kassifikationsergebnisse mit geeigneten Methoden zu einem Ergebnis kombiniert. Durch die Kombination von Modellen, die in ihren Fehlklassifikationen möglichst unterschiedlich sind, können die Defizite der Einzelnen kompensiert

werden und besserer Klassifikationsergebnisse erreicht werden. Neben statistischen Gründen lassen sich auch Vorteile in Bezug auf großen oder kleinen Datensets, komplexeren Problemen oder heterogenen Datenquellen finden. Ein Nachteil liegt in dem höheren Rechen- und Zeitaufwand. [14]
In [16] wurde dieses Verfahren beispielsweise auf Entscheidungsbäume, Bayes-Klassifikatoren und Backpropagation Neueronaler Netze erfolgreich angewendet. In [17] wurden Kombinationen aus den Klassifikatoren J48, Multi-layer Perceptron und Logistic Regression getestet.
Die Literatur bietet ebenfalls zahlreiche theoretische Grundlagen zu diesem Themengebiet. Die Methoden beziehen sich zum einen auf das Anlernen bzw. die Auswahl möglichst unterschiedlicher Modelle. Zum Anderen werden geeignete Methoden zur Kombination der Klassifikationsergebnisse beschrieben. Einige der bekanntesten Methoden lassen sich unter Stichworten wie „Majority Voting“, „Weighted Voting“, „Bagging“, „Boosting“ und „AdaBoost“, „Stacking“, „Random Forests“, „Stacking“, „Random Subspace Method“, „Mixture of Experts“, „Error Correcting Output Codes“ sowie „Bayesian Voting“ weiter recherchieren. [14, 3, 18]

Die beschriebenen Verfahren für eine Nachbearbeitung variieren je nach verwendeten Daten, Lernverfahren, ausgewählten Labelgruppen oder Zweck der Klassifikation. Einige setzen direkt bei den Trainingsergebnissen an, andere beziehen sich bereits auf den späteren Verwendungszweck der klassifizierten Daten.
Werden Entscheidungsbäume auf kleinen oder verrauschten Datensets antrainiert, kann durch Post-Pruning der Baum wieder verkleinert und zu kleine Unterteilungen wieder verallgemeinert werden, um eine bessere Klassifizierung zu erreichen. Ähnlich ermöglicht Rule-Truncation bei regelbasierten Ansätzen die Auswahl von möglichst allgemeingültigen Regeln und filtert zu spezielle heraus.
Erfolgt keine Anpassung der Klassifikatoren, werden ihre Ergebnisse ggf. durch die zuvor beschriebenen Verfahren oder ähnliche aufbereitet. Bei anderen Ansätzen, z. B. aus der Bild- und Videoverarbeitung zur Objekterkennung kann beispielsweise Non-Maximum-

Suppression eingesetzt werden, um sich überlappende Bereiche, die das selbe Objekt erkennen, zu einem zusammenzufassen.
Die Erhebung und Verarbeitung der Daten folgt in der Regel neben dem reinem Forschungsvorhaben einem speziellem Ziel, so dass die generierten Ergebnisse ggf. auch direkt unter Einbezug von weiterem Hintergrundwissen für den Endnutzer ausgewertet und visualisiert werden.[2]
In [11] wird im Zuge der Datenauswertung beispielsweise noch vorgeschlagen, die mit Gehen gelabelten Bereiche unter 3 Sekunden als „schlurfen“ (shuffling) zu labeln, um bei der weiteren Auswertung die Bereiche besser vom „richtigen“ Gehen trennen zu können. Auf die Möglichkeiten der Auswertung der Daten wird in dieser Ausarbeitung jedoch nicht weiter eingegangen.

3 Analyse häufiger Fehler in den Klassifikationsergebnissen

Im Zuge der Senior-Home-Studie der PGMAMSKFZ wurden 29 Datensätze aufgezeichnet und mit Referenzlabeln (SeniorHomeLabel) versehen. Die Auswertung der mittels des von der PGMAMKS generierten mamks-Modell erstellten Klassifikationen dieser Daten zeigt eine Genauigkeit von lediglich 51,08 % (siehe Abbildung 3.1).

	STATIC				TRANSITION		DYNAMIC		
	0	1	2	3	4	5	6	7	8
STATIC									
LIE_ON_BACK 0	246076	2917	11001	2668	1589	1687	0	0	0
LIE_ON_SIDE 1	2056	44411	405	837	1102	475	0	20	23
SIT 2	23698	623	7774326	953827	31214	34893	402	2401	1843
STAND 3	1150	63378	173563	4459250	25295	206033	3908	44331	196502
TRANSITION									
SIT_STAND 4	106	0	2431	1874	20782	4220	0	760	583
STAND_SIT 5	340	80	1674	2806	2340	25478	84	1923	430
DYNAMIC									
JUMP 6	0	0	0	0	0	0	274	0	0
STAIR_CLIMB 7	0	0	0	810	2	827	643	46978	10900
WALK 8	98	452	1768	125517	7210	57746	1752	74035	1840828

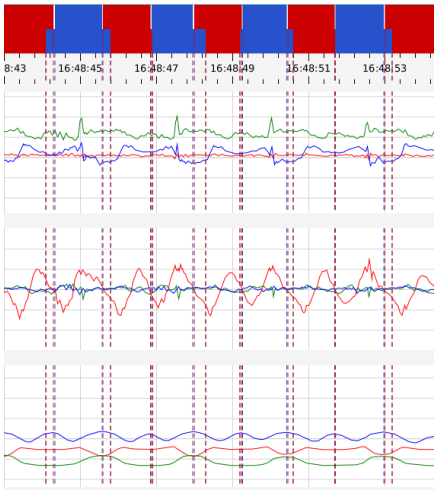
	Recall	Precision	Accuracy	F1-score
STATIC				
LIE_ON_BACK	0.9253	0.8997	0.9971	0.9123
LIE_ON_SIDE	0.9003	0.3970	0.9956	0.5510
SIT	0.8811	0.9760	0.9251	0.9262
STAND	0.8620	0.8038	0.8911	0.8319
TRANSITION				
SIT_STAND	0.6757	0.2321	0.9952	0.3455
STAND_SIT	0.7247	0.0769	0.9809	0.1390
DYNAMIC				
JUMP	1.0000	0.0388	0.9996	0.0747
STAIR_CLIMB	0.7809	0.2756	0.9917	0.4074
WALK	0.8727	0.8975	0.9711	0.8849
Average	0.8470	0.5108	0.9719	0.5637

Abbildung 3.1: Konfusionsmatrix aus dem Vergleich der SeniorHomeLabel mit den mittels mamks-Modell generierten Klassifikation sowie daraus berechnete Werte für Recall, Precision, Accuracy und F1-Score

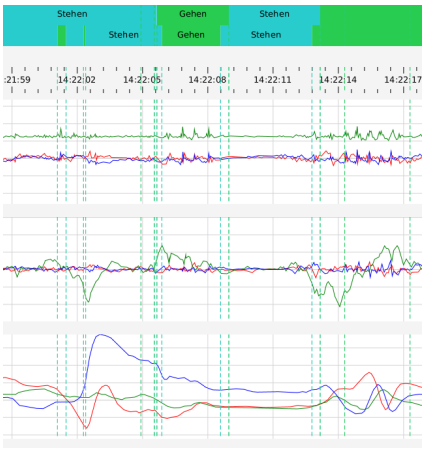
Neben der Analyse der „Zahlen“ aus der Konfusionsmatrix ermöglicht die Analyse der in der mamks-Software visualisierbaren Label einen guten Einblick in häufige Klassifikationsfehler. Die auffälligsten Probleme, die bei der Klassifikation durch das mamks-Modell auftreten, werden nachfolgend beschrieben. In den beispielhaft gezeigten Abbildungen ist in der oberen Labelreihe immer das Referenzlabel und in der unteren Reihe das Klassifikationsergebnis dargestellt.

Wie in Abbildung 3.2 zu sehen, gibt es häufig in den Randbereichen zweier Label Ungenauigkeiten bezüglich der genauen Position der Grenze zwischen den Aktivitäten. Auch wenn es sich in der Regel nur um wenige Sample handelt, summiert sich dieser Fehler besonders bei vielen kurzen Aktivitäten. Die Ursache für diesen Fehler liegt wahrscheinlich neben einer möglichen Fehlklassifikation auch in dem Verfahren zur Erstellung und Anwendung der Modelle. Zum einen erfolgt das Training der Modelle ebenso wie der Vergleich anhand von evtl. ebenfalls nicht 100 % richtig von Menschen manuell erstellten Referenzlabeln. Des Weiteren wird bei der Klassifikation nicht jedes Sample einzeln gelabelt, sondern je nach Verfahren über einen bestimmten Radius z. B. 0,2 Sekunden ein Label vergeben.

Zusätzlich sind durch das gewählte manuelle Labelverfahren auch in wenigen Fällen Fehler in den Referenzlabeln zu finden. Diese sind nicht nur in den zuvor gezeigten Randbereichen möglich, sondern auch in anderen Aktivitätsbereichen. Hier lassen sich ggf. sogar korrektere Label in den Klassifikationen des Modells finden, die in der Auswertung als Fehlklassifikation gewertet werden (vgl. Abbildung 3.3).

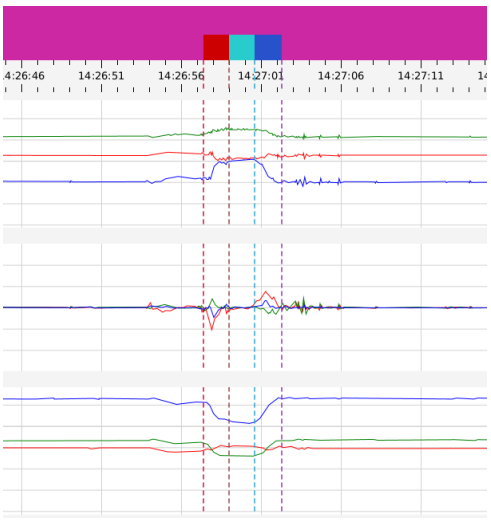


(a) Abfolge Hinsetzen - Aufstehen im Zuge des 5-Time Chair Rise Tests

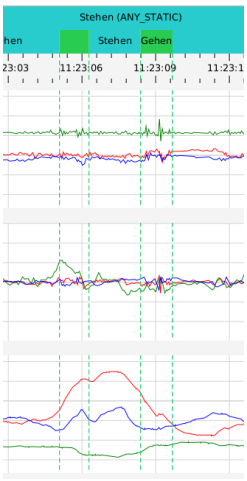


(b) Abfolge von Aktivitäten Stehen und Gehen im häuslichen Umfeld

Abbildung 3.2: Labelbeispiele für die Problematik von ungenauen Randbereichen



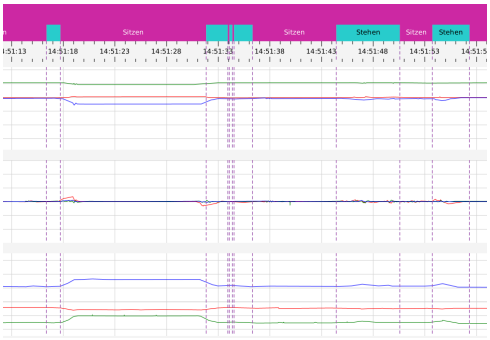
(a) Fehlende Aufstehen-Stehen-Hinsetzen-Folge im Referenzlabel



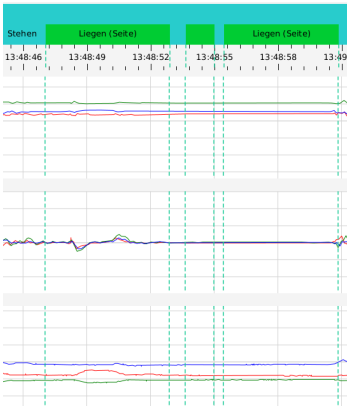
(b) Ungenaue Referenzlabel bezüglich Gehen bzw. Bewegung im Stehenabschnitt

Abbildung 3.3: Labelbeispiele für die Problematik von fehlerhaften oder ungenauen Referenzlabeln

Ein häufig in den Daten zu sehendes Problem ist die Unterscheidung der statischen Aktivitäten Liegen, Sitzen und Stehen untereinander (vgl. Abbildung 3.4). Für die angelernten Modelle ist die Unterscheidung aufgrund der geringen Unterschiede in den jeweiligen Sensordaten der Bewegungen besonders schwierig. Analog zu den statischen Aktivitäten lassen sich auch Fehlklassifikationen im Bereich der dynamischen Aktivitäten finden. Hier ist die Unterscheidung der Aktivitäten Gehen und Treppensteigen nicht immer gegeben.



(a) Fehlerhafte Erkennung von Stehen im Sitzen-Bereich



(b) Fehlerhafte Erkennung von Liegen im Stehen-Bereich

Abbildung 3.4: Labelbeispiele für die Problematik der Unterscheidung der statischen Aktivitäten Liegen, Sitzen und Stehen

Einer Vielzahl der fehlerhaft erkannten Bereiche lässt sich zumindest ohne eine detailliertere Analyse der zugehörigen Sensordaten und Aktivitäten keine genaue Ursache zuordnen (vgl. Abbildung 3.5). Besonders bei den im häuslichen Umfeld aufgenommenen Daten handelt es sich nicht um Bewegungsdaten, die die einzelnen Aktivitäten „optimal“ abbilden. Die natürlichen Bewegungen sind im Gegensatz zu den Assessments stärker verrauscht mit Artefakten anderen Bewegungen, die nicht explizit gelabelt werden können.

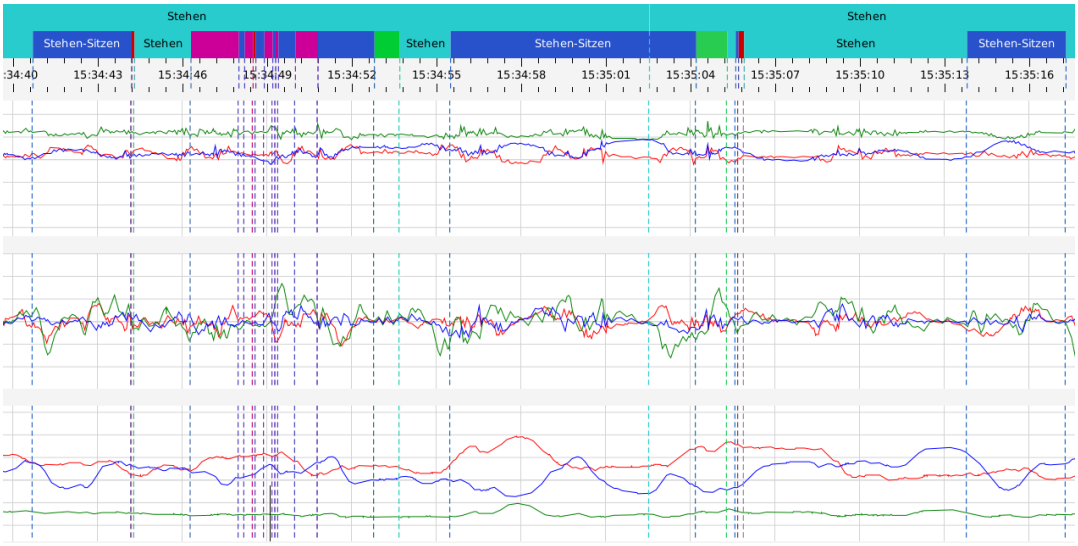
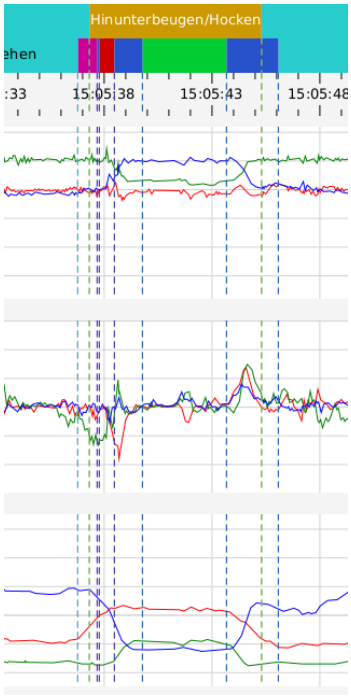


Abbildung 3.5: Labelbeispiel

Ein Weiterer auffälliger Aspekt, der in den visualisierten Daten sichtbar ist, sich aber nicht auf die Evaluationsergebnisse auswirkt, sind die ungelabelten oder speziell als unbekannt gelabelten Bereiche. Bei diesen Bereichen handelt es sich zum Teil um Daten, über die keine konkreten Referenzinformationen vorliegen, es gibt aber auch Abschnitte wie z. B. das Anlegen des Sensorgürtels, die keinem der ausgewählten Label zugeordnet werden können. Auch Aktivitäten wie das Hocken, welches nicht durch das mamks-Modell abgedeckt wird oder das Fahrradfahren gehören dazu. Durch das mamks-Modell wird diesen Bereichen fälschlicher Weise eines der zur Verfügung stehenden, bekannten Label zugeordnet (vgl. Abbildung 3.6)



(a) Fehlerhafte Klassifikation des Hockens durch andere Label

(b) Fehlerhafte Klassifikation des Fahrradfahrens durch andere Label

Abbildung 3.6: Labelbeispiele für die Problematik der Klassifikation von Bereichen, die nicht durch die bekannten Label abgedeckt werden

4 Integration in MAMKSFZ

In der Ausarbeitung wurden in Kapitel 2 verschiedene Methoden zu Nachbearbeitung von Klassifikationsergebnissen, die in der Literatur Anwendung finden, dargestellt. Auf der Basis der in Kapitel 3 beschriebenen Klassifikationsfehler, die sich aktuell in den mittels mamks-Modell generierten Labeldateien finden lassen, werden nachfolgend Möglichkeiten zur Korrektur dieser abgeleitet und diskutiert.

Das zu Beginn genannte Problem ungenauer Randbereiche ebenso wie die fehlerhaften Referenzlabel lassen sich teilweise durch genaues Setzen der Referenzlabel vermeiden. Dies ist besonders zu beachten, da die manuell erzeugten Label, die nicht genau oder fehlerhaft sind, auch für das Training der Modelle verwendet werden und bereits dort Probleme verursachen. Ob eine Anpassung der verwendetet Fenstergrößen eine Verbesserung der Erkennung an den Labelgrenzen ermöglicht, ist zu überprüfen.

Im Zuge der Arbeit der PGMAMKSZF werden zur Zeit mittels den maschinellen Lernverfahren Convolutional Neural Networks, Bagging Trees, Boosted Decision Trees, Quadratische Diskriminanzanalyse und K-Nearest Neighbor Modelle trainiert und für die Klassifikation der Senior-Home-Daten angewendet. Da aus diesen bereits unterschiedliche Labeldateien generiert werden können, ist eine einfach zu realisierende Verbesserungsmöglichkeit die Implementierung eines Mehrheitsentscheids. Da die unterschiedlichen Modelle bereits vorhanden sind, muss nur ein geeignetes Verfahren zur Kombination dieser ausgewählt werden. Die einfachste Methode wäre ein einfaches Voting über die einzelnen Label und die Auswahl des Labels, dass am häufigsten vorkommt. Eine Gewichtung einzelner Modelle entspechend ihrer Güte wäre ebenfalls möglich. Spezielle Verfahren wie das Stacking sind ggf. nochmal im Detail zu recherchieren. In der An-

wendung ist zu testen, welche Label bzw. welche der Modelle möglichst unterschiedliche Fehler erzeugen und daher zusammen die besten Ergebnisse liefern. In der finalen Klassifikation könnten so einzelne Fehlklassifikationen herausgefiltert und Aktivitäten besser erkannt werden.
Im Zuge der Arbeit der PGMAMKSFZ wurde bereits diskutiert, dass Lösungen gesucht werden, welche es ermöglichen, Datenbereiche ohne Label zu versehen. Dies ist besonders für Aktivitäten relevant, die durch die Modelle nicht abgedeckt werden. Da bei unbekannten Sensordaten zwar ein Label vergeben werden muss, die Modelle aber ggf. auch unterschiedlich entscheiden, ist es beispielsweise möglich durch setzen einer Grenze für das Voting, also wie viele der Modelle sich einig seien müssen, bevor das Label auch übernommen wird. Herrscht keine Einigkeit, wird kein Label vergeben bzw. der Bereich als Unbekannt markiert um Fehler zu vermeiden. Bei der Auswahl der Grenze ist zu beachten, dass zwar möglichst viele Fehlklassifikationen herausgefiltert werden, aber auch nicht zu viele Bereiche als Unknown markiert werden, die bei einer niedrigeren Grenze noch richtig erkannt worden wären.
Das in der Literatur genannte Glätten der Klassifikationsergebnisse könnte ebenfalls mittels Mehrheitsentscheid, der auf einer Labeldatei über eine bestimmte Fenstergröße ausgeführt wird, realisiert werden.

Die Anwendung regelbasierter Nachtbearbeitungsschritte erfordert eine detaillierte Betrachtung der Fehlklassifikationen und Aktivitäten in den verschiedenen Datensätzen. Zum einen scheinen Aktivitätsabfolgen wie Sitzen - Aufstehen - Stehen mit Aufstehen als einzig logischer Übergang einfach zu definieren, in der durchgeführten Senior-Home-Studie konnten aber auch Abfolgen wie Sitzen - Hocken - Stehen beobachtet werden. Ist die Definition des benötigten Hintergrundwissens kein Problem, ist es aufgrund der Erkennungsgenauigkeit dennoch nicht einfach entscheidbar, ob es sich z. B. bei der Abfolge Sitzen - Stehen - Sitzen um eine fehlerhafte Erkennung des Stehens handelt oder die Transitionen nicht erkannt wurden. Werden für eine spätere Auswertung beispielsweise nur logisch korrekte Abschnitte gesucht, könnten solche aber mittels Regeln für

die Weiterverarbeitung herausfiltert werden.
Ein anderer einfacherer Anwendungszweck für Regeln könnte auch hier das Glätten der Klassifikationsergebnisse sein. In der nachfolgend gezeigten XML-Struktur ist ein beispielhafter Entwurf einer Regel dargestellt. In dieser können die betroffenen Aktivitäten, anzuwendende Bedindungen, die dieses erfüllen soll, und Aktivitäten, die folgen, sollte es zu einem Verstoß kommen, definiert werden.

```
<rule>
        <ruleID>001</ruleID>
        <activity>BEND_OVER_OR_SQUAT</activity>
        <conditions>
                <minDuration>20</minDuration>
                <maxDuration></maxDuration>
                <validPreviousActivity>
                        <activity></activity>
                </validPreviousActivity>
                <validFollowingActivity>
                        <activity></activity>
                </validFollowingActivity>
        </conditions>
        <action>
                <newActivity>UNKNOWN</newActivity>
        </action>
        <comment></comment>
</rule>
```

Bezüglich der Glättung könnten so z. B. für die verschiedenen Label minimale Längen in Samplen definiert werden und betroffene Abschnitte durch Unknown ersetzt werden. Auch das Ersetzen von Labeln, z. B. fehlerhaftes Stehen oder Unknown in einem Sitzen-Abschnitt kann realisiert werden. Die aufgestellten Regeln sind jedoch auf ihr korrekte Wirkung hin, besonders im Hinblick auf andere Fehlklassifikationen, zu überprüfen. Es sollten im Verhältnis keine neuen bzw. mehr Fehler produziert werden. Die Eliminierung von false-positiv-Fehlern ist dabei in der Regel einfacher als die Zuweisung neuer Label

bei den false-negativ-Fehlern.

Die Verwendung von Hidden Markov Modelle scheint anhand der gefundenen Literatur ein ebenfalls vielversprechendes Verfahren zu sein. Eine Einschätzung, wie und ob dieses Verfahren auf den Anwendungsfall der PGMAMKSFZ übertragbar ist, erfordert weitere Recherche.

5 Fazit

Im Zuge der Arbeit der Projektgruppen MAMKS und MAMKSFZ wurden unter Anwendung maschineller Lernverfahren Modelle generiert, die die Klassifizierung von Sensordaten bezüglich Aktivitäten wie „Sitzen“, „Gehen“ und „Aufstehen“ ermöglichen. Zur Verbesserung der begrenzten Klassifikationsgenauigkeit der Modelle konnten in der Literatur verwendete Nachbearbeitungsmöglichkeiten wie Glättung von kurzfristigen Fehlklassifikationen oder Ensemble Methoden identifiziert werden.
Bei der Analyse der Ergebnisse der Klassifikation mittels mamks-Modell konnten verschiedene Fehlerursachen identifiziert werden. Neben Fehlern in den Referenzlabeln gehören zu diesen beispielsweise Probleme bei der Unterscheidung von unterschiedlichen statischen oder dynamischen Aktivitäten aber auch das Sensorrauschen durch die natürlichen Bewegungen im häuslichen Umfeld. Für die spätere Anwendung sind aber auch speziell die ungelabelten Bereiche mit unbekannten Aktivitäten problematisch.
Für die Anwendung in der PGMAMKSFZ sind besonders der Mehrheitsentscheid und einfache regelbasierte Ansätze interessant. Dabei soll es zusätzlich möglich sein, bei hoher Unsicherheit oder identifizierten fehlerhaften Labeln, diese mit dem Label Unknown zu klassifizieren. Beim Einsatz dieser Verfahren ist genau zu überprüfen, ob diese die vorhandenen Fehler beheben können, die bereits korrekt zugeordneten Bereiche nicht zu sehr beeinträchtigen oder bestimmte Label im Vergleich zu anderen benachteiligen. Ebenso ist auf den späteren Verwendungszweck der Daten zu achten. Soll beispielsweise allgemein der Anteil bestimmter Aktivitäten wie z. B. Gehen innerhalb eines Tages betrachtet werden, dürfen nicht zu viele Bereiche mit Unbekannt gelabelt werden. Werden hingegen ganz bestimmte Datenbereiche gesucht, können diese durch z. B. den Einsatz entsprechender Parameter oder Regeln für die Weiterverarbeitung herausgefiltert oder besonders markiert werden.

5 Fazit

Wie die genannten Verfahren je nach Parametrisierung die Klassifikationsgenauigkeit verbessern können, ist im Zuge der Arbeit der PGMAMKSFZ mit den neu erstellten finalen Modellen zu testen. Insgesamt sollte darauf geachtet werden, dass die Verfahren sich auch auf andere Datensätze übertragen lassen und kein Overfitting an dem Datensatz stattfindet.

Literatur

[1] Kjel Barjenbruch u. a. *Abschlussbericht Projektgruppe Mobilitäts-Assessments mit körpernahen Sensoren*. Mai 2017.

[2] Ivan Bruha. „Pre- and Post-processing in Machine Learning and Data Mining“. In: *Machine Learning and Its Applications: Advanced Lectures*. Hrsg. von Georgios Paliouras, Vangelis Karkaletsis und Constantine D. Spyropoulos. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, S. 258–266. ISBN: 978-3-540-44673-6. DOI: 10.1007/3-540-44673-7_13.

[3] Thomas G. Dietterich. „Ensemble Methods in Machine Learning“. In: *Proceedings of the First International Workshop on Multiple Classifier Systems*. MCS '00. London, UK, UK: Springer-Verlag, 2000, S. 1–15. ISBN: 3-540-67704-6.

[4] Jordan Frank, Shie Mannor und Doina Precup. „Activity and Gait Recognition with Time-delay Embeddings“. In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. AAAI'10. Atlanta, Georgia: AAAI Press, 2010, S. 1581–1586.

[5] J. Garcia u. a. „Trajectory classification based on machine-learning techniques over tracking data“. In: *2006 9th International Conference on Information Fusion*. Juli 2006, S. 1–8. DOI: 10.1109/ICIF.2006.301629.

[6] Nicholas Gillian. *class_label_filter*. Jan. 2018. URL: https://github.com/nickgillian/grt/wiki/class_label_filter.

[7] Nicholas Gillian. *class_label_timeout_filter*. Jan. 2018. URL: https://github.com/nickgillian/grt/wiki/class_label_timeout_filter.

[8] Nicholas Gillian und Joseph A. Paradiso. „The Gesture Recognition Toolkit“. In: *Journal of Machine Learning Research* 15 (2014), S. 3483–3487.

[9] Rene Grzeszick u. a. „Deep Neural Network Based Human Activity Recognition for the Order Picking Process“. In: *Proceedings of the 4th International Workshop on Sensor-based Activity Recognition and Interaction*. iWOAR '17. Rostock, Germany: ACM, 2017, 14:1–14:6. ISBN: 978-1-4503-5223-9. DOI: 10.1145/3134230.3134231.

[10] Chang Woo Han, Shin Jae Kang und Soo Kim. „Implementation of HMM-Based Human Activity Recognition Using Single Triaxial Accelerometer“. In: 93-A (Juli 2010), S. 1379–1383.

[11] Astrid Johnsen Tessem Hans-Olav Hessen. „Human Activity Recognition With Two Body-Worn Accelerometer Sensors“. Magisterarb. Norwegian University of Science und Technology, Juni 2016.

[12] Narayanan C. Krishnan und Diane J. Cook. „Activity Recognition on Streaming Sensor Data“. In: *Pervasive Mob. Comput.* 10 (Feb. 2014), S. 138–154. ISSN: 1574-1192. DOI: 10.1016/j.pmcj.2012.07.003.

[13] Jonathan Lester u. a. „A Hybrid Discriminative/Generative Approach for Modeling Human Activities.“ In: *IJCAI International Joint Conference on Artificial Intelligence*. Jan. 2005, S. 766–772.

[14] Robert A. Meyers. „Encyclopedia of Complexity and Systems Science - V. 1-10“. In: 1st. Springer Publishing Company, Incorporated, 2009. Kap. 315. ISBN: 9780387758886.

[15] Tanatorn Tanantong, Ekawit Nantajeewarawat und Surapa Thiemjarus. „False Alarm Reduction in BSN-Based Cardiac Monitoring Using Signal Quality and Activity Type Information“. In: *Sensors* 15.2 (2015), S. 3952–3974. ISSN: 1424-8220. DOI: 10.3390/s150203952.

[16] J. Wang u. a. „Wearable sensor based human posture recognition“. In: *2016 IEEE International Conference on Big Data (Big Data)*. Dez. 2016, S. 3432–3438. DOI: 10.1109/BigData.2016.7841004.

[17] M. N. S. Zainudin u. a. „Activity recognition based on accelerometer sensor using combinational classifiers“. In: *2015 IEEE Conference on Open Systems (ICOS)*. Aug. 2015, S. 68–73. DOI: 10.1109/ICOS.2015.7377280.

[18] Cha Zhang und Yunqian Ma. *Ensemble Machine Learning: Methods and Applications*. Springer Publishing Company, Incorporated, 2012. ISBN: 9781441993250.

[19] L. Zhang, X. Wu und D. Luo. „Improving activity recognition with context information“. In: *2015 IEEE International Conference on Mechatronics and Automation (ICMA)*. Aug. 2015, S. 1241–1246. DOI: 10.1109/ICMA.2015.7237663.

[20] M. Zhong u. a. „Advancing Android activity recognition service with Markov smoother“. In: *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. März 2015, S. 38–43. DOI: 10.1109/PERCOMW.2015.7133990.

C. Eigenständigkeitserklärung

Hiermit versichern alle Autoren dieser Ausarbeitung, dass sie diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichern sie, dass sie die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt haben.