CARL
VON
OSSIETZKY
universität OLDENBURG

FAKULTÄT II – INFORMATIK, WIRTSCHAFTS- UND RECHTSWISSENSCHAFTEN
DEPARTMENT FÜR INFORMATIK

Abstract Flexibility Description for Virtual Power Plant Scheduling

Von der Fakultät für Informatik, Wirtschafts- und Rechtswissenschaften
der Carl von Ossietzky Universität Oldenburg
zur Erlangung des Grades und Titels eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

angenommene Dissertation von

Judith Fröhling, geb. Neugebauer, M.Sc.

geboren am 02.12.1988 in Sassenberg

Gutachter:
Prof. Dr. Michael Sonnenschein
Prof. Dr. Oliver Kramer
Associate Professor Dr. Wei Lee Woon

Tag der Disputation: 24.04.2017

This document was created with a slightly modified version of the Latex dissertation template[1] by Christian Hinrichs.

---

[1] available at https://github.com/ambimanus/thesis-template, version 27.08.2015

Zusammenfassung

Im Zuge der Energiewende und dem damit verbundenen Wandel des Energiemarktes werden große zentrale Kraftwerke mehr und mehr durch viele kleine dezentrale Energieanlagen ersetzt. In diesem Rahmen spielen Virtuelle Kraftwerke (VK) eine entscheidende Rolle. VK bündeln die Kapazitäten der kleinen dezentralen Anlagen, sodass die VK wie herkömmliche Kraftwerke am Markt agieren können. Die Planung des VK Betriebs basiert auf den Flexibilitiäten der steuerbaren VK Teilnehmer, wie beispielsweise Blockheizkraftwerke, Wärmepumpen oder Batterien. Je nach Aufgaben des VK und der Art der VK Teilnehmer, müssen bei der VK Planung unterschiedliche einzelne Anlagen und Verbünde von Anlagen berücksichtigt werden. Mathematisch gesehen ist die VK Planung ein beschränktes Optimierungsproblem. VK Einsatzplanungsalgorithmen nutzen die Flexibilitäten steuerbarer Teilnehmer. Zur Nutzung dieser Flexibilitäten werden Flexibilitätsbeschreibungen verwendet. Verwandte Flexibilitätsbeschreibungen sind jedoch nur für einige Arten von VK Teilnehmern geeignet. Deshalb wird eine neue Flexibilitätsbeschreibung benötigt, die für unterschiedliche Arten einzelner Anlagen und Verbünde von Anlagen geeignet ist.

Das Ziel dieser Arbeit ist die Entwicklung einer abstrakten, einheitlichen und genauen Flexibilitätsbeschreibung für die VK Planung. Aus diesem Grund habe ich das Kaskaden-Klassifikations-Modell als Flexibilitätsbeschreibung entwickelt. Dieses Kaskaden-Klassifikations-Modell ist ein anpassbarer Klassifikator und eine abstrakte Flexibilitätsbeschreibung, die alle alternativ realisierbaren Anlagenfahrpläne beschreibt. Anschließend wird das Kaskaden-Klassifikations-Modell experimentell evaluiert und die Erfüllung der Flexibilitätsbeschreibungs Anforderungen analysiert. Abschließend wird die Anwendbarkeit des Kaskaden-Klassifikations-Modells in der VK Planung evaluiert.

Abstract

In the ongoing paradigm shift of the energy market from big power plants to more and more small and decentralized power plants, virtual power plants (VPPs) play an important role. VPPs bundle the capacities of the small and decentralized resources (DER), so that VPPs can participate in the energy market like conventional power plants. Planing of VPP operation, that is also called scheduling, relies on the flexibilities of controllable DER in the VPP, e.g., combined heat and power plants (CHPs), heat pumps and batteries. As far as VPPs perform different tasks in the energy system and consist of different participants, VPP scheduling can either mean planing the operation of single energy units or planing the operation of coalitions of energy units as a unit. Mathematically VPP scheduling tasks are constrained optimization problems. VPP scheduling algorithm use the flexibilities of the controllable VPP participants. Respective related flexibility description approaches are only successful for some VPP participants. Thus a new flexibility description for all controllable VPP participants including different kinds of single energy units and also coalitions of energy units is required.

The aim of this thesis is the development of an abstract, consistent and precise flexibility description for VPP scheduling. Therefore I have developed the cascade classification model. The cascade classification model is an adaptable classifier and an abstract description of flexibilities in terms of alternatively realizable operation schedules. After that the cascade classification model is evaluated experimentally and the fulfillment of the flexibility description requirements is analyzed. Finally the applicability of the cascade classification model as a flexibility description in VPP scheduling tasks is analyzed.

CONTENTS

Abbreviations XIII

Symbols XV

List of Figures XVII

List of Tables XXI

1 Introduction 1
- 1.1 VPP . . . 2
- 1.2 Flexibility . . . 5
- 1.3 Scheduling . . . 9
  - 1.3.1 Optimization in Scheduling Tasks . . . 12
- 1.4 Constraint Optimization Approaches for VPP Scheduling . . . 14
  - 1.4.1 VPP Scheduling Approaches . . . 16
  - 1.4.2 Search Space Modeling in COHDA . . . 18
  - 1.4.3 Search Space Modeling in AVPP Scheduling . . . 20
  - 1.4.4 Search Space Encoding Comparison . . . 23
- 1.5 Objective . . . 28
  - 1.5.1 Structure of this Thesis . . . 30

2 Methods 31
- 2.1 Classification . . . 31
- 2.2 Classifier Choice . . . 33
- 2.3 Classification Categories Related to the Identified Classification Task and Data Properties . . . 35
  - 2.3.1 Time Series Classification . . . 35
  - 2.3.2 High-dimensional Learning . . . 37

2.3.3 Imbalanced Learning 39
2.3.4 High-dimensional and Imbalanced Learning 43
2.3.5 One-class Learning 44
2.3.6 Result of the Literature Review 45
2.4 Classifiers Applied in this Thesis 46
2.5 Classifier Performance Evaluation 48
2.5.1 Precision 48
2.5.2 Parameter Sensitivity 51
2.5.3 Temporal Complexity 54

3 Classifier Development for Flexibility Descriptions 55
3.1 Problem Formulation and Classification Model Development 55
3.2 Basic Cascade Classification Model 58
3.3 Adaptation of the Basic Cascade Classifier to Given Classification Tasks 59
3.3.1 Classification Task Splitting (Projection): 2d -> pd 60
3.3.2 Baseline Classifier Choice 61
3.3.3 Result Aggregation 63
3.4 Data Preprocessing 63
3.4.1 Selection of Feasible Examples 65
3.4.2 Generation of Artificial Infeasible Examples 67
3.5 Generalized Cascade Classification Model 69
3.5.1 Transformation-Based Ensembles 72
3.5.2 Generalized Cascade Classification Model 73
3.6 Summary 76

4 Evaluation of the Cascade Classification Model 79
4.1 Basics of the Evaluation Experiments 79
4.1.1 Data 80
4.1.2 Evaluation Scenarios 85
4.1.3 Experimental setup 86
4.1.4 Baseline Classifiers and their Default Parametrization 88
4.1.5 Experimental Evaluation 88
4.1.6 Implementation 90

4.2 Precision and Model Fitting 92
4.2.1 Scalability of the basic cascade classification model 92
4.2.2 Baseline Classifier Choice 93
4.2.3 Dimensionality of the Low-Dimensional Data Subsets 102
4.2.4 Data Distribution Improvement due to Data Preprocessing . . 108
4.2.5 Ensemble Preparation: Transformations and Data Preprocessing 114
4.2.6 Ensemble Precision 118
4.2.7 Classifier Selectivity 121
4.2.8 Comparative Experimental Study 123
4.2.9 Summary 128
4.3 Parameter Sensitivity and Model Parametrization 131
4.3.1 Baseline Classifier Parametrization 131
4.3.2 Data Preprocessing Parametrization 135
4.3.3 Summary 139
4.4 Runtime Complexity 141
4.4.1 Runtime Complexity of the Basic Cascade Classification Model 141
4.4.2 Runtime Experiments 145
4.4.3 Summary and Runtime Improvement Suggestions 148
4.5 Discussion of the Evaluation Results 150

5 Application of the Cascade Classification Model as Flexibility Description in VPP Scheduling Tasks 151
5.1 Decoder Approach 153
5.1.1 Algorithm for a Basic Decoder Approach 153
5.1.2 Experimental Study with the new Decoder Approach 156
5.2 Remarks on the Application of the Developed Flexibility Description . 162

6 Final Review 165
6.1 Summary and Conclusion 165
6.1.1 Evaluation Results of the Cascade Classification Model 166
6.1.2 Applicability Results of the Developed Flexibility Description in VPP Scheduling Tasks 168
6.1.3 Conclusion 169
6.2 Perspective 169

A Energy Data sets of single energy units 171
A.1 Micro Combined Heat and Power Plant 171
A.2 Heat Pump 175

B Artificial Data Sets 179
B.1 *Hypersphere* Task 179
B.2 *Hyperbanana* Task 182

C Energy data sets of coalitions of energy units 187
C.1 Generation of Feasible Global Operation Schedules 187
C.1.1 Step 1: Sample Generation 188
C.1.2 Step 2: Example Shifting 193
C.1.3 Properties of the Generated Global Feasible Operation Schedules 196
C.2 Generation of Infeasible Global Operation Schedules 203
C.2.1 Sample Generation 203
C.2.2 Feasibility Estimation 205

Bibliography 211

ABBREVIATIONS

Energy System

AVPP	Autonomous Virtual Power Plant
CHP	combined heat and power plant respectively micro combined heat and power plant (μCHP)
COHDA	Combinatorial Optimization Heuristic for Distributed Agents
DER	distributed energy resource
HP	heat pump
VPP	Virtual Power Plant

Time Series

acf	autocorrelation function
AR	autoregressive process
fft	Fourier transformation
pacf	partial autocorrelation function
pca	principal component analysis

Classification

CM	confusion matrix
kmeans	kmeans data description
knn	k nearest neighbors
MOG	mixture of Gaussian models
OCC	one-class classification
OCSVM	one-class support vector machine
SVDD	support vector data description
SVM	support vector machine
TN	true negative rate
tn	true negatives
TP	true positive rate
tp	true positives

Further Abbreviations

HB	*Hyperbanana* data set
HS	*Hypersphere* data set
SA	sensitivity analysis

SYMBOLS

Energy Context

S_{g}	global operation schedule
S	operation schedule of a single energy unit

Basic Cascade Classification Model

dim	dimensionality of the classification task
d	dimensionality of the classification task
p	dimensionality of the low-dimensional subsets
c_i	p-dimensional classifier of a basic cascade classification model C
C	cascade of low-dimensional classifiers c_i with $i \in \{1, \ldots, dim - (p-1)\}$

Model Adaptations and Data Preprocessing

ϵ	minimal distance between feasible nearest neighbors
ϵ_b	minimal distance between artificial infeasible examples and their nearest feasible neighbors
$\mathcal{N}(\mu, \sigma)$	normal distribution with mean μ and standard deviation σ
S	quantity of selected feasible examples
Γ	quantity of generated artificial infeasible examples

sn	number of selected feasible examples

Generalized Cascade Classification Model

S_1	aggregation scheme 1
S_2	aggregation scheme 2
w	weighting factor for the aggregation of the intermediate ensemble results

Baseline Classifier Parameters

N	number of training examples
k	knn paramter
ν, γ	OCSVM parameters
C, γ	SVM and SVDD parameters
k_t, ϵ	MOG parameters
k, ϵ_t	kmeans parameters

LIST OF FIGURES

1.1 Decoder mapping 20

2.1 Explicable variance 39
2.2 Multi-task learning example. 40
2.3 Imbalance Categories. 41
2.4 Confusion matrix. 49

3.1 Correlation of μCHP operation schedules 57
3.2 Low-dimensional data structure 58
3.3 Low-dimensional data structure II 61
3.4 Multi-task learning example 62
3.5 Selection of feasible examples 66
3.6 Generation of infeasible examples 67
3.7 Unfavorable μCHP coalition data structure 70
3.8 Unfavorable heat pump data structure 70
3.9 Aggregation schemes for the generalized cascade classification model 75
3.10 Process model: cascade classification model 77

4.1 Scalability 93
4.2 Multi-task learning on B-HS 94
4.3 Decision boundaries of different baseline classifiers on A-CHP 95
4.4 TP and TN values on A-CHP 95
4.5 Decision boundaries of different baseline classifiers on A-HS 96
4.6 TP and TN values of different baseline classifiers on A-HS 97
4.7 min-max classifier for the μCHP data set 98
4.8 Decision boundaries of different baseline classifiers on B-HS 100
4.9 TP and TN values of different baseline classifiers on B-CHP 100
4.10 Decision boundaries of different baseline classifiers on B-HS 101

4.11 TP and TN values of different baseline classifiers on B-HS 102
4.12 Feasible μCHP concepts 103
4.13 Correlation of single μCHP time series 104
4.14 Overall correlation of μCHP time series 104
4.15 Overall correlation of *Hypersphere* time series 105
4.16 Projection error example 106
4.17 TP and TN values for varying values of p on A-CHP 106
4.18 TP and TN values for varying values of p on B-CHP 107
4.19 Decision boundaries and TP values on A-CHP with data preprocessing 110
4.20 Decision boundaries on A-HB1 with data preprocessing 111
4.21 TP and TN values on A-HB1 with data preprocessing 112
4.22 Example of wrong artificial infeasible examples 113
4.23 Classifier precision with wrong artificial infeasible examples 113
4.24 Data structure of μCHP coalition 114
4.25 Data structure of transformed μCHP coalition 115
4.26 TP values for different transformations of C-HP 117
4.27 TP values for different transformations of C-CHP 117
4.28 TP and TN values of C-HP ensembles 119
4.29 TP values of C-CHP ensembles 120
4.30 Classifier selectivity on C-CHP 122
4.31 Classifier selectivity on A-HB2 123
4.32 Classifier Comparison on B-CHP 125
4.33 Classifier Comparison on C-CHP ensembles 126
4.34 Comparison of different OCSVM kernels 132
4.35 Baseline classifier parametrization comparison 133
4.36 Baseline classifier parametrization comparison II 134
4.37 Functional relation between sn and ϵ 136
4.38 Distance parameter combination regions 138
4.39 Distance parameter combination regions II 139
4.40 Runtime dependence on N 146
4.41 Runtime dependence on N II 147
4.42 Runtime dependence on p 147
4.43 Runtime dependence on p II 148

5.1 Functioning of a basic decoder approach 155

A.1 Data structure of B-CHP 173
A.2 Correlation of single time series of B-CHP 173
A.3 Overall correlation of B-CHP 174
A.4 Correlation of single time series of C-HP 176
A.5 Overall correlation of C-HP 177
A.6 Data structure of C-HP 177
A.7 Data structure of transformations of C-HP 178

B.1 Data structure of B-HS 180
B.2 Correlation of single time series of A-HS 181
B.3 Overall correlation of A-HS 181
B.4 Data structure of A-HB1 184
B.5 Correlation of single time series of A-HB1 184
B.6 Overall correlation of A-HB1 185

C.1 Data structure of μCHP coalitions 191
C.2 Data structure of improved μCHP coalitions 192
C.3 Data structure of μCHP coalitions with shifting 195
C.4 Data structure of C-CHP 198
C.5 Data structure of coalitions of μCHPs and heat pumps 199
C.6 Overall correlation of C-CHP 200
C.7 Overall correlation of coalitions of μCHPs and heat pumps 201
C.8 Data structure of transformations of C-CHP 202
C.9 Distance distribution of operation schedule approximations 208
C.10 Distance distribution of operation schedule approximations II 209
C.11 Overall power production of C-CHP 210

LIST OF TABLES

1.1 Comparison of related flexibility description approaches 24

2.1 Classification task properties. 34
2.2 Data set properties. 34

3.1 Complete classification task properties 59
3.2 Cascade classification model parameters. 76

4.1 μCHP data set properties 81
4.2 Heat pump data set properties 82
4.3 μCHP coalition data set properties 82
4.4 *Hypersphere* data set properties 83
4.5 *Hyperbanana* data set properties 84
4.6 Scenarios A 85
4.7 Scenarios B 86
4.8 Scenarios C 86
4.9 Parameter sets 89
4.10 Classifier parameters for the comparative experimental study (B-CHP)124

5.1 Errors in new decoder examples. 158

C.1 Optimized x values for μCHP coalitions 196
C.2 Properties of feasible and infeasible global operation schedules. 207

1 INTRODUCTION

In the ongoing paradigm shift of the energy market traditional big power plants are more and more replaced by smaller and decentralized power plants. A successful market integration of small and decentralized power plants including renewable energies is still a current research issue. This PhD thesis was drawn up in the context of the graduated program *Systemintegration Erneuerbarer Energien (SEE)* [1] in the field of operational management. In operational management virtual power plants (VPPs) play an important role and therefore VPP operation has been considered in several projects, amongst others *SmartNord*, see Sonnenschein and Hofmann 2015. The research question of this thesis results from *SmartNord*. In *SmartNord* the operation of VPPs was studied. While planing of VPPs with single energy units as members succeeded, planing of VPPs with energy unit coalitions as members remains an unsolved problem. The aim of this thesis is to solve this problem.

In this introductory chapter the problem of VPP planing with coalitions as VPP members is presented in the context of VPP operation in more detail. First of all the application context is presented. Therefore VPPs, their application and their operation, especially scheduling are introduced in general in Sect. 1.1. Since VPP scheduling relies on flexibilities, flexibilities are presented in Sect. 1.2. After that VPP scheduling is presented in detail in Sect. 1.3. The term VPP scheduling means on the one hand scheduling of single energy units and on the other hand scheduling of coalitions of energy units. With regard to these two meanings of VPP scheduling two currently much discussed VPP scheduling approaches are presented and discussed in Sect. 1.4.1. Based on these results a research need is identified and the research question is formulated in Sect. 1.5. Finally an overview of the complete thesis is given in Sect. 1.5.1.

---

[1] http://www.uni-oldenburg.de/en/see/,last accessed on 24.02.2017

1.1 VPP

The idea of VPPs is to bundle decentralized capacities of small energy units. These bundled capacities are comparable to traditional power plants in terms of the amount of power production and controllability, see Bitsch et al. 2002. VPPs attract interest either from a technical and a commercial point of view and this can be retrieved in VPP definitions. Often employed VPP definitions are the one from the *FENIX* project, see Schmid 2009 by Pudjianto et al. 2007,

> A Virtual Power Plant is a flexible representation of a portfolio of DER (distributed energy resources). A VPP not only aggregates the capacity of many diverse DER, it also creates a single operating profile from a composite of the parameters characterizing each DER and incorporates spatial (i.e. network) constraints into its description of the capabilities of the portfolio.

and another definition given by Asmus 2010,

> VPPs rely upon software systems to remotely and automatically dispatch and optimize generation or demandside or storage resources in a single, secure Web-connected system.

Further definitions can be found in Braun and Strauss 2008; Saboori et al. 2011 and general overviews on VPP are e.g. Pudjianto et al. 2007; Asmus 2010; Nikonowicz and Milewski 2012; Plancke et al. 2015. Recently an enhanced VPP definition was proposed in Plancke et al. 2015, based on the above presented definitions. The definition takes different aspects of the integration of DER into the power system into consideration.

> A portfolio of DERs, which are connected by a control system based on information and communication technology (ICT). The VPP acts as a single visible entity in the power system, is always grid-tied and can be either static or dynamic.

This definition points out three essential and characteristic components of a VPP, its participants, the control system and its role in the energy system.

- The first component addresses the **participants of a VPP**. Each VPP consists of a portfolio of decentralized energy resources (DER) that are operated as a unified and flexible resource on the energy market. DER comprise distributed generation e.g., photovoltaic (PV) modules, combined heat and power plants (CHPs) and wind turbines, controllable consumers e.g., heat pumps (HPs) and cold stores and prosumers e.g., batteries or electric vehicles. Based on the type of DER, in the portfolio, VPPS are called supply-side VPPS (only generation), demand-response VPPs (flexible loads and storage units) and mixed asset type VPPs (all types). Furthermore VPPs can be static or dynamic coalitions. While common static coalitions are used for different tasks, dynamic coalitions are formed task based, e.g., product based, see Nieße, Lehnhoff, et al. 2012; Beer 2013. VPP based on dynamic coalitions are called dynamic VPPs (DVPPs).

- The second component concerns the **VPP control system**. VPPs are coalitions in terms of information and communication technologies (ICT), see moreover e.g., Tröschel 2010; Asmus 2010. VPP control systems can be classified according to the communication within the VPP or the organization of the control structure or both of these two aspects. Such a VPP control system classification based on the communication structure distinguishes between three classes of different organizational structures, see Nikonowicz and Milewski 2012. The three classes are centralized-, distributed- and fully distributed control structures. Other classification schemes focusing on both control system aspects (communication and organization), distinguish between more classes, see e.g., Hinrichs 2014, 27 et seqq. or Nieße 2015, 47 et seqq. In Hinrichs 2014 the control system classes: central, decentralized, hierarchical, distributed and fully distributed are distinguished. While VPPs with a central control system are operated by one unit with global knowledge, VPPs with a fully distributed control system coordinate themselves without a central unit via communication among the VPP participants. All other control systems are hybrid forms of these two control systems types.

- The third component refers to the **roles of VPPs in the energy system**. VPP services rely on the VPP inherent flexibility and comprise grid services, energy services and further services. VPPs providing grid services are called technical VPPs (TVPPs). Grid services comprise ancillary services, see Blank 2015, that help to "... *maintain the integrity and stability of the transmission system as well as power quality...*", Electricity Industry EURELECTRIC 2014. VPPs providing energy services are called commercial VPPs (CVPPs). CVPPS can participate, depending on their size and the VPP portfolio composition, in portfolio management, wholesale market trading and contract optimization tasks. Portfolio management refers to the ability of a VPP to balance either its own portfolio of DER and VPPs can also be part of larger portfolios and balance theses larger portfolios with their flexibility. Wholesale market trading takes place at the forward market months or even years in advance, at the day-ahead market one day before product delivery and at the intra-day market hours before delivery. Another CVPP operation form is the contract optimization, where VPP operation is optimized according to a contract between the VPP operator and the electricity supplier. Beside these grid services and energy services, future VPPs are supposed to provide services at the capacity and the flexibility market, see also Kouzelis 2015.

In this thesis CVPPs are considered in the context of wholesale market trading and portfolio management, but flexibility market applications are also possible. Even though VPPs depend on the grid, grid information is usually not available in market-based applications. Therefore grid restrictions are not considered in this thesis.

To accomplish the different roles in the energy system, VPPs have to be operated according to their role and their control system. In the following only the role of VPP scheduling is considered. VPP scheduling, consists of planing and controlling the VPP task. For commercial VPPs (CVPPS), considered in this thesis scheduling has two meanings. Scheduling can either mean planing single energy units of a VPP or planning a whole VPP as a unit for subordinate tasks.

Scheduling of single energy units according to a global scheduling objective is applied in wholesale market applications like e.g., the day-ahead market or the intra-day market at the energy spot market EPEX SPOT, see e.g., Hinrichs 2014, p 186.
This kind of scheduling is related to planing of energy products.

Scheduling of coalitions of energy units, respectively scheduling of whole VPPs as a unit is used to react to deviations from the initially chosen schedules, e.g.,

> caused by an energy unit failure. The consideration of energy unit coalitions, also called 'Holone', see Tröschel 2010 or 'AVPPs', see Anders, Siefert, et al. 2010 reduces the effort of correcting the initially chosen schedules.

All VPP scheduling tasks mainly rely on the VPP's inherent flexibility. The terms flexibility and scheduling are presented in detail in the following subsections Sect. 1.2 and Sect. 1.3.

1.2 FLEXIBILITY

The term flexibility is ubiquitous and often not clearly defined, see Golden and Powell 2000. Flexibility is defined generally as

> The quality of bending easily without braking.
>
> The ability to be easily modified.
>
> Willingness to change or compromise.

in the oxford dictionaries, Press 2016. Flexibility in organizational terms is defined as the 'the capacity to adapt across four 'dimensions', see Golden and Powell 2000. Flexibility can be achieved in temporal-, range-, intention- and focus dimension. This means flexibility can be measured in terms of the response time to an event, the degree of adaptability to events, the degree to which an offensive or defensive stance is taken towards flexibility and the possibility of flexibility attainment in the process.

In the energy system context flexibility has been defined from the modeling perspective as a unique, innate, state- and time dependent quality, see Petersen et al. 2013. This definition can be abbreviated as 'the ability to deviate from the plan'. Flexibility definitions are related to their application context as for instance electricity system flexibility is closely related to grid frequency, voltage control, delivery uncertainty, -variability and power ramping rates, see Lund et al. 2015.

Since flexibility is an essential component in the energy grid, several flexibility definitions, descriptions and measures have been proposed amongst other in the following research projects.

TotalFlex The project *TotalFlex*, see Skou 2015 dealt with the establishment of a flexible electricity grid. The concrete aim was the development of a cost-efficient market-based system that utilizes all available flexibility and considers grid constraints. *Flexibility is characterized in terms of time and/or energy amount* and flexibility is *described as flexibility objects*, see Valsomatzis et al. 2015. In energy market considerations flexibility is treated with respect to five properties that add or subtract value from the commodity flexibility, see Kouzelis 2015. The first property 'energy scaling' describes the ability to increase or decrease energy consumption within given limits. The second property 'energy shifting' quantifies operation time shifting of flexible devices. The geographical location of generation and consumption is the third property. The fourth and the fifth property are time of demand/supply activation and uncertainty resulting from demand and renewable generation.

Flexines The project *Flexines*, see Gemert 2012 aimed at the development of a set of electronic instruments to operate household devices with an intelligent energy management system. Flexibility is considered in terms of household devices. Flexibility of one household device is defined in Roossien 2012; MacDougall et al. 2013.

> *A device has flexibility if it is capable of shifting its production or consumption of energy in time within the boundaries of end-user comfort requirements and without changing its total energy production or consumption.*

Flexibility of the single household devices is divided into five types based on their amount of energy and their capacity. Corresponding mathematical flexibility *measures describe the amount of power that can be ramped up or down*, see Roossien 2012.

Single household devices employ to small capacities and energy demands compared to the requirements of the electricity grid. Therefore coordinated clusters of household devices, like VPPs aggregate the flexibilities of the household devices, see Roossien 2012; MacDougall et al. 2013; Lünsdorf 2012.

> *Flexibility of a coordinated cluster of devices or (virtual) power plant is a statistical interpretation of the shiftability of the devices. It is*

> *measured as the amount of power increase or decrease, with respect to its current power consumption or production, that can be sustained for a given period of time.*

The aggregated flexibility is characterized by its aggregated shiftability, the overall available power to ramp up or down. Flexibility of coordinated clusters is merely *measured as the sum of the flexibilities of all aggregated devices*, see Roossien 2012. The flexibility of each single device in the cluster is assumed to be independent of the flexibility of all other devices.

SmartNord The *SmartNord* project, see Sonnenschein and Hofmann 2015 focused on the provision of decentralized active power, operating reserve and reactive power in smart grids, including the design of new information and communication technology (ICT) structures. Flexibility is defined according to VPP operation. *The flexibility of an energy unit is the quantity of alternative operation schedules that can be realized within a given time interval*, see Bremer 2015. The flexibility of an energy unit is *described mathematically as a black box* that models all alternatively realizable operation schedules.

Beside the flexibility approaches in these projects a taxonomy for modeling flexibility of consumer VPPs was proposed in Petersen et al. 2013. Flexibility of a system is defined as a unique, innate, state- and time dependent quality. The flexibility taxonomy is based on power system characterizing constraints, the power capacity, the energy capacity, the energy level at a specific deadline and the minimum runtime. The resulting flexibility taxonomy consists of three class with increasing flexibility.

The above presented flexibility definitions, descriptions and measures are more or less related to VPPs. VPP operation has been considered in more detail in *Flexines* and *SmartNord*. While in the *Flexines* project VPPs of household devices have been considered to match supply and demand, in *SmartNord* VPP formation, planing and controlling operations have been considered.

Based on the flexibility definitions and descriptions from *Flexines* and *SmartNord* I have developed flexibility definitions for single VPP participants and coalitions of energy units or even whole VPPs. The flexibility definitions rely on the definition of an energy unit.

Definition 1.2.1 (**Energy unit**)
The term energy unit is used in this thesis for single units of distributed energy resources (DER). Energy units can produce power, consume power or store power and evolve it.

Definition 1.2.2 (**Coalition of energy unit**)
A coalition consists of arbitrary numbers of energy units of arbitrary types.

Definition 1.2.3 (**Flexibility of a single energy unit**)
The flexibility of a single energy unit (DER) is the possibility to influence its operation mode by shifting its production or consumption under given constraints.

This flexibility is described as a set of all alternatively realizable operation schedules (power production- or consumption time series, see Def. 1.3.2) of the considered energy unit.

Definition 1.2.4 (**Aggregated Flexibility of a coalition of energy units or even a whole VPP**)
The aggregated flexibility of a coalition of energy units, e.g., in a VPP is an aggregation of the flexibilities of all energy units in that coalition.

This aggregated flexibility is described as a set of all alternatively realizable global operation schedules (global power production- or consumption time series of a coalition of energy units, see Def. 1.3.3). The flexibilities of the devices in the coalition can be independent or dependent.

If the flexibilities of all energy units are independent, the aggregated flexibility is the set of all possible combinations of the realizable operation schedules of all energy units. But if the flexibilities of the single energy units depend on each other, only a part of these possible combinations of realizable operation schedules of all energy units belong to the set of aggregated flexibility.

The flexibility of single energy units results from the possibility of operation mode modifications. Controllable energy units like combined heat and power plants (CHPs), heat pumps (HPs) or batteries usually employ flexibility, while non controllable energy units like photovoltaic modules or most wind turbines employ no flexibility. The flexibilities in CHPs are due to thermal buffers which decouple the combined production of thermal and electrical energy from the thermal demand of a building. Heat pumps also gain flexibility from thermal buffers that decouple the thermal demand from the power consumption needed for the heat pump operation. Batteries offer flexibility to energy consumers and producers in a similar way as thermal buffers.

The flexibility, respectively the possibility to modify the operation mode of controllable energy units is limited by several restrictions, also called constraints. These constraints are mainly technical constraints, like ramping rates, switching frequencies or limitations of the thermal buffer. Furthermore economical constraints, like the fuel consumption of an energy unit or operator defined constraints might be relevant. These different constraints might have different priorities. A more detailed consideration of constraints can be found in the scheduling section, see Sect. 1.3.1.

The resulting flexibility of an energy unit or a coalition of energy units strongly depends on the type of energy units and the constraints resulting from the application context. Since flexibility plays a fundamental role in scheduling processes, descriptions of flexibilities are considered in the scheduling section for different scheduling algorithms, see Sect. 1.3.

1.3 SCHEDULING

Scheduling is defined generaly in Pinedo 2012[p. 1] as

> ...a decision-making process that is used on a regular basis in many manufacturing and services industries. It deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives.

This scheduling definition can be transfered to VPP scheduling by adapting the two main characteristics the allocation of resources to tasks in combination with an optimization of objectives. These two characteristics can be considered on two different levels, a general level of several tasks (energy products) and a more detailed level of a single task (energy product). On the level of all tasks, the given energy units are allocated to energy products. This allocation is done with portfolio optimization. Portfolio optimization is a mathematical optimization, mainly applied for investment decisions, see e.g., Markowitz 1952; Detemple 2014, but also applied to energy products, see e.g., M. Liu and F. F. Wu 2007; Nieße, Beer, et al. 2014.

In this thesis only the level of single tasks is considered. At this level the two characteristics can be again related to VPP scheduling actions.

- **Allocation of resources to tasks over given time periods** corresponds to an assignment of operation schedules to the energy units in a VPP for a given temporal horizon with a given temporal resolution.
- In terms of VPP scheduling, the **scheduling goal optimization of one or more objectives** means an optimization according to the compliance of an energy product. Compliance of energy unit constraints and further objectives might be e.g., minimization of operation costs or profit maximization. Further details concerning the optimization are given in Sect. 1.3.1.

In this thesis VPP scheduling is defined as follows.

> **Definition 1.3.1** (**VPP Scheduling**)
> VPP scheduling consists in the assignment of feasible operation schedules S to all energy units and feasible global operation schedules S_g to all coalitions, participating in the considered VPP. The assignment of schedules is restricted by the fulfillment of a given energy product and possible further goals. This assignment of operation schedules to energy units or coalitions of energy units is called scheduling task.

Operation schedules of single energy units and global operation schedules of coalitions of energy units are defined according to Hinrichs 2014, p 42, Nieße 2015, p 72 and Bremer and Sonnenschein 2012.

> **Definition 1.3.2** (**(Operation) schedule** of a single energy unit)
> An operation schedule S is a power production or consumption time series of a single energy unit for a defined temporal horizon with an equal temporal resolution.
> Operation schedules can be either realizable (feasible) or not realizable (infeasible).

A feasible operation schedule is one power production or -consumption time series, that corresponds to a realizable operation mode of the considered energy unit.

Respectively infeasible operation schedules represent time series of not realizable operation modes. If the energy unit employs flexibility there are several feasible operation schedules. But non controllable energy units without flexibility employ only one feasible operation schedule, that corresponds to the only realizable operation mode of that energy unit, see Sect. 1.2.

> **Definition 1.3.3** (**Global (operation) schedule** of a coalition of energy units)
> A global operation schedule S_g of a coalition of energy units, e.g., a VPP consists of aggregated operation schedules of all energy units participating in the coalition. Global operation schedules can be feasible or infeasible depending on the feasibility of the underlying operation schedules and dependencies between the energy units in the coalition. If all units in the coalition are operated independently from each other, except from the weather conditions, a global operation schedule is feasible if all underlying aggregated operation schedules were feasible.

In literature mainly two different kinds of scheduling are distinguished according to the point in time of the scheduling action, predictive scheduling and reactive scheduling.

- **Predictive scheduling** is used to plan future tasks, see Hinrichs 2014; Bremer 2015 and consists in the generation of optimal initial operation schedules for the energy units in a VPP, see Tröschel and Appelrath 2009.

- **Reactive scheduling** takes place during the execution of a plan, see e.g.,Tröschel 2010; Steghöfer et al. 2013; Nieße 2015. Reactive scheduling is used to correct deviations from the initial planning due to e.g., prediction uncertainties or energy unit failures. Reactive scheduling can be specified in greater detail according to the way and the time to react to an event, see Priore et al. 2014. According to the way of reaction, schedules can be either repaired or a complete rescheduling can be done. The time to react in rescheduling can be periodic, event-driven or a hybrid form. A special form of reactive scheduling, named continuous scheduling is applied in dynamic VPPs (DVPPs), see Nieße 2015, p. 9. Continuous scheduling aims at a continuous adaptation of the energy unit operation modes to changing conditions, especially to a divergent energy unit behavior.

1.3.1 Optimization in Scheduling Tasks

After the general introduction of VPP scheduling, the underlying optimization problem is considered in more detail in this subsection. An optimization task consists in the optimization of an objective function ϕ, see e.g., Gritzmann 2013; Kramer 2014. Optimization tasks are formulated as minimization or maximization tasks. All possible solutions x form a feasible region (search space) F. A solution $x \in F$ is the optimum x^* if $\phi(x^*) \leq \phi(x)$ for minimization problems or $\phi(x^*) \geq \phi(x)$ for maximization problems. In evolutionary optimization tasks the optimization problem is evaluated with a fitness function $f(x)$ instead of an objective functions $\phi(x)$.

VPP scheduling optimization tasks show some variations from a general optimization task. In VPP scheduling tasks operation schedules (parameters) of several energy units are optimized. Furthermore there can be several optimization objectives. Usually F is restricted by some constraints to a constraint search space. These constraints turn the optimization problem into a constrained optimization problem. The treatment of constraint optimization problems is considered below.

Constraint Problem

Optimization problems with constraints form two classes, see Rossi et al. 2006 and Bremer 2015, p 43, constraint satisfaction problems (CSP) and constraint optimization problems (COP). Constraint satisfaction problems aim at finding at least one solution that satisfies all constraints, while constraint optimization problems aim at finding the optimal solution of all valid solutions. Therefore a COP is a CSP with an additional fitness function. Constraint optimization problems that contain only constraints with local effects are called distributed COP (DCOP). Distributed constraint optimization problems (DCOPs) are solved according to one or more global objectives with regard to the constraints, see Chapman et al. 2011.

VPP scheduling is a COP, because the best solution is searched among the possible solutions. VPP scheduling tasks are constrained e.g., by the feasibility of operation schedules, as well as operation mode preferences of individual energy units. Furthermore COPs can be characterized by the number of variables affected by a constraint as the ’arity’ of a constraint, see Rossi et al. 2006, p 14. According to the arity, VPP scheduling constraints can be divided into the two groups, low-arity constraints and high-arity constraint, see Hinrichs 2014, p 27. Low-arity constraints are all constraints that affect only a few energy units, e.g., up to three units in Hinrichs 2014, p 27, while high-arity (global) constraints affect more energy units e.g., power grid restrictions. Since power grid restrictions are not considered in this thesis only low-arity constraints have to be handled.

Beside the distinction of the arity, constraints can be classified into hard and soft constraints, see Tröschel and Appelrath 2009; Bremer et al. 2010. Hard constraints influence the validity of a solution and have to be obeyed at any time. Hard constraints are usually technical constraints. Soft constraints describe properties that should be reached and therefore do not influence the validity of a solution. Soft constraints are additional optimization objectives, like e.g., user preferences or carbon dioxide emissions.

All constraints restrict the space of valid solutions. Most of the times constraints do not simplify the space of valid solutions, but in contrary they aggravate the optimization with an inconvenient geometry of the search space, see e.g. Bremer 2015, p 40. In literature different constraint-handling methods have been proposed for constraint optimization problems (COPs).

Constraint-handling

Constraint-handling depends on the arity of the constraints, see e.g., Hinrichs, Bremer, and Sonnenschein 2013. In general constraints can be treated either during the optimization (implicit treatment) or separate from the optimization (explicit treatment). Furthermore a mixture of implicit and explicit constraint-handling is possible. A classification of constraint handling methods for COPs was proposed in Kramer 2010 in the context of evolutionary algorithm. Constraint handling methods are categorized into the following five classes.

- **Penalty functions** treat the constraints separately from the optimization. They decrease the fitness of invalid solutions according to the deviation from valid ones.

- **Repair algorithms** replace or repair invalid solutions during the evaluation of new solutions.

- **Decoder functions** build up a relationship between the constrained solution space and an artificial and easier to handle solution space.

- **Feasibility preserving representations and operators** turn all solutions into feasible ones.
- **Multiobjective optimization techniques** are based on constraint separation. The separated constraints are treated as further optimization objectives.

The choice of appropriate constraint handling methods depends on the optimization problem and the given constraints. Beside the arity of constraints, convenient constraint-handling techniques for VPP scheduling depend on the control structure of the VPP. In the following subsection, Sect. 1.4 different optimization approaches with different constraint-handling methods for VPP scheduling are presented.

1.4 CONSTRAINT OPTIMIZATION APPROACHES FOR VPP SCHEDULING

In this section different VPP scheduling approaches are presented and compared with regard to search space modeling and constraint-handling. The simplest solution for VPP scheduling would be a transfer of unit commitment approaches from conventional power plant scheduling.

Unit commitment is a constraint optimization problem, that assigns operation schedules to the power plants. The optimization task is solved at an hourly basis, amongst others with the objectives to meet the load requirement and to maximize the profit, see e.g., Saravanan et al. 2013. Unit commitment approaches reach from simple conventional techniques like listing approaches or dynamic programming to advanced non conventional techniques like heuristics. Furthermore hybrid techniques of both approach types can be found.

But unit commitment shows a great problem concerning scalability, see Saravanan et al. 2013. Increasing numbers of energy units and the evaluation of infeasible solutions can be very time consuming. Beside the scalability problem which hampers a transfer of unit commitment approaches to VPP scheduling, there are some differences between the scheduling tasks of unit commitment and VPPs. The main difference consist in the kind of considered energy units (large power plants vs. DER), the planing horizon (hourly based vs. quarter-hourly based) and the control structure of the coalition of energy units (central vs. different control schemes).

Beside conventional unit commitment approaches, there are VPP scheduling approaches in literature. These approaches solve VPP scheduling tasks (constraint optimization problems) with different methods corresponding to the VPP control structure. Here only the two contrary control schemes: central control and fully distributed control are distinguished, see also Sect. 1.1. All remaining control schemes are summed up as intermediate control schemes.

- **Central control schemes** are often applied, because they are relatively simple and find the best solution. But due to their worse scalability they exhibit a high computational complexity for VPP applications with numerous DER. More or less central approaches are COBB, see Peña Landaburu 2006, Stigspace, see Li et al. 2008; Li et al. 2010 and further approaches can be found e.g., in Steck 2013.

Intermediate control schemes and fully distributed control systems overcome the drawback of scalability, see Hinrichs 2014, 5 et seq. and Faltings and Yokoo 2005. While central approaches fit the central control structures of previous commercial power plant management, distributed approaches fit the dynamical structures of numerous independent and distributed energy units, see Hinrichs 2014, 3 et seq.

- **Intermediate control schemes** can employ e.g.,a hierarchical structure with a central organization on different hierarchical levels. Example algorithms are autonomous VPP (AVPP) scheduling, see Anders, Siefert, et al. 2010, EPOS, see Pournaras 2013 or the approach presented in Tröschel 2010.

Central control systems and intermediate ones with their central elements suffer from data protection. Furthermore information transfer within a VPP is a problematic issue. DER owners do not always want to share information about their device and for some DER there might not be enough information available at all. In central control systems DER information has to be transfered to a central scheduling unit and in intermediate control systems to a pool coordinator or intermediary agents, see Hinrichs 2014, p 3 and Faltings and Yokoo 2005.

- **Fully distributed control systems**, where distributed units communicate without a central unit, are the only control schemes, where operation schedules are the only information DER have to share. In VPPs with fully distributed control systems each DER acts as an independent and intelligent agent in the VPP scheduling processes, e.g., in COHDA, Hinrichs, Lehnhoff, et al. 2014 or DynaSCOPE, Nieße 2015.

Fully distributed control systems are commonly implemented with heuristics. Beside the two advantages of full distributed control systems, a good scalability and data protection, the applied heuristics employ one disadvantage concerning the solution quality. Fully distributed control systems do not always find the best solution, due to incomplete information, see Gatterbauer 2010. But altogether fully distributed approaches were identified as the most promising approaches for VPP scheduling, see F. F. Wu et al. 2005; Ilić 2007; McArthur et al. 2007, Hinrichs 2014, 3 seqq. and Bremer 2015, p. 30.

Fully distributed systems are often realized with multi agent systems (MAS). In multi agent systems, an agent is allocated to each decentralized stakeholder. In the case of VPP scheduling the stakeholders are the energy units or energy unit coalitions. In the following subsection two such VPP scheduling algorithms are presented.

1.4.1 VPP Scheduling Approaches

In literature there are currently two much discussed VPP scheduling approaches with a more or less fully distributed control structure. One approach called COHDA (Combinatorial Optimization Heuristic for Distributed Agents) that works with a fully distributed control system. The other approach called AVPP (autonomous VPP) scheduling utilizes hierarchical (intermediate) control structures. Even though fully distributed control structures were identified as more promising than intermediate control structures, the AVPP scheduling approach has some advantages properties.

COHDA (Combinatorial Optimization Heuristic for Distributed Agents) is a distributed heuristic for DCOPs that operates in a distributed and asynchronous way, see Hinrichs, Sonnenschein, et al. 2013; Hinrichs, Lehnhoff, et al. 2014. COHDA was developed for predictive VPP scheduling tasks in the context of wholesale market trading at the day-ahead market. Therefore each software agent represents an energy unit of the participants of the considered VPP. Each agent employs a search space containing all realizable solutions (feasible operation schedules) that the unit can contribute to the solution of the optimization problem. Due to the main scheduling objective to plan a certain energy product, it is not possible to choose an optimal operation schedule for one energy unit from its search space without considering the operation schedule choice of all other energy units, see Hinrichs 2014, p 25.

Beside the presented application context COHDA has been adapted and extended to applications in continuous/ adaptive scheduling tasks in Nieße 2015 to DynaSCOPE (Dynamic Scheduling Constraint Optimization for Energy Units).

AVPP Scheduling is related to resource allocation as a hierarchical optimization problem. AVPPs (autonomous VPPs) are VPPs with a hierarchical control system. AVPP scheduling was proposed in Anders, Siefert, et al. 2010; Steghöfer et al. 2013 for reactive VPP scheduling to meet a predicted demand. The approach is a system of systems approach, where each of these systems is called AVPP. AVPPs on the lowest hierarchical level coordinate some energy units, while AVPPs on higher levels coordinate AVPPs on subordinate levels. The division into AVPPs represents the power grid structure. Each AVPP optimizes the supply-demand-matching either according to a regio-central control strategy, see Schiendorfer, Steghöfer, et al. 2014; Schiendorfer, Anders, et al. 2015 or an auction-based decentralized control strategy, see Anders, Schiendorfer, et al. 2015. Both control strategies employ search space encodings for all agents on the different levels, representing an energy unit, a collective of energy units or a collective of collectives.

The optimization task in both VPP scheduling approaches are restricted by constraints. The constraints can be either integrated into the search spaces (separate treatment) or they can be treated directly. In COHDA the constraints are integrated into the search spaces, while in AVPP scheduling some constraints are integrated into the search spaces and others are handled directly. The search spaces in both approaches represent all realizable operation modes in terms of feasible operation schedules. This means the flexibility of the respective energy units and coalitions of energy units is encoded in the search spaces, see Sect. 1.2. Therefore the search spaces can be also called flexibility descriptions. Search space modeling of both approaches is presented in more detail in the following. Afterwards both approaches are compared with regard to their applicability.

1.4.2 Search Space Modeling in COHDA

The search space of each agent in COHDA is bounded by intra energy unit constraints. These constraints are used to restrict the search spaces of all energy units to feasible solutions with the help of separate constraint handling.

Two different search space realizations were proposed in Hinrichs 2014, 117 seqq.

- One possibility is a list of alternatively feasible operation schedules that were sampled from a functional description of the energy unit.
- The other possibility are machine learning surrogate models in combination with a decoder approach.

The list representation is suitable for very small sets of alternatively feasible operation schedules. The higher the number of schedules in the list the better are the optimization results. But larger sets of operation schedules strongly increase the required amount of memory capacity and also the computational complexity, see Hinrichs, Bremer, and Sonnenschein 2013. It would be also possible to use functional descriptions as search space descriptions, because all feasible operation schedules could be sampled from that functional description. But functional descriptions are not always easy to derive. Due to dispensable information, functional descriptions can slow down the optimization. Additionally lists and sampling from a functional description hamper a target-directed search. Surrogate models with decoder approaches overcome these difficulties, see Nieße 2015, 104 seq. They are simple description of the set of all alternatively feasible operation schedules. In Bremer et al. 2011 a constraint handling method for encoding of distributed search spaces for virtual power plants was proposed and integrated into COHDA in, Hinrichs, Bremer, and Sonnenschein 2013.

Search Space Encoding

The proposed encoding of distributed search spaces for single energy units in Bremer et al. 2011 is a decoder approach consisting of the two steps: finding an unconstrained search space representation and sampling from the search space.

Search space representation An unconstrained search space representation is built with a machine learning classifier. Support Vector Machines (SVMs) were identified as promising models, because they describe the set of all alternatively feasible operation schedules in an abstract way and can determine the feasibility of new operation schedules, see Bremer 2015, p. 72. Since the set of all alternatively feasible operation schedules relies on the energy units flexibility, the description is also called flexibility description. This flexibility description is built from given feasible operation schedules. As far as usually only feasible operation schedules are available for building the classifier a classifier from one class classification, Support Vector Data Description (SVDD) is applied, Bremer 2015, p. 60. The SVDD classifier is build from a training set consisting of some alternatively feasible operation schedules in combination with a (Gaussian) kernel transformation. This kernel transformation performs a mapping of the training data to high dimensional space, where the training examples form a minimal hypersphere. The most relevant training examples lie on the surface of the hypersphere and they are called support vectors. Based on those support vectors a hypersphere center a and a hypersphere radius $R_{\mathscr{S}}$ can be determined. The gained abstract description of the search space ($F = f(R_{\mathscr{S}}, a)$) allows an easy decision concerning the feasibility of new operation schedules.

Search space sampling The second step of search space encoding consists in sampling feasible operation schedules from the search space description as possible solutions for the optimization task. For convenience the description and all operations schedules are scaled according to maximal power production/ consumption to values between 0 and 1. This scaling effects that all feasible and infeasible operation schedules form a hypercube. As far as sampling from the SVDD description is not possible, a mapping is needed, that maps each given operation schedule into the region of feasible ones. Instead of a direct mapping in d-dimensional feature space $\mathbb{R}^d$, a mapping has to be done in three steps by a mapping through the ℓ-dimensional kernel space $\mathscr{H}^{(\ell)}$, see Fig. 5.1.

1. At first an arbitrary point x from the unconstrained hypercube is chosen and mapped onto $\hat{\Psi}_x$ in an ℓ-dimensional manifold in kernel space $\mathscr{H}^{(\ell)}$. This ℓ dimensional manifold is spanned by the ℓ support vectors.

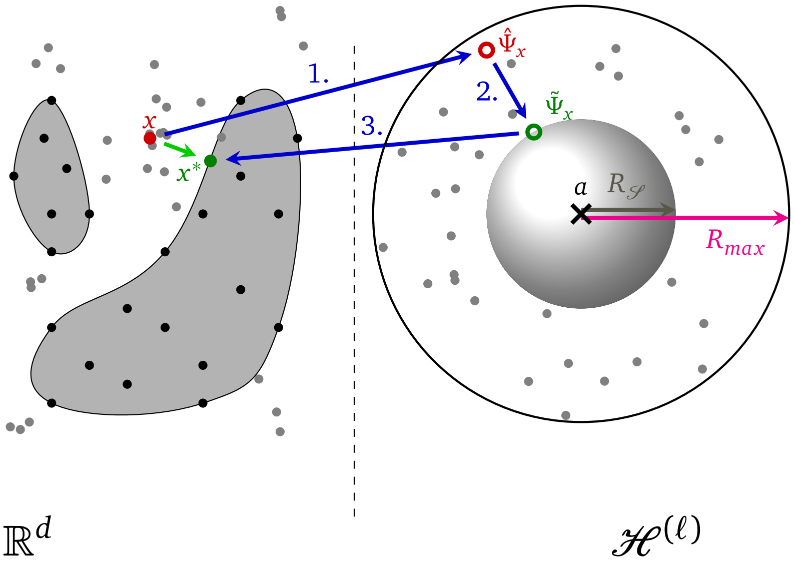


Figure 1.1: Decoder mapping. The figure is an adaptation of the decoder mapping figure in Bremer 2015, p 119. The point x is pulled into the feasible region of the feature space $\mathbb{R}^d$ to the point x^* via a mapping through the kernel space $\mathscr{H}^{(\ell)}$. R_{max} indicates the maximal distance of an infeasible point to the hypersphere center a and $R_{\mathscr{S}}$ indicates the maximal distance of a feasible point to a.

2. Secondly the mapped point $\hat{\Psi}_x$ in kernel space $\mathscr{H}^{(\ell)}$ is drawn to the hypersphere to $\tilde{\Psi}_x$ in order to pull it into the image of the feasible region.
3. Thirdly an appropriate pre-image x^* of the manifolds is searched. Since it is hardly possible to find the exact pre-image a pre-image is searched, that lies closest to the given image by using an iterative procedure after Mika et al. 1999.

This flexibility description encodes the search space of a single energy unit for a given initial state.

1.4.3 Search Space Modeling in AVPP Scheduling

The search spaces in AVPP planning are encoded with control models. Control models of single energy units are called individual agent models and control models of a collective of energy units or a collective of collectives are called abstract agent models. All these control models are descriptions of all feasible contributions for each time step (supply trajectories), see Schiendorfer, Anders, et al. 2015. Depending on the underlying energy unit(s), the feasible contribution ranges do not imply all constraints and the remaining constraints have to be treated during optimization.

Search Space Encoding

These control models are piecewise linear functions and they are mainly formulated as supply automata. Supply automata of single energy units and coalitions of energy units are derived in different ways.

Control Models of Single Energy Units Control models of single energy units, also called individual agent models are mainly formulated as supply automata. Control models of single energy units can be derived from energy unit models or from observed supply trajectories.

Individual agent models from energy unit models are derived from given constraints, such as minimal up/ down times, ramp up/ down rates or cold/ warm start-up times, see Schiendorfer, Anders, et al. 2015.

Individual agent models from observed supply trajectories are derived when energy unit models do not exist or when they are not available due to privacy concerns, see Anders, Schiendorfer, et al. 2015. Control models are derived with the help of a classifier, that describes the set of observed supply trajectories, see Anders, Schiendorfer, et al. 2015. A support vector data description (SVDD) classifier is trained on the observed feasible schedules. The classification accuracy is increased by feeding some infeasible schedules into the training set and into cross-validation. Then the feasibility of a grid of test points spanning the observed range of contributions is determined. To achieve the control model formulation of feasible contribution intervals for each time step, the positively classified test points are projected on each axis (corresponding to one time step) and all (one or more) feasible contribution ranges are identified for each time step.

Control Models of Coalitions of Energy Units Control models of coalitions of underlying energy units or coalitions are called abstract agent models and they are formulated as supply automata. The abstract control models represent the joint behavior of all underlying energy units or coalitions, see Schiendorfer, Anders, et al. 2015. Supply automata are derived in two steps, synthesis and abstraction. Synthesis defines necessary interfaces to convert a set of heterogeneous control models along with the objective to obtain an optimization problem, see Schiendorfer, Steghofer, et al. 2014. Abstraction based on supply automata aims at reducing the complexity of high level optimization processes, see Schiendorfer, Steghöfer, et al. 2014; Schiendorfer, Anders, et al. 2015.

Synthesis Synthesis is the process of converting a set of control models into one optimization problem. This procedure consists in a combination of models provided on lower levels and an augmentation with organization specific constraints, see Schiendorfer, Steghöfer, et al. 2014; Schiendorfer, Steghofer, et al. 2014.

Abstraction Abstraction reduces complexity and means abstracting the information contained in the models of one region to form a new model of the considered hierarchy level. The reduction comes at the cost of errors due to imprecision. Abstraction is based on the constraint models of the underlying units and is divided into general abstraction, temporal abstraction and sampling abstraction, see Schiendorfer, Steghöfer, et al. 2014; Schiendorfer, Anders, et al. 2015. These three steps start with the combination of the subordinate models to a general model that is fine tuned by adding operation specific constraints in the following steps.

general abstraction During general abstraction the generally feasible contribution ranges of the collective are determined. This leads to a sorted list of intervals corresponding to joint modes of the underlying suppliers, that are merged.

temporal abstraction In the next step, temporal abstraction, the generally feasible contribution range is restricted by the current state and constraints resulting from the possible behavior of the underlying energy units. Even though temporal abstraction excludes ranges after t time steps, it does not offer any boundaries between two consecutive time steps and therefore the third abstraction step, sampling abstraction is applied.

sampling abstraction In the sampling abstraction step the constraints given by the composition of agent models and the constraints given by the input values are handled. Sampling probes a collective's functional relationship. Sampling points are selected equidistantly across the full range of power production. To achieve more informative points a systematical selection is proposed using importance sampling from active learning, see Schiendorfer et al. 2015b; Schiendorfer et al. 2015a.

Control models can be partly re-used for further applications, because global abstraction does not consider initial states and therefore only temporal abstraction and sampling abstraction have to be adapted.

1.4.4 Search Space Encoding Comparison

The different search space modeling approaches, employed in COHDA and AVPP scheduling, describe the flexibilities of all VPP participants differently. In this subsection both search space approaches, respectively flexibility descriptions, are compared with regard to applicability. Therefore the scheduling approaches and some search space modeling properties of both approaches are briefly presented in Table 1.1. Both

	flexibility description (COHDA)	**control model (AVPP)**
search space modeling		
scheduling	predictive	reactive
no. of energy units	one	one or more
search space description	set of feasible schedules	ranges of feasible contributions
constraint handling	SVDD + decoder	control models and abstraction
search space model	SVDD	supply automata
model is build from	feasible schedules	given constraints or supply trajektories or supply automata
search space sampling	decoder approach	sampling from a list of feasible contribution ranges
search space properties		
time modeling	time discrete	time discrete
no. of time steps	96 (default)	48 (default)
max. no. of time steps	up to ca. 100	50 per control model concatenation of control models
temporal resolution	15 min (default)	15 min (default)
tested mainly with	μCHPs, cooling devices	hydro-, biofuel-, gas- and conventional power plants

Table 1.1: Comparison of the two search space modeling approaches, the flexibility description from COHDA and the control models form AVPP scheduling.

search space models are compared. Applicability is considered with regard to scalability, reusability, data protection and performance issues. This comparison is concluded by a discussion of the findings in the context of the two meanings of the term VPP scheduling, scheduling of single energy units and scheduling of coalitions.

Scalability The applicability of the search space models depends on temporal scalability and scalability concerning the kind and the number of considered energy units.

temporal scalability As indicated in Tab. 1.1 both search space models can be applied with different temporal horizons and temporal resolutions within reasonable ranges. But the temporal horizon and the temporal resolution are limited by a maximal number of time steps. The flexibility description used in COHDA suffers from the dimensionality of the operation schedules, see Bremer and Sonnenschein 2013b; Bremer and Sonnenschein 2013c. On the one hand the curse of dimensionality means increasing numbers of time steps that require a drastic increase of training examples for the SVDD to prevent precision losses. Precision losses in the SVDD model are transfered to the decoder model, see Bremer and Sonnenschein 2013b; Bremer and Sonnenschein 2013c. On the other hand the decoder has problems to distinguish between high-dimensional feasible and infeasible operation schedules. This is due to decreasing distances between schedules for increasing numbers of time steps. This phenomenon is described in Köppen 2000; Evangelista et al. 2006, as increasing dimensionality leads to pairwise the same distance of two arbitrary points.

Recently a partitioning of search spaces was proposed in Hinrichs, Bremer, Martens, et al. 2016 to reduce the computational complexity of the SVDD and the decoder approach and to avoid the drawbacks of the curse of dimensionality. This search space partitioning could improve the solution quality in scheduling tasks even though some sub search space parts are missed. The application of search space partitioning turns the optimization problem into a sequential optimization problem.

Control models in AVPP scheduling are also bounded by a maximal number of time steps. But this limitation is overcome by a periodically creation of new control models every 48 time steps. Since the initial state for each control model is needed, 96 new time steps have to be simulated before the creation of a new control model, see Anders, Schiendorfer, et al. 2015. Independently from temporal restrictions, arbitrarily long temporal scheduling horizons are not reasonable due to prediction inaccuracies.

energy unit scalability Energy unit scalability considers the kind of energy units and the number of energy units represented in one search space. While the flexibility description was designed for different kinds of controllable DER and has been mainly tested with μCHPs and cooling devices, the control model was designed for producers and consumers and is mainly tested with hydro-, biofuel- and gas power plants. As far as the control models are built on the assumption that power output only depends on the previous time step, see Anders, Schiendorfer, et al. 2015, energy units with further dependencies cannot be applied without adaptations, like e.g., CHPs and heat pumps that depend on the thermal buffer temperature and a thermal demand profile, or cooling devices that depend on more than one previous time step.

The number of energy units that can be represented in one search space is limited to one in the flexibility description. Search space modeling of a collective of units constitutes two problems, see Bremer 2015. First of all the number of feasible global operation schedules grows exponentially with the number of energy units in the collective. Therefore the number of required training examples for the SVDD black box model is expected to increase with increasing numbers of considered energy units. The second problem consists more or less in the generation of global operation schedules that represent the whole feasible class. Usually the generation of feasible global operation schedules leads to many schedules near the center of the feasible class and hardly any examples at the margins of the feasible class. Beside the inhomogeneous distribution also the representativity of the feasible class is problematic.

Thus SVDD black box models learn only the region with the high schedule density.

In contrast to flexibility models, control models can represent the search space of one or more energy units. Control models of coalitions of energy units lose precision during the aggregation of the subordinate models and their abstraction, see Schiendorfer, Steghöfer, et al. 2014; Schiendorfer, Anders, et al. 2015.

Concerning scheduling of single energy units, the flexibility description applied in COHDA is applicable to different kinds of energy units with a more or less limited scheduling horizon. Search space partitioning allows longer scheduling horizons. Control models in AVPP scheduling have no temporal limits, but the kind of applicable energy units is restricted by the assumption that energy production at the next time step $(t + 1)$ only depends on the previous time step (t). Control models of coalitions of energy units are aggregations of all control models of the coalition members and therefore all control models of the coalition members need to be known.

Reusability Reusability of search space models is only possible for control models, because general and scenario specific constraints are integrated after another. But the stepwise constraint integration requires specific energy unit information, like e.g., on and off settings, minimal and maximal supply, ramping rates and start-up times. Flexibility descriptions cannot be reused, because they depend on the initial state of the energy unit. But the parametrization of the SVDD is expected to be similar for different initial states of one energy unit.

Data Protection Data protection and privacy issues depend on the search space encoding.

The flexibility description achieved a high degree of privacy protection, because the flexibility description is built from operation schedules and yields only an abstract description of the set of all feasible operation schedules. Opposed to the flexibility description, control models protect privacy less. Especially control models on higher levels need beside the subordinate control models also information about feasible operation modes.

Performance Search space model performance relies on precision and time complexity. As mentioned for the scalability item before, precision decreases with increasing numbers of considered time steps. Long scheduling horizons are less reasonable anyway, because they depend on inaccurate predictions of generation and loads. Furthermore control models of coalitions of energy units lose precision compared to single energy unit control models, due to imprecisions during aggregation and abstraction.

Time complexity of search space model computation is less important. Search space models have to be computed once before the optimization of the scheduling task.

Summing up the pros and cons of both search space modeling approaches with regard to the two meanings of the term scheduling, non of the two approaches is applicable without strong limitations. Both search space models describe the flexibilities of the VPP participants. The search space model applied in COHDA (named flexibility descriptions) is only successfully applicable to single energy units. The search space modeling from AVPP scheduling (named control models) are also applicable to scheduling of coalitions of energy units, but they are not applicable to all kinds of controllable DER. Furthermore research need is claimed for more realistic and complex control models in Anders, Schiendorfer, et al. 2015.

Altogether none of the search space model approaches is applicable to VPP scheduling with its two meanings for arbitrary kinds of DER. It seems easier to extend the search space model of COHDA, especially because of the advantages: applicability to different kinds of DER and data protection.

1.5 Objective

The comparison of existing search space descriptions revealed a need for further research. None of the existing search space descriptions is applicable to different kinds of controllable energy units in form of single energy units and coalitions of energy units. Beside these two requirements search space descriptions should be applicable to VPP scheduling with a fully distributed control structure which has been identified as a promising control structure e.g., in F. F. Wu et al. 2005. The advantages of fully distributed control structures, a good scalability and data protection can be utilized best with a separate constraint handling as applied in COHDA, see Hinrichs 2014, 117 sepp. Separate constraint handling does not require detailed energy unit information which is not always available. Search space models like the flexibility description used in COHDA only rely on feasible operations schedules. Furthermore search space descriptions should precisely represent feasible possible solutions for all applications. The properties reusability and time complexity are less important, because search space descriptions always have to be computed before the optimization in scheduling tasks. Overall a search space model should be abstract, consistent and precise.

Abstract A search space description should be an abstract description representing the flexibility. Therefore such a search space description is called flexibility description like the search space modeling approach used in COHDA. A flexibility description should be less complicated than simulation models of the respective energy unit information and should hide energy unit information. Preferably a flexibility description is a description of the set of all alternatively realizable operation schedules that is derived from given schedules.

Consistent A flexibility description should be consistent for various applications. For one thing the flexibility description should be applicable to different kinds of controllable energy units. For another thing the flexibility description should be applicable to single energy units and coalitions of energy units. Furthermore the scheduling horizon and the temporal resolution should be adaptable.

Precise The flexibility description should be a precise description of the quantity of all alternatively realizable operation schedules. Furthermore the flexibility description should only contain feasible operation schedules or at least as few inaccuracies as possible, to allow good optimization results for the scheduling tasks.

These requirements lead to the aim of my PhD project.

The aim of my PhD project is the development of an abstract, consistent and precise flexibility description for the application as a search space model in VPP scheduling in terms of scheduling the single energy units in a VPP and in terms of scheduling coalitions of energy units as units.

Even though I develop a flexibility description in the context of VPP scheduling, its application should not be restricted to this context.

1.5.1 Structure of this Thesis

This thesis is structured as follows. In the following chapter, Chapt. 2 the objective is transfered into a methodological task. Therefore different methods are presented and compared concerning their applicability as flexibility description. Furthermore evaluation methods for the flexibility model to be developed are presented. Then the development of the flexibility description and the resulting flexibility description are presented in Chapt. 3. In Chapt. 4 the flexibility description is evaluated concerning precision, parameter sensitivity and temporal complexity. In the next chapter, Chapt. 5, the applicability of the developed flexibility description is discussed in the context of VPP scheduling. Finally a summary of this thesis, a conclusion and a perspective are given in Chapt. 6.

2 METHODS

As identified in the previous chapter in Sect. 1.5, the flexibility description should be an abstract consistent and precise description of the set of all alternatively realizable operation schedules. Machine learning models in combination with decoder functions are suitable as flexibility descriptions due to different reasons, see e.g., Alpaydin 2010, 1 seqq. Machine learning models extract information from given data sets. Furthermore machine learning models aim to be general and precise. Therefore they are applicable to various different tasks. An overview of machine learning models can be found e.g., in Bishop 2006; Hastie et al. 2009; Alpaydin 2010; Kuncheva 2014. Especially methods from classification and clustering aim at describing classes and separating them from the remaining examples. Classification methods are supervised methods that yield more precise results on labeled data, than unsupervised clustering methods. Operation schedules used for flexibility descriptions are labeled according to their feasibility. Therefore classification methods are more appropriate.

This chapter consists of a brief classification method overview in Sect. 2.1, the identification of appropriate classifiers for a flexibility description in Sect. 2.2, Sect. 2.3 and the presentation of the applied classifiers in this thesis in Sect. 2.4 and classifier performance evaluation methods in Sect. 2.5.

2.1 CLASSIFICATION

Classification in general means grouping similar objects into classes according to properties that all objects share in a class, see Sammut and Webb 2010. Classification in terms of machine learning is the assignment of a class label to an object based on a set of example objects, see e.g., C. C. Aggarwal 2014, p. 2, Kuncheva 2014, p. 9ff and D. M. J. Tax 2001, p. 2ff. An object is described by a vector x_i, consisting of a set of d measurements of different object properties. This vector is called feature vector and represents one point in feature space. The object $x_i \in \mathbb{R}^d$ can be represented as feature vector $(x_{i,1}, \ldots, x_{i,d})$ with $x_{i,j} \in \mathbb{R}$. Each object x_i belongs to a class $\omega \in \Omega = \{\omega_1, \ldots, \omega_k\}$. The class affiliation of x_i is indicated by a label y_i. Usually classification tasks consist of two classes with $k = 2$, $\Omega = \{\omega_1, \omega_2\}$ and the class labels are indicated as $y_i \in \{-1, +1\}$.

Classifiers are learned from a so called training set X_{train} consisting of N objects and the corresponding labels. A classifier is a function $f(x)$, that predicts the class label y of a new object x as $y = f(x)$ with $f : \mathbb{R}^d \rightarrow \Omega$.

Classification is a broad field and emerged in different communities. Classification can be divided into three broad categories, according to C. C. Aggarwal 2014. These three categories are technique-centered methods, data-type centered methods and variations on classification analysis.

- The first category focuses on the **algorithms** and covers most popular classification methods ranging from probabilistic methods over decision trees, rule-based methods, instance based methods to support vector machines and neural networks.

- The second category focuses on different **data types**, because different data types require different classification methods and a special data treatment. The most frequently considered data types are data streams, big data, texts, multimedia data, time series, sequence data, data from networks and uncertain data. Beside the data type examples in C. C. Aggarwal 2014, further well known data types are micro-arrays, handwriting data sets and document data sets.

- The third category covers **classification algorithm adaptations**, that were proposed for variations in the data sets. This category covers variations of the standard classification problem and enhancements of classification with the use of additional data. Variations of the standard classification problems are e.g., rare class learning, distance function learning and ensemble learning. Furthermore there are enhanced classification methods with additional data, semi.-supervised learning, transfer learning, classification methods incorporating human feedback, active learning and visual learning.

Beside the presented classification categories by C. C. Aggarwal 2014 further categorizations, e.g., based on the classification algorithms properties can be found in literature.

2.2 CLASSIFIER CHOICE

The identification of an appropriate classifier for a given task can be a challenging task due to the high number of existing classifiers. Based on the classification task and the given data set an appropriate classifier has to be identified. A scheme for choosing data mining algorithm, including classification algorithms was proposed in Gibert and Codina 2010. This scheme suggests to consider on the one hand "*the main goal of the problem to be solved*" and on the other hand "*the structure of the available data*".

Main goal of the problem to be solved The main goal is the identification of a classifier, that describes the set of all feasible operation schedules. This classifier, serving as flexibility description should be an abstract description of the set of all alternatively feasible operation schedules. Classifiers yield abstract descriptions, because they define a class e.g., by their boundaries, reconstruction properties or density properties, see D. M. J. Tax 2001, chap. 3. Furthermore the classifier should be precise and consistent. This means the learned decision boundary should resemble the true class boundary for different applications with different data properties and structures.

Beside the requirements resulting from the main goal, see Sect. 1.5, the classification task itself has some characteristic properties, that are also important for the classifier choice. These additional classification task properties resulting from the VPP scheduling application are summarized in Tab. 2.1.

Structure of the available data The data structure of operation schedules is analyzed with regard to the data type and the data set properties identified in pre-studies exemplarily for CHP and heat pump operation schedules of single units and coalitions of energy units. Operation schedules in VPP scheduling tasks belong to the data type: time series. Data set properties commonly considered in classification tasks, see e.g. Weiss 2004; Sudjianto et al. 2010; Lusa

property	value
number of classes	2 (feasible, infeasible)
balance of classes	(severely) imbalanced (small feasible class)
dimensionality	high-dimensional (e.g., $24h$, $15min$ resolution)
feature values	numeric values (mainly scaled to $[0,1]$)
separability of classes	clearly separable (no class overlap)
fragmentation (feasible class)	possible (one or more concepts)

Table 2.1: Classification task properties.

property	value
number of classes	1 or 2 (only feasible class or both)
data source	simulation models
number of examples	arbitrary numbers of examples can be sampled
distribution of examples	from homogeneous and representative to inhomogeneous and less representative
correct labels	yes
missing labels	no
balance of data set	classes are balanced
noisy examples	no
missing values	no
concept drift	no

Table 2.2: Data set properties.

and Blagus 2012; C. C. Aggarwal 2014; Bak 2015; Stefanowski 2016 and their characteristics in operation schedule data sets are listed in Tab. 2.2. Data set properties can deviate from the classification task properties in Tab. 2.1, like e.g., the number of classes. The classification task is based on two classes, but the data sets may contain only examples of one class.

2.3 CLASSIFICATION CATEGORIES RELATED TO THE IDENTIFIED CLASSIFICATION TASK AND DATA PROPERTIES

The analysis of common classification task properties and common data properties revealed characteristics that are representative of different classification categories. The main classification task characteristics are the data type time series, (severe) imbalanced classes and high dimensional data. Each of these properties fits into one or more of the following classification category: time series classification, high-dimensional learning, imbalanced learning, high-dimensional and imbalanced learning and one-class classification. But there is no category including all these properties. The categories called "...-learning" comprise beside classification methods also other methods like clustering and regressing. But I focus only on classification methods. In the following related classification categories are presented with respect to suitable classification methods for flexibility descriptions.

2.3.1 Time Series Classification

Time series are sets of of m real-valued numbers, Sammut and Webb 2010; Kotsakos and Gunopulos 2014. A time series is represented as a sequence $T = (t_1, t_2, \ldots, t_m)$. Time series classification differs a little from common classification mainly due to two properties.

- The first difference consists in the **dimensionality**, Sammut and Webb 2010. Time series data sets often employ hundreds or thousands of time steps, while classification algorithms for common tasks assume lower numbers of features.

- The second difference consists in the **data structure**, see Sammut and Webb 2010; Bagnall, L. M. Davis, et al. 2012 and J. A. Lines and Bagnall 2015, p. 7ff. For one thing ordering of the features (time steps) is important. For another thing data points within one time series can be highly correlated.

According to B. Fulcher and N. Jones 2014 time series classification tasks belong to the two categories: instance-based classification and feature-based classification. Instance-based classification comprises approaches, where new and short time series are classified by matching them to similar labeled time series. Feature-based classification is based on transformations of the time series representations and is usually applied to longer time series. The time series are represented by a set of derived properties or features.

Time Series Classification Approaches

Classification tasks from both categories have to be treated differently. Instance-based classification tasks (short time series) are similar to common classification and can be treated with common classification algorithms. Feature-based classification tasks (long time series) require data preprocessing. Often applied preprocessing methods are dimensionality reduction, see e.g., Al-Naymat and Taheri 2008; Krawczak and Szkatuła 2014, feature selection and instance selection, see e.g., Tsimpiris and Kugiumtzis 2012; Tomašev, Buza, et al. 2015, indexing, see e.g., Xiong and Funk 2008; Camerra et al. 2010; Fu 2011 and segmentation, see e.g., Fu 2011; Keogh et al. 2004, Spiegel 2015, 20 seq. Furthermore transformations into a different representation, Fu 2011; Kotsakos and Gunopulos 2014 are used in combination with different preprocessing methods, see Ding et al. 2008; Hills et al. 2014. Beside preprocessing methods, time series classification algorithms are often applied with special metrics, see e.g., Giusti and G. E. A. P. A. Batista 2013; Serrà and Arcos 2014; Spiegel 2015; J. A. Lines and Bagnall 2015. In experiments k nearest neighbors (knn) with dynamic time warping turned out to be the best time series classification approach for smaller data sets, see Ding et al. 2008. Recently superior time series classification algorithms have been proposed: *transformation based ensembles*, Bagnall, L. M. Davis, et al. 2012 and *collective of transformation-based ensembles*, seen Bagnall, J. Lines, et al. 2015.

Applicability to Flexibility Descriptions

Time series classification algorithms have to be treated with care for flexibility descriptions. Common time series classification algorithms rely on similarities, like frequencies, shifts, etc. within the time series, like e.g., the k nearest neighbor classifier used in the transformation based ensembles time series classification model, see Bagnall, L. M. Davis, et al. 2012. But feasibility of operation schedules relies on the power production or consumption value at each time step. A feasible operation schedule gets infeasible, if only the power production/ consumption value of a single time step is changed to a not realizable value. Therefore common classification algorithms used for instance-based classification are more appropriate for operation schedule classification, even though operation schedules often employ 96 time steps $(24\,h,\ 15\,min)$.

From the field of feature-based classification only one type of preprocessing is applicable, namely time series segmentation. All other methods can not be applied, because the feasibility of a new operation schedule depends on the power production or consumption at each time step. Time series segmentation is also called time series splitting, splits time series into segments according to time series inherent logical units, e.g. patterns, Molina et al. 2009; B. D. Fulcher et al. 2013; B. Fulcher and N. Jones 2014; Gensler and Sick 2014.

An similar version of time series splitting can be also found for common classification tasks in the context of random forests, called feature bagging. Feature bagging is a procedure, where the features are divided into a specific number of subsets of possibly overlapping features (feature bags), see e.g., Breiman 2001; Sutton et al. 2005. These feature bags are learned with separate models and their predictions are combined to final results.

2.3.2 High-dimensional Learning

Classification tasks with many features are a problem in general due to the curse of dimensionality, see e.g., Sammut and Webb 2010; Bak 2015. The curse of dimensionality is related to the exponential increase in volume due to additional dimensions in Euclidean space, Bellman and Bellman 1961. This volume increase implies, that the number of required samples for classification tasks increases exponentially with the number of dimensions (features). The influence of high-dimensionality one classifier performance depends on additional factors. In Lin and Chen 2013 five major factors were identified that often influence classification performance of high dimensional data. Three factors concern data properties: imbalance ratio, minority and majority class distribution and sample size and two factors concern the algorithms: feature selection and the strategy for imbalance correction.

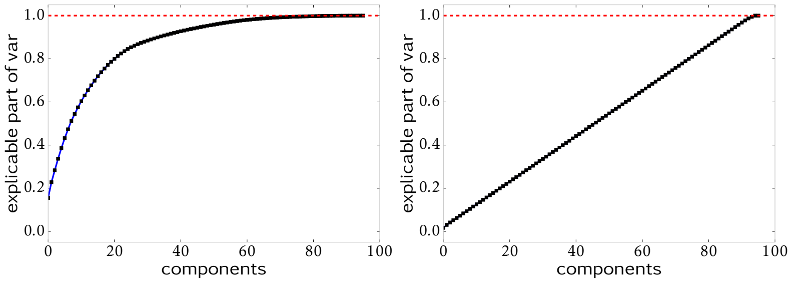
High-dimensional Classification Approaches

In literature, various solutions have been proposed for high-dimensional classification tasks. High-dimensional data sets are usually treated with preprocessing algorithms and or special classification algorithms. Such preprocessing methods are dimensionality reduction, Engel et al. 2012; Wang et al. 2015, feature selection, Saeys et al. 2007; Chandrashekar and Sahin 2014; Yin and Gai 2015 and data sampling methods, see e.g., Yin and Gai 2015. Special classification algorithms are e.g., ensemble approaches, see Piao et al. 2014; Krawczyk 2016, multi-task learning approaches, see Seo and Oh 2013; X. He et al. 2014; Piao et al. 2014 or hierarchical classification approaches, see Naeini et al. 2014; Valentini 2014.

Applicability to Flexibility Descriptions

Common feature selection and dimensionality reduction work well for common classification tasks, but they are not appropriate for time series classification tasks. As the ordering of all time steps is important and features (time steps) cannot be redundant or meaningless, see Bagnall, L. M. Davis, et al. 2012. Dimensionality reduction with linear Principle Component Analysis (PCA) and PCA with a Gaussian kernel revealed exemplarily for μCHP power output time series that most of the components contribute to the explicable variance, see Fig. 2.1. The explicable variance ratio indicates the percentage of variance, that is explained by a selected component.

From the field of special classification algorithms, multi-task learning approaches are promising for operation schedule classification, because they simplify classification tasks without a loss of information. The main idea of multi-task learning is to split the classification task in such a way, that each concept or parts of a concept are learned separately, see Fig. 2.2 and see e.g., Seo and Oh 2013. Beside multi-task learning approaches, ensemble approaches seem also promising for operation schedule classification. The advantage of ensembles approaches consists in a high overall classification precision, due to a combination of a set of diverse classifiers, e.g., see Sammut and Webb 2010.



(a) linear PCA (b) PCA with gaussian kernel

Figure 2.1: Cumulative sum of the explicable variance ratio of 10,000 CHP operation schedules (black line) and 100% of the explicable variance is indicated by a red dotted line.

2.3.3 Imbalanced Learning

Imbalance refers to unequal distributions that often cause classification errors on the minority class. Generally the term imbalance is used for data sets employing different numbers of examples for different classes. But imbalance holds more facets, see Weiss 2004; Kotsiantis et al. 2006; H. He and E. Garcia 2009; H. He 2013; Weiss 2013. According to H. He and E. Garcia 2009 two main categories of imbalance can be distinguished: between-class imbalance and within-class imbalance. These two categories employ several subcategories.

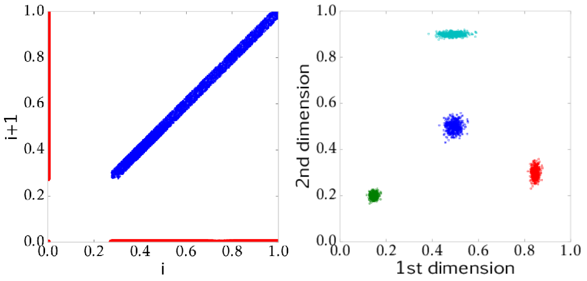
Between-class Imbalance This category subsumes data sets with imbalanced classes independent from the origin and the cause the of the imbalance. Imbalance originates either from the problem or the sampling.

Intrinsic Imbalance Imbalance, directly resulting from the nature of the problem is called intrinsic imbalance.

Extrinsic Imbalance Extrinsic imbalance results from sampling, e.g., if sample recording of balanced classes misses numerous samples of one class.

Further subcategories can be distinguished according to the causes of imbalance.

Relative Imbalance Relative imbalance refers to the sample ratio between



(a) μCHP concepts (b) Hypersphere concepts

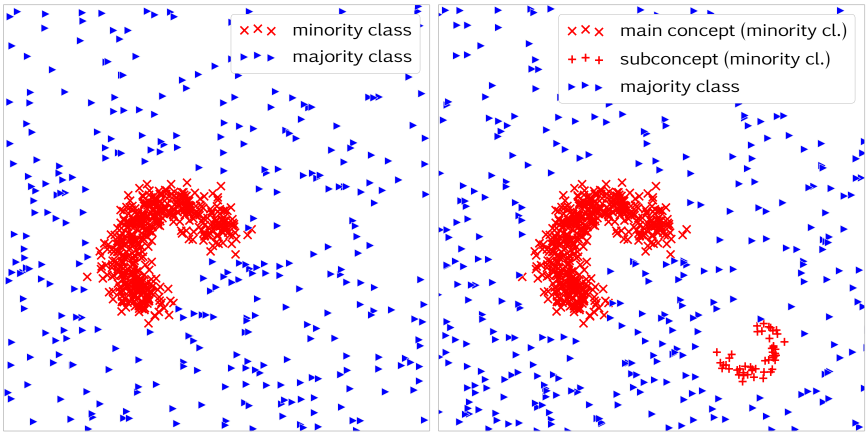
Figure 2.2: 2-dimensional figures of data sets, where the feasible class consists of several concepts. Different concepts are indicated with different colors.

the classes. The number of examples of the minority class is only small in comparison to the number of examples of the majority class. This means doubling the number of examples doubles the number of examples in both classes.

Imbalance due to Rare Cases In the case of imbalance due to rare cases, the minority class examples are very limited, e.g., the target concept is rare. This means a lack of representative data of the minority class and can lead to an underrepresentation. This imbalance form can be even worse in combination with within-class imbalance.

Within-class Imbalance Within-class imbalance is present, when a class employs one or more subconcepts with a limited number of samples. This form of imbalance is related to small disjuncts that are easily misclassified, especially in the presence of noise.

Most of the presented subcategories of imbalance are shown in Fig. 2.3. Imbalanced learning is related to rare class learning (rare classes, rare cases and absolute or relative rarity), see e.g., Weiss 2004; C. C. . Aggarwal 2014. The choice of appropriate imbalance treatment methods either depends on the kind of imbalance and also on the degree of imbalance. High degrees of imbalance, also called severe or extreme imbalance are even worse than a weak imbalance, see e.g., Attenberg and Ertekin 2013. A possible effect of the degree of imbalance is described in the following example 2.3.1.



(a) imbalance (b) complex imbalance

Figure 2.3: The figures show different categories of imbalance: (a) a data set with between-class imbalance, (b) a more complex data set with within-class imbalance, between-class imbalance, multiple concepts.

Example 2.3.1

A data set resulting from a problem with (a severe) intrinsic imbalance can contain equal numbers of examples of the minority and the majority class. In this case the data set is balanced but the classes are not equally represented in the data set and lead to worse classification results of the minority class examples. Energy unit operation schedules pose such problems.The class of feasible operation schedules is much smaller than the class of infeasible ones. For example the volume of the class of feasible operation schedules of a μCHP is smaller than 1% of the volume of the infeasible class, see Bremer et al. 2010.

Imbalance is not the only factor that hinders learning classifiers. Imbalance in combination with data complexity amplifies classifier performance deterioration, see Stefanowski 2016. Data set complexity refers mainly to rare subconcepts of the minority class (small disjuncts), overlapping classes, presence of outliers, rare instances or noise, see Sudjianto et al. 2010; Stefanowski 2016. Additionally the interaction of class imbalance with small disjuncts, rare cases, data duplication and overlapping classes was investigated in, Kotsiantis et al. 2006 and in López et al. 2013.

Imbalanced Classification Approaches

Imbalanced classification approaches comprise data-based approaches, algorithm-based approaches and special assessment metrics, see Kotsiantis et al. 2006; Weiss 2013; Menardi and Torelli 2014; Stefanowski 2016.

Data-based approaches are mainly sampling techniques e.g., undersampling, oversampling, hybrid techniques and feature selection and instance selection approaches, see e.g., Van Hulse et al. 2007; Guo et al. 2008; H. He and E. Garcia 2009; Hoens and Chawla 2013; Yin and Gai 2015. Sampling of severely imbalanced data sets was considered in Klement et al. 2011.

In literature many imbalanced classification algorithms have been proposed, see e.g., Van Hulse et al. 2007; H. He and E. Garcia 2009; Hoens and Chawla 2013; Yin and Gai 2015 and comparative experimental studies can be found e.g., in Van Hulse et al. 2007; G. Batista et al. 2012; S. Zhang et al. 2015. The most common algorithm-based approaches are one-class learning, see e.g., Mazhelis 2006; Zhuang and Dai 2006; H. He and E. Garcia 2009; Khan and Madden 2014 and see Sect. 2.3.5, ensemble-learning, see e.g., X.-Y. Liu and Zhou 2013, multi-task learning, see e.g., Yang et al. 2010; X. He et al. 2014, cost-sensitive learning and skew-insensitive learning, see e.g., Hoens and Chawla 2013 and active learning, see e.g., Attenberg and Ertekin 2013.

The evaluation of imbalanced classification requires special evaluation metrics, because common binary metrics do not considered the imbalance that distorts the meaning of precision for the minority and the majority class. Such evaluation metrics are e.g. sensitivity, specificity or the F-measure. An overview of different evaluation metrics for imbalanced learning can found e.g., in H. He and E. Garcia 2009; Sun et al. 2009; Hoens and Chawla 2013; Japkowicz 2013.

Regardless of the various different imbalance treatment methods on the data level, on the algorithm level and the assessment metrics, López et al. 2013; Stefanowski 2016 concluded in their studies on imbalance that data complexity might have a stronger influence than imbalance. They suggest a consideration of imbalance treatment in the context of further data properties and to use these data intrinsic properties.

Applicability to Flexibility Descriptions

For flexibility descriptions, data preprocessing methods are not appropriate, because of the severe imbalance between the feasible and the infeasible classes. So imbalance has to be treated with appropriate algorithms and assessment metrics. Since further data properties should be taken into account, possible algorithms can be restricted to a small selection. Due to the severe imbalance between the classes one-class classifiers are an appropriate choice, see H. He and E. Garcia 2009; Raskutti and Kowalczyk 2004 and ensemble methods, see Lin and Chen 2013. These methods can be used for flexibility descriptions in combination with respective assessment metrics.

2.3.4 High-dimensional and Imbalanced Learning

The combination of high-dimensionality and imbalance appears rather often in real world data sets, e.g., biomedical data sets. Both data properties hinder classification performance and a combination of both properties poses additional challenges, see Lusa and Blagus 2012; Bak 2015.

High-dimensional and Imbalanced Classification Approaches

High-dimensional and imbalanced classification tasks are treated with methods from both fields. The performance of different methods has been evaluated in different experimental studies, e.g., Blagus and Lusa 2010; Lusa and Blagus 2012; Lin and Chen 2013; Yin and Gai 2015; Bak 2015; Bak and Jensen 2016. Often feature selection and dimensionality reduction methods are applied, see e.g., Blagus and Lusa 2010; Yang et al. 2010; Shanab et al. 2011; Lusa and Blagus 2012; Lin and Chen 2013; López et al. 2013; Maldonado et al. 2014; Bak and Jensen 2016. Furthermore the papers, cited in this subsection, focus on various facets of imbalance in combination with high dimensionality, point out the strengths and weaknesses of different methods and discuss the effect of data intrinsic characteristics. Overall they all recommend a careful consideration of the classification task to choose an appropriate classifier.

Applicability to Flexibility Descriptions

Appropriate methods for flexibility descriptions are the ones already identified for high-dimensional classification in Sect. 2.3.2 and imbalanced classification, see Sect. 2.3.3. Methods from one-class learning, ensemble-learning and multi-task learning are promising.

2.3.5 One-class Learning

One-class classification (OCC) is also known as novelty detection, outlier detection, and concept learning, resulting from the different application fields. One-class classification aims at learning classifiers, when only examples of one class are available and the other class is sparsely sampled or no examples are available, see D. M. J. Tax 2001, 13 seqq. Bellinger et al. 2012; Khan and Madden 2014. The well sampled class is usually called target class and the other class is called outlier class. The main challenge in one-class classification is the trade of between missing target examples and classifying outlier examples as target examples, Bartkowiak 2010. According to an OCC taxonomy by Khan and Madden 2014 OCC problems are divided into the following three groups.

Availability of training data: One-class classifiers can be either build only from positive examples or from positive examples in combination with some negative examples or unlabeled data.

Methodology used: OCC algorithms are organized into SVM and non SVM algorithms. SVM-based algorithms form a group, due to their frequent application in different application fields, their advancements and their significance.

Application domain Application domains are divided into text/ document classification and other fields.

One-class Classification Approaches

OCC methods according to the availability of training data are often SVM-based. These and further approaches can be found e.g., in Khan and Madden 2014.

The second group of the taxonomy consists of the different methods used for one-class classification. OCC methods consist of boundary methods, density methods and reconstruction methods, see D. M. J. Tax 2001, 17 seq., 57 seqq. Density methods are build from density estimates of the target class, e.g., in Parzen density estimators and Gaussian models. Boundary methods aim at defining a boundary around the target class, like e.g., SVM-based methods and some knn-based methods. Reconstruction methods rely on prior knowledge and aim at a minimization of the reconstruction error of new examples. Such methods are e.g., some knn-based methods. Furthermore several hybrid classifiers have been proposed, like multi-task learning, ensemble methods or active learning, Yang et al. 2010; Krawczyk 2016; Barnabé-Lortie et al. 2015.

The third group of algorithms focuses on OCC with domain specific properties. For the application field time series classification e.g., SVM-based approaches or ensemble approaches have been proposed in Ding et al. 2008; Sachs et al. 2009; Bagnall, L. M. Davis, et al. 2012; Bagnall, J. Lines, et al. 2015.

Applicability to Flexibility Descriptions

Flexibility description classifiers are built from only feasible examples or feasible and some additional infeasible examples. The data sets are labeled correctly in both cases.

From the methodological point of view, boundary methods are promising as flexibility descriptions, because the learned decision boundary is a simple description of the set of the feasible operation schedules. The other methods do not yield a readily usable description. They define the class boundary in terms of target densities or reconstruction errors. From the field of boundary methods SVM-based methods are an obvious choice, because they are promising methods due to their significance and broad application field, see Khan and Madden 2014.

From the application domain, SVM-based methods and ensemble methods are promising

2.3.6 Result of the Literature Review

In literature there is no classifier available, that fits all identified classification task, see Tab. 2.1 and data properties, see Tab. 2.2. Most papers focus only on one or some-times on two of these properties at the same time. Overall one-class classification, especially boundary methods, multi-task learning and ensemble methods fit most identified properties. From the OCC methods SVM-based methods seem to be the most promising methods. But an SVDD approach has been applied in the flexibility description proposed in Bremer et al. 2010. This SVDD approach faces problems, describing sets of global operation schedules, see Sect. 1.4.4. Thus methods with more generalizability are required, probably from the field of ensemble learning. Ensemble methods promise generalizability and a good classification precision, see Rokach 2010; Bagnall, L. M. Davis, et al. 2012; Elish et al. 2013; Whalen and Pandey 2013; Jurek et al. 2014; Bagnall, J. Lines, et al. 2015.

But combinations of different data set properties with different classification task properties can amplify classification deterioration resulting from single properties, see Kotsiantis et al. 2006; López et al. 2013; Stefanowski 2016. Therefore López et al. 2013 and Stefanowski 2016 suggest to take all properties into consideration for the method choice. Additionally López et al. 2013 suggest to search for properties which can be exploited for a classification task simplification. Therefore a further classification task and data set analysis has to be done with regard to additional characteristic properties that can be exploited.

2.4 CLASSIFIERS APPLIED IN THIS THESIS

In this thesis different classifiers are applied. These classifiers are briefly introduced in the following. They are one-class classifiers and binary classifiers. The respective one-class classifiers are a one-class-SVM (OCSVM), a mixture of Gaussian models (MOG) and a nearest neighbor method called kmeans.

OCSVM and SVDD The one-class SVM, introduced by Schölkopf et al. 2001 is a boundary method. A hyperplane is computed that separates the interesting class from the origin with a maximum margin. If a linear separation does not yield a a good description of the interesting class, kernel functions can be applied. Kernel functions are used to transform the data set to a new data space, where the data is linearly separable.

The most common OCSVM parameters are the margin parameter ν and the kernel parameter γ. The margin parameter ν is an upper bound on the fraction

of training errors and a lower bound on the fraction of support vectors. The kernel parameter γ adjusts the kernel bandwidth.
Under certain conditions a similar classifier, the Support Vector Data Description (SVDD) yields the same results as an OCSVM. A SVDD yields the closest boundary around the interesting class. The description is achieved by describing the interesting class as a hypersphere in high-dimensional space and minimizing the radius of that hypersphere. The SVDD employs similar parameters as an OCSVM. The parameter C is comparable to the parameter ν of an OCSVM and γ controls again the kernel bandwidth. SVDD and OCSVM classifiers yield the same results when the data set is processed to unit norm, see the proof in D. M. J. Tax 2001. A comparison of both classifiers can be found in Zhuang and Dai 2006. Since OCSVM and SVDD can yield similar results, mainly OCSVMs are used in this thesis even for comparison to the flexibility description from Bremer et al. 2010 and Bremer 2015

MOG The mixture of Gaussian models is a density based method, proposed by Duda and Hart 1973. The interesting class is modeled with a mixture of k Gaussian models. Gaussian models describe data as a Gaussian distribution.
The main parameters to optimize in MOG modes are the error ϵ on the interesting class and the number of models k used to describe the interesting class. In general MOG models are suitable for less-dimensional spaces.

kmeans Kmeans is a density based method from Bishop 1995, that describes the interesting class with k clusters. These k clusters are placed in such a way that the average distance to a cluster center is minimized. The number of clusters k can be optimized as well as the error ϵ on the interesting class. New examples are characterized by their distance to the nearest cluster center. This method is more suitable for less dimensional space.

The employed binary classifiers are a common support vector machine (SVM) and a k-nearest neighbor classifier (knn).

SVM The common support vector machine is a boundary method introduced in Boser et al. 1992; Vapnik 1995. SVMs calculate a linear separation of both classes in high-dimensional space. If both classes are not linearly separable, kernel mappings can be applied to achieve linear separability. To optimize SVM results, the penalty parameter C of the error term and the kernel coefficient γ are usually optimized.

knn The k-nearest neighbors classifier is an instance based method introduced in Duda and Hart 1973. The class affiliation of new examples is computed from a majority vote of the k nearest neighbors of each example. Therefore the parameter k has to be optimized.
Many nearest neighbor methods like knn show problems with high-dimensional data sets due to computational complexity and a phenomenon called hubness, see Tomašev and Mladenić 2013. Hubness means training examples are not homogeneously distributed in space and form hubs. New examples in the vicinity of a hub are often assigned to the same class as the examples in the hub belong to. Therefore knn should be used in less-dimensional space or hub-insensitive variants can be applied in higher dimensions.

2.5 Classifier Performance Evaluation

The flexibility description classifier to be developed should be evaluated according to classifier performance and the applicability in VPP scheduling. The evaluation is divided into three parts: precision analysis, parameter sensitivity analysis and an analysis of the temporal complexity. Corresponding evaluation methods are presented in the following subsections for precision in Sect. 2.5.1, for parameter sensitivity in Sect. 2.5.2 and for temporal complexity in Sect. 2.5.3.

2.5.1 Precision

The precision of a classifier is usually assessed with the help of a test data set. The labels of the test set are known and they are compared to the predictions of the classifier on the test set. The true and the predicted labels are set into relation in a confusion matrix, e.g., in H. He and E. Garcia 2009; Lin and Chen 2013, see Fig. 2.4. The number of correctly predicted positive test examples is indicated as true positives tp and the number of correctly predicted negative test examples is indicated as true negatives tn. The numbers of incorrectly predicted test examples are indicated as false positives fp and false negatives fn.

		True class	
		positive	negative
Predicted class	positive	*tp* (true positive)	*fp* (false positive)
	negative	*fn* (false negative)	*tn* (true negative)

Figure 2.4: Confusion matrix.

Classifier precision can be quantified with different assessment metrics based on the entries of the confusion matrix (CM). Since nearly all terms describing the quality of classifiers like precision or accuracy are also names of assessment metrics, I would like to prevent misconception. I use the term precision in a general sense in the whole thesis and do not use the assessment metric *precision*, see e.g., H. He and E. Garcia 2009.

The most frequently applied metric is *accuracy*:

$$accuracy = \frac{tp + tn}{tp + fp + fn + tn}. \tag{2.1}$$

Beside *accuracy* several other assessment metrics have been proposed to evaluate common binary classifiers, imbalanced classifiers, one-class classifiers, etc., see e.g., D. M. J. Tax 2001; H. He and E. Garcia 2009; Sun et al. 2009; Hoens and Chawla 2013; Lin and Chen 2013; Japkowicz 2013.

Since operation schedule classification employs severely imbalanced classes, I prefer a separate classifier evaluation on the feasible and the infeasible class. A separate evaluate allows an interpretation of each class and the minority class is not biased by the majority class, see e.g., H. He and E. Garcia 2009; Japkowicz 2013. Therefore flexibility description classifiers will be evaluated according to the true positive rate (TP) also called *sensitivity* is given by

$$TP = \frac{tp}{tp + fn}. \tag{2.2}$$

and the true negative rate (TN) also called *specificity* is given by

$$TN = \frac{tn}{fp + tn}. \tag{2.3}$$

Most assessment metrics including the presented metrics do not take data quality into consideration, even though classification performance highly depends on the quality of the underlying distribution of the available examples of each class, Lin and Chen 2013. To take data quality into account, the distribution of training-, validation- and test data will be considered in the evaluation chapter.

Increasing Precision

Since the flexibility model should achieve a high precision for various applications and classification performance highly depends on the underlying data distribution, see Lin and Chen 2013, the flexibility model needs to adapt to different applications. Therefore I consider components that influence classification precision. Classification is mainly influenced by two components the given data set and the applied classifier with its parametrization. To achieve higher classification precision either the data set can be improved or the classifier or both, see e.g., Kotsiantis et al. 2006.

Improvement on the data level Precision improvement on the data level can be achieved with the following three possibilities:

- Usually an **increase of training examples** leads to an increased classification performance. But training examples can be limited and an increase of the number of training examples increases the computation time.
- **Data preprocessing** methods can be divided into two groups. The first group of data preprocessing methods consists of sampling methods for choosing training data, see e.g., H. Liu et al. 2001; Jankowski and Grochowski 2004; S. Garcia et al. 2012; Tsai et al. 2013; Blachnik 2014; Tomašev, Buza, et al. 2015 and the second group consists of an introduction of examples of the badly represented or hard to sample class, see e.g., D. M. J. Tax and Duin 2002; Bánhalmi et al. 2007. Additionally, infeasible examples can further improve the classification performance

by increasing the selectivity of the decision boundaries, Zhuang and Dai 2006.

- **Data transformations** originate from time series classification. Discriminatory features are not equally represented in different domains, e.g., time domain or the change domain, Bagnall, L. M. Davis, et al. 2012 and therefore classification in time domain does not necessarily yield the best results.

Improvement on the algorithm level Precision improvement on the algorithmic level depends on two factors:

- The **classifier choice** should incorporate the properties of the given data set. Further more a precision improvement can be achieved by using ensemble classifiers and multi-classifier systems instead of single classifiers at the cost of increasing computation costs, see e.g., Rokach 2010; Jurek et al. 2014; X.-Y. Liu and Zhou 2013; Ranawana and Palade 2006.
- A good classifier can perform badly without **parameter optimization**. Common techniques for parameter optimization are grid search, random search, e.g., Zhuang and Dai 2006; Bergstra and Bengio 2012 and evolutionary algorithm, e.g., Kramer 2014.

2.5.2 Parameter Sensitivity

Sensitivity analysis (SA), also known as elastic theory, response surface methodology or design of experiment, examines the response of model output parameters to input parameter variations, see e.g., Nguyen and Reiter 2015. Model outputs are sensitive to input parameters in the two following distinct ways, according to Hamby 1994.

- Sensitive input parameters are associated with a variability or uncertainty. The variability or uncertainty are propagated through the model and contribute to a great degree to the overall output variability.
- Significant changes in the model output can also result from small input parameter changes, if the model results are highly correlated with input parameters.

The goal of SA is a ranking of the input parameters in a "sensitivity ranking" according to their amount of influence on the model output. The amount of influence can be quantified with the help of sensitivity indices, Hamby 1994.

In literature numerous sensitivity analysis methods have been proposed and they are categorized according to different criteria. Such categorized focus e.g., on the methods, their purpose and sensitivity indices. In Frey et al. 2003 a categorization of SA methods is proposed with respect to the nature of the model to be examined. The categories are mathematical methods, statistical methods and graphical methods. Mathematical methods are used for deterministic and probabilistic models. Statistical methods are useful for probabilistic models and graphical methods are used for all kinds of models.

In Hamby 1994 three groups of different SA methods are distinguished. The first group comprises methods that 'operate on one variable at a time', the second group comprises methods that 'rely on the generation of an input matrix and an associated output vector'. The third group contains those methods that 'require a partitioning of an input vector based on the resulting output vector'. A categorization with some similarities was proposed by Heiselberg et al. 2009: local methods, global methods and screening methods. In local SA methods one parameter is varied, while all other parameters are held constant. Theses methods investigate the relative importance of different input parameters. Global SA methods analyze the influence of an input parameter, while varying all other parameters as well. Screening methods are used for SA analysis of high-dimensional and computational expensive models with various input parameters. Screening methods evaluate the sensitivity of input parameters in turn.

Since the flexibility description should be realized with a classification model, SA has to be conducted on a mathematical or statistical model. SA methods for mathematical and statistical models can be found e.g., in Hamby 1994; Frey et al. 2003; J. Wu et al. 2013; Kleijnen 2015; Borgonovo and Plischke 2016. Furthermore classification models are black-box models, which only represent the input output behavior and no functional relationships. Therefore also SA methods for black box models are of interest. In Cortez and M. Embrechts 2011; Cortez and M. J. Embrechts 2013 SA methods have been applied to data mining black-box models such as neural networks or SVMs or random forests. The SA results were used to increase model interpretability either by rule extraction or by visualization techniques.

In general a sensitivity analysis is conducted according to the following six steps, see Heiselberg et al. 2009. The execution scheme is independent from the applied SA method.

1. Formulation of a question to be answered and definition of output variable(s) to be analyzed.
2. Determination of input parameters, that should be included in the SA, e.g., with an initial screening analysis.
3. Determination of input parameter ranges and appropriate sampling methods according to the chosen SA method, e.g., equidistant or random sampling.
4. Creation of an input vector or matrix by sampling the input parameters.
5. Model output computation for the input vector or matrix from step 4.
6. Assessment of the influence of the input parameters on the model output e.g., with statistical tests or sensitivity indices. This step results in sensitivity ranking of the analyzed input parameters according to their influence on the model output.

The sensitivity analysis of the classifier to be developed can be done with respect to these six steps. The conduct of the SA will be done as roughly described below.

1. Which effect do the model parameters have on the classification precision or more precisely the true positive rate (TP) and the true negative rate (TN)?
2. Different classifier parameters are of interest.
3. Input parameter ranges and the respective resolution have be determined depending on the parameters.
4. Input vectors or matrices contain the values identifies in 3).
5. Computation of the classifier precision for the input vector or matrix from step 4.
6. This step is changed to an interpretation of the resulting precision. Based on the results model fitting advices are derived.

2.5.3 Temporal Complexity

Temporal complexity describes the asymptotic behavior of an algorithm, when the size of the input (N) goes to infinity, see e.g., Aho et al. 1983; Ottmann and Widemayer 2012; Knuth 1976. The runtime of an algorithm is indicated in terms of the number of elementary operation, e.g., functional calls, performed by the algorithm. Since the time of performance may vary for different inputs of the same size, temporal complexity is usually estimated for the worst case and the best case. If a functional description of the algorithm can be derived, the temporal complexity can be proved analytically. Otherwise the temporal complexity has to be estimate empirically based on simulation experiments for inputs of increasing size (N).

For the analytic complexity estimation a functional relation of the complexity is derived as a function of the problem size (N). From this functional relation asymptotic bounds can be derived. The asymptotic upper bound is indicated as $\mathcal{O}(N)$ and the asymptotic lower bound as $\Omega(N)$.

An empirical estimation of the temporal complexity is derived from experiments of increasing problem size (N). Since the temporal complexity is estimated as an asymptotic behavior, a consideration of large problems is necessary.

3 CLASSIFIER DEVELOPMENT FOR FLEXIBILITY DESCRIPTIONS

In the previous chapter, Chapt. 2 machine learning classifiers were identified as appropriate flexibility descriptions. The classification tasks of operation schedule classification are related to mainly three classification categories: time series classification, high-dimensional learning and imbalanced learning. There is no classifier in literature available that fits all these three properties. Additionally López et al. 2013 and Stefanowski 2016 suggest to consider the different classification task properties and data properties and to exploit these properties for classification task simplification. Based on these suggestions, this chapter starts with a further data property and classification task property analysis. Then appropriate characteristics are identified in Sect. 3.1 and used to develop a classification model. After that the classification model is presented stepwise in Sect. 3.2 - Sect. 3.5 and summarized with a process model in Sect. 3.6.

The classification model idea and the classification model were published in Neugebauer et al. 2015; Neugebauer et al. 2016; Neugebauer, Bremer, et al. 2016.

3.1 Problem Formulation and Classification Model Development

A further analysis of the classification task properties and data set properties was conducted with regard to time series properties. Operation schedules can theoretically employ values between 0% and 100% of the maximal power output or consumption for each time step. This means both classes (feasible and infeasible) fill together a hypervolume, in this case a hypercube. Most of the times the small feasible class is surrounded by the infeasible class.

Since time series classification tasks often show a high correlation between neighboring time steps, Sammut and Webb 2010; Bagnall, L. M. Davis, et al. 2012, the correlation behavior of operation schedules was analyzed with regard to autocorrelation and partial autocorrelation.

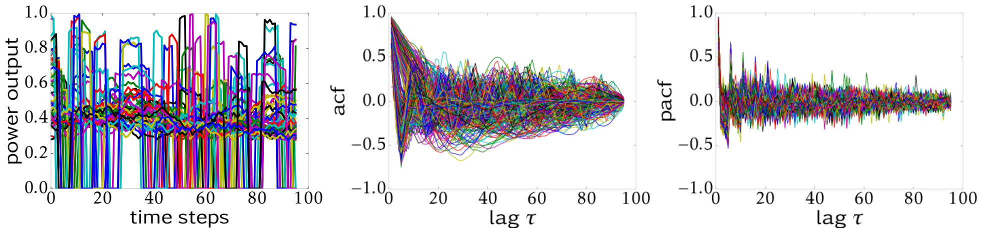
Autocorrelation The autocorrelation describes the linear dependence between two points in time of a time series. For time series with an underlying stationary process the autocorrelation between two arbitrary points depends only on the lag τ between those points. Weak stationarity is sufficient and can be detected as a constant expectation value and a constant autocovariance, because the process is time independent, see Hamilton 1994, p. 45. The autocorrelation $\hat{\rho}$, see Hamilton 1994, p. 110, of an observed time series x with T time steps and the expectation value $\bar{x} = \frac{1}{T}\sum_{t=1}^{T} x_t$ is estimated for the lag τ as

$$\hat{\rho_\tau} = \frac{\sum_{t=\tau+1}^{T}(x_t - \bar{x})(x_{t-\tau} - \bar{x})}{\sum_{t=1}^{T}(x_t - \bar{x})^2}. \tag{3.1}$$

Partial autocorrelation The partial autocorrelation of two points x_t and $x_{t-\tau}$ is the autocorrelation ρ_τ after removing the linear dependencies on $x_1, x_2, \ldots,$ $x_{t-\tau+1}$. Since the computation of the partial autocorrelation is elaborate, no formula is presented, but a detailed description can be found e.g., in Hamilton 1994, p. 111.

The analysis of different energy unit operation schedules of single μCHPs, heat pumps and coalitions of these energy units yielded a weak stationarity which allows the application of the autocorrelation function (acf) and the partial autocorrelation function (pacf) to these operation schedules. The correlation analysis revealed a high correlation between neighboring time steps or time steps with a short distance. Furthermore the correlation strongly decreases for time steps with larger distances, see Fig. 3.1 From these findings two classification task simplifications are derived.

1st Simplification Most information concerning the feasibility of a time step is encoded in its neighboring time steps. When most feasibility information is encoded in neighboring time steps, the high dimensional data set could be split into several low-dimensional data subsets of neighboring time steps without a



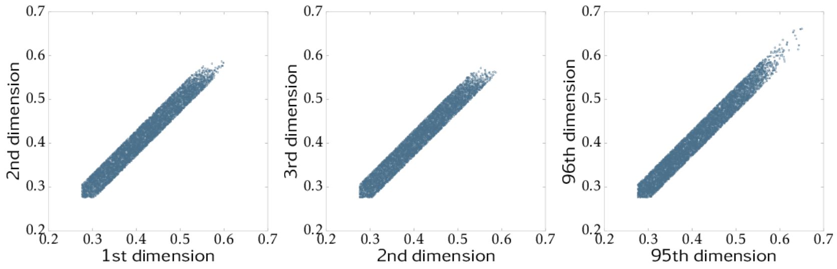
(a) CHP schedules (b) auto-correlation (c) partial auto-correlation

Figure 3.1: μCHP power production time series: (a) plot of 50 normalized operation schedules for $24\,h$ with $15\,min$ resolution ($d = 96$ steps), (b) auto-correlation plotted for 500 schedules, (c) partial auto-correlation plotted for 500 schedules.

great information loss. Splitting the high-dimensional classification task into several low-dimensional ones simplifies the original classification task. The new low-dimensional classification tasks are now only time series classification tasks with imbalanced classes. In literature there are several classifiers for imbalanced time series classification tasks, see e.g., Liang and C. Zhang 2012; Bagnall, L. M. Davis, et al. 2012; B. Fulcher and N. Jones 2014; J. A. Lines and Bagnall 2015; Bagnall, J. Lines, et al. 2015. The most promising classifiers for the low-dimensional classification tasks are one-class classifiers, because infeasible training examples are not always available.

2nd Simplification Low-dimensional data subsets with overlapping features (time steps) show a similar data structure, see Fig. 3.2. This observation can be used for additional classification task simplifications. The low-dimensional classification tasks can be solved with similar classifiers with similar parametrizations.

All classification task properties identified in the previous chapter, see Tab. 2.1 and this subsection are summed up in Tab. 3.1. The identified classification task properties describe the time series classification task of operation schedules in VPP scheduling. The classification model to be developed refers to classification tasks with these properties. Very important classification task properties are high-dimensionality, (severe) imbalance and the possibility to split the classification task without a great information loss. Additionally the classification task has to be binary, classes have to



(a) 1st, 2nd time step (b) 2nd, 3rd time step (c) 95th, 96th time step

Figure 3.2: 2-dimensional plots of the 1st and the 2nd (a), the 2nd and the 3rd (b) and the two last time steps (c) of μCHP power production time series.

be clearly separable and the small interesting class should preferably consist of only one easily learnable concept. Furthermore operation schedules are assumed to be correctly. All prediction inaccuracies in the operation schedules e.g., concerning the weather forecast or the thermal demand of a household are ignored in the following.

3.2 BASIC CASCADE CLASSIFICATION MODEL

The basic cascade classification model is based on the two classification task simplifications possibilities, identified in Sect. 3.1. The cascade classification model, see Neugebauer et al. 2015 is a step wise classifier, that splits high-dimensional classification tasks into a cascade of low-dimensional classifiers. These low-dimensional classifiers are each based on two neighboring time steps (features) with a feature overlap between neighboring classifiers. The cascade approach works as follows. Let $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$ be a training set of N time series $\mathbf{x}_i = (x_i^1, x_i^2, \ldots, x_i^d)^T \in \mathbb{R}^d$ of d time steps and their class labels $y_i \in \{+1, -1\}$. For each 2-dimensional training set

$$((x_1^j, x_1^{j+1}), y_1), \ldots, ((x_N^j, x_N^{j+1}), y_N) \tag{3.2}$$

with $j \in \{1, \ldots, d-1\}$, a classifier is trained with a feature overlap between the classifiers. All $d-1$ classification tasks can be solved with arbitrary baseline classifiers. As far as single classifiers employ similarly structured data spaces, the same baseline classifiers can be employed for all $d-1$ low-dimensional classification tasks and less

property	value
number of classes	2 (feasible, infeasible)
balance of classes	(severely) imbalanced (small feasible class)
dimensionality	high-dimensional (e.g., $24h$, $15\,min$ resolution)
feature values	numeric values (mainly scaled to $[0,1]$)
separability of classes	clearly separable (no class overlap)
fragmentation (feasible class)	one or more concepts
correlation (feasible class)	high for neighboring time steps (steep decrease)
data structure (feasible class)	similar in low-dimensional subsets (shape, number of concepts)
class structure	feasible class is most commonly surrounded by the infeasible class

Table 3.1: Complete classification task property table based on Tab. 2.1 and additional properties presented in Sect. 3.1.

effort is needed for parameter tuning. In most cases only feasible low-dimensional examples are available and therefore one-class baseline classifiers are suitable. The predictions $f_1, \dots, f_{d-1}$ of all $d-1$ classifiers are aggregated to a final result

$$F(\mathbf{t}) = \begin{cases} +1 & \text{if } f_i \neq -1 \ \forall i = 1, \dots, d-1 \\ -1 & \text{else} \end{cases} \tag{3.3}$$

for a new time series $\mathbf{t}$. A new time series $\mathbf{t}$ belongs to the small class, only if all classifiers in the cascade predict the small class for each time step. In energy time series tasks, the small class is the class of feasible time series.

3.3 ADAPTATION OF THE BASIC CASCADE CLASSIFIER TO GIVEN CLASSIFICATION TASKS

The cascade classification model can be easily modified. Since the cascade classification model was developed for a category of classification tasks, see Tab. 3.1 and should yield precise results, the cascade classification model needs to be adapted to new classification tasks. The main variations among classification tasks occur in the correlation behavior of the feasible class, the data structure of the feasible class (number and shape of concepts) and the availability of feasible and possibly also infeasible training examples. Classification task adaptations comprise adaptations of the dimensionality of the low-dimensional data subsets, a baseline classifier choice and an adaptation of the aggregation of intermediate classification results to an overall result. These adaptations were published in Neugebauer et al. 2015 and they are presented in the following subsections, see Sect. 3.3.1 - Sect. 3.3.3.

3.3.1 Classification Task Splitting (Projection): 2d -> pd

Depending on the geometry of the small class and the length of the highest correlation between time steps it is not always advantages to divide the data set into 2-dimensional subsets. Even though time series of the small and interesting class employ a high correlation between neighboring time steps, the highest correlation is not necessarily between two direct neighboring time steps i and $i+1$. Furthermore the correlation decrease with increasing distance between time steps can be fast within one time step or slower.

An adaptation to the correlation behavior decreases the information loss due to classification task splitting and increases the classification precision at the same time. Therefore high dimensional data sets can be split into p-dimensional data subsets with $p < d$. Classification tasks are split into low-dimensional ones with regard to the correlation length τ of the highest correlation and with regard to the correlation decrease. In comparison to the 2-dimensional-cascade, now $d-(p-1)$ classifiers $f_1,\ldots,f_{d-(p-1)}$, are trained and their predictions are again aggregated to a final result $F(\cdot)$. Each classifier f_j is trained with the pattern-label pairs,

$$((x_1^j,\ \ldots,\ x_1^{j+p-1}), y_1),\ldots,((x_N^j,\ \ldots,\ x_N^{j+p-1}), y_N),\ j=1,\ldots,d-1. \tag{3.4}$$

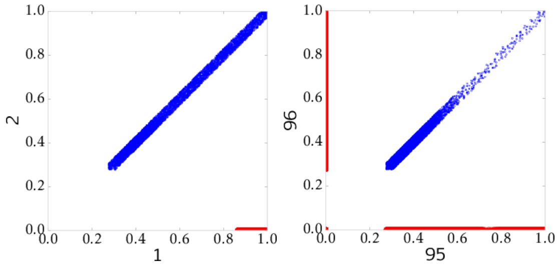
Depending on the classification task two neighboring low-dimensional data sets can overlap within a number of features between 1 and $p-1$. The greater the overlap, the more low-dimensional subsets are generated and the higher the computational effort. But greater overlaps include more information on the dependencies between different time steps and therefore they lead to more precise results.

Further suggestions for the adaptation of p are given in the evaluation chapter in Sect. 4.2.3.

3.3.2 Baseline Classifier Choice

Baseline classifiers are chosen according to the available training data and the data structure (number and shape of concepts) of the low-dimensional feasible class. Generally only the small and interesting class can be projected to low-dimensional space. In the case of operation schedules the feasible class is the small and interesting one. Infeasible schedules are infeasible, when at least on time step in the time series employs an infeasible value. Therefore low-dimensional infeasible examples could be found in the region of infeasible examples and also in the region of feasible examples. If no low-dimensional infeasible examples are available, classifiers have to be trained only on the feasible examples with one-class classifiers. If there were low-dimensional infeasible examples available, common binary classifiers could be applied, because pre-studies reveled a less severe imbalance of the low-dimensional classification tasks than in the high-dimensional classification tasks.

Beside the availability of training data, the data structure is important for the classifier choice. Preferably the feasible class employs only one concept and the chosen baseline classifier should be able to describe the feasible concept as good as possible. In the case of operation schedule classification, the feasible schedules can employ several concepts, resulting from the operation constraints of the energy units, e.g., CHPs, see Fig. 3.3. CHP operation schedules can consist of power production values > 0 and values $= 0$, where the energy unit is switched off.



(a) 1st, 2nd time step (b) 95th, 96th time step

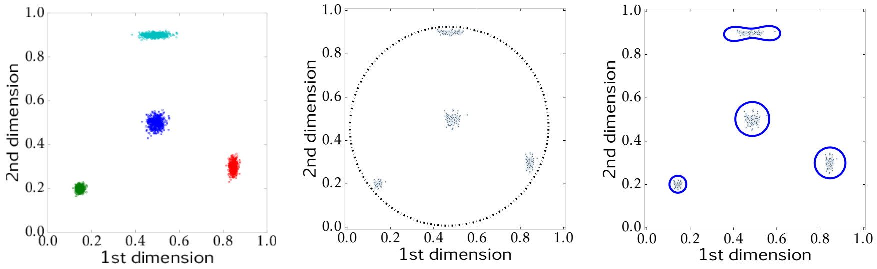
Figure 3.3: 2-dimensional plots of μCHP power production time series, exhibiting a concept in the middle and concepts on the axes.

If the feasible class employs more than one concepts, precision of the baseline classifier can be improved by learning each concept separately. This division of the low-dimensional classification tasks according to the concepts is called multi-task learning. In multi-task learning an appropriate baseline classifier is chosen for each concept. The application of multi-task learning in the cascade classification model is introduced in the following subsection.

Further suggestions for the baseline classifier choice with and without multi-task learning can be found in the evaluation chapter in Sect. 4.2.2.

Multi-Task Learning

Multi-task learning means learning each concept separately and merging the classification results to an overall result. Training one classifier for each concept leads to a higher classification precision than a single classifier, see Fig. 4.2. To achieve good classification results, each concept should present a typical cascade classification model task, see Tab. 3.1 Different multi-task learning methods have been proposed



(a) class with four concepts (b) OCSVM decision boundary (c) OCSVM with multi-task learning decision boundary

Figure 3.4: Binary classification example where one class consists of four separate concepts. a) All clusters are colored differently. b) Decision boundary learned with a OCSVM. c) Decision boundaries learned with separate OCSVMs for each concept.

e.g., for one-class learning, see Yang et al. 2010, high-dimensional data, see Seo and Oh 2013; X. He et al. 2014; Piao et al. 2014 or to time series classification, see Al-Hmouz et al. 2015. For the cascade classification model mainly multi-task learning for one-class is of interest.

Depending on the the separability of the concepts multi-task learning can be integrated into the cascade classification model in two ways.

- First case: The concepts are better separable in low-dimensional space. In this case the data set is split into p-dimensional data subsets as before. These p-dimensional data subsets are split based on knowledge about the concepts or based on a clustering analysis. In contrast to the basic cascade approach see Sect. 3.2 where one classifier is trained per cascade step, now one classifier is trained for each concept per cascade step. The results of the different classifiers on the test set are combined as before.
- Second case: The concepts are better separable in high-dimensional space. The high-dimensional data set is split into subsets according to the high-dimensional concepts. Each concept is learned with one cascade classifier like individual data sets. The classification results for all concept are aggregated by an aggregation of the confusion matrices of the classifiers of all concepts.

The multi-task approach is not easily applicable to classification tasks with overlapping and inhomogeneously sampled concepts. In such cases a clustering analysis, see e.g., Sim et al. 2013; Kang and Kim 2013; Bouveyron and Brunet-Saumard 2014 could be more helpful in describing all concepts separately.

3.3.3 Result Aggregation

The aggregation of intermediate classification results on the low-dimensional data subsets need to be aggregated to overall results. The employed aggregation scheme, see (3.3) can be adapted to given classification tasks, e.g., a majority vote might be useful in certain applications.

3.4 DATA PREPROCESSING

Beside algorithm adaptations also data improvements lead to an increased classification precision, see e.g., Kotsiantis et al. 2006. Classification precision depends strongly on the distribution of the underlying data set, see Lin and Chen 2013. Therefore, an improvement of the data distribution could improve classification precision.

Time series classification tasks with a cascade classifier have mainly two reasons for unfavorable data distributions. Beside the original often not homogeneous distribution of the time series examples in feature space, the projection of feasible time series leads to an inhomogeneous distribution in low-dimensional space. A selection of more homogeneously distributed feasible examples (instances) would lead to an improvement in classification precision for a constant number of training examples or decrease the number of training examples, that are necessary to achieve a certain classification precision. Depending on the data distribution and the application several instance selection (also called record reduction / numerosity reduction / prototype selection) approaches have been proposed in literature. Beside data compression and classification precision improvement, instance selection also works as noise filter and prototype selector, Blachnik 2014; Tsai et al. 2013; Wilson and Martinez 2000. In the last years, several instance selection approaches have been proposed and an overview can be found e.g., in S. Garcia et al. 2012; Jankowski and Grochowski 2004; H. Liu et al. 2001. Based on these algorithms advanced instance selection algorithms e.g based on ensembles, Blachnik 2014, genetic algorithms, Tsai et al. 2013 or instance selection for time series classification with hubs, Tomašev, Buza, et al. 2015 were developed. But all these instance selection approaches have more or less high computational complexity, because they are developed for d-dimensional data sets, while the cascade classifier has several similar structured data subsets in low-dimensional space. Therefore, we propose a simple and fast instance selection method for low-dimensional space.

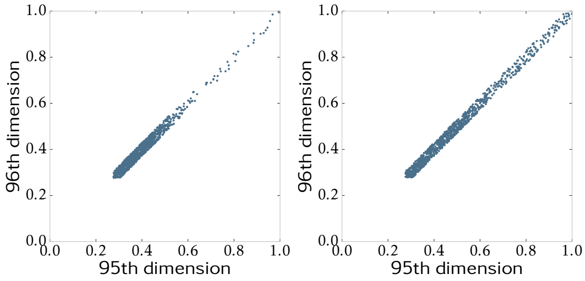
Additionally, infeasible examples can further improve the classification precision by increasing the selectivity of the decision boundaries, Zhuang and Dai 2006. If there are enough infeasible examples, binary classification can be applied and yield better results than one-class classification, see Bellinger et al. 2012. But even if there are infeasible examples available in high-dimensional space, they can not be used for training of the low-dimensional classifiers. Energy time series e.g., are only feasible, if all time steps are feasible. Due to this property infeasible power production time series projected to low-dimensional space can be located in the region of feasible ones. But generation of artificial infeasible examples can solve the problem of missing infeasible examples. Some algorithms have been proposed for sampling of infeasible examples. One such algorithm generates counter examples around the feasible class based on points near the class boundary, see Bánhalmi et al.

2007. Another algorithm presented in D. M. J. Tax and Duin 2002 can sample outliers from a hyperbox or a hypersphere, that cover the target object (feasible class). The artificial infeasible examples of these algorithms comprise either high computational complexity or contain some feasible examples. But the cascade classifier requires a fast and simple sampling approach for all low-dimensional data subsets, where the generated infeasible examples are located in the region of the infeasible class. Thus we propose an artificial outlier generation method for the data subsets of the cascade classifier.

The two data preprocessing algorithms for instance selection and outlier generation were proposed in Neugebauer et al. 2016 and further specified in **Neugebauer** Both algorithms are based on distances between nearest neighbors. They operate on the low dimensional training subsets, that fulfill the cascade model requirements. Both classes are clearly separable. The low-dimensional subsets incorporate similar structures in feature space and employ values in the same ranges for all time steps (features). For convenience all features are scaled, preferably to values between 0 and 1. Scaling of the features allows the use of the same parametrization for the data preprocessing methods for all low-dimensional subsets of the cascade classifier.

3.4.1 Selection of Feasible Examples

Selection of feasible examples is an instance selection method for the low-dimensional feasible training subsets of the cascade classifier. The goal is to achieve more representative training examples by homogenizing the point density of the training subsets, see Fig. 3.5. The example figures for selection of feasible examples show an increase in the point density in the upper right corner and a decrease in the point density in the lower left corner, see Fig. 3.5(b) in comparison to the original distribution shown in Fig. 3.5(a). Homogenization is achieved by selecting feasible examples for the training subsets based on the distance to the nearest feasible neighbors. Therefore a large set of feasible examples is needed, from which representative examples can be chosen. We assume that the inhomogeneous distribution of the training examples and their rarity in some regions is due to relative rarity. Relative rarity means examples are observed (sampled) less frequently than others, see e.g., H. He and E. Garcia 2009. But the rare examples constitute a certain percentage of a data set and an increase of the number of examples in the data set increases the absolute number



(a) Initial distribution (b) Resampled features

Figure 3.5: 1000 examples of the $95th$ and $96th$ dimension of the feasible class of the μCHP data set: a)initial distribution, b) resampled distribution

of rare examples. If the rarity would be an absolute rarity, the absolute number of rare examples could not be increased with an increase of examples in the data set, see e.g., H. He and E. Garcia 2009. For this reason selection of feasible examples can only increase homogeneity of training examples, if rarity is relative. Based on the data properties resulting from the cascade classifier and the above described requirements, selection of sn feasible examples works as follows for each low-dimensional training subset, see Algorithm 1.

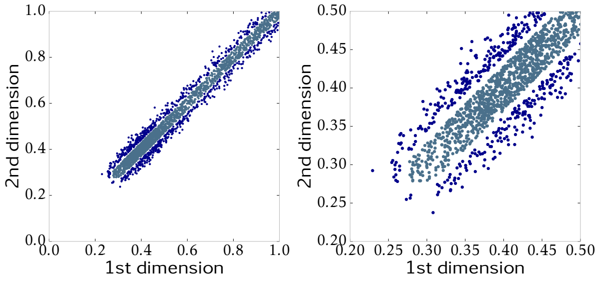
Algorithm 1 Selection of feasible examples

```
Require: 2-dimensional data set $\mathbf{X}$ with $n$ feasible examples
choose $t$ start examples $\mathbf{S}$ from $\mathbf{X}$
repeat
    choose $t$ new examples $\mathbf{E}$ from $\mathbf{X}$
    calculate euclidean distance $\delta$ of the examples in $\mathbf{E}$ to their nearest neighbors
    in $\mathbf{S}$
    if $\delta \geq \epsilon$ then
        append respective examples to $\mathbf{S}$
    end if
until all $n$ examples are processed
shuffle $\mathbf{S}$
```

The distribution of the feasible examples can differ a little in homogeneity and shape among the 2-dimensional data subset despite previous scaling. Therefore the parameters of the procedure have to be adapted carefully, especially the minimum distance ϵ of new examples (E) to the nearest selected neighbors (S). Preferably, ϵ is selected in such a way, that the selection of feasible examples yields round about the number of examples required for training and validation. Such ϵ values yielded in pre-tests good classification results, because the resampled data sets maintain the integrity of the original data subsets best for the desired number of training and validation examples. Further parametrization suggestions can be found in the evaluation chapter in Sect. 4.3.2.

3.4.2 Generation of Artificial Infeasible Examples

Sampling of infeasible examples near the class boundary is an outlier generation algorithm for low-dimensional space. The aim of this algorithm is the generation of low-dimensional infeasible examples near the true class boundary as additional training and or validation examples, see Fig. 3.6. Low-dimensional infeasible examples are generated at a certain distance to their nearest feasible neighbor. Due to this distance dependence to the feasible class, examples of the feasible class are required. These examples have to represent the feasible class as good as possible and furthermore they have to be distributed more or less homogeneously. Sampling of



(a) Fig. 3.5(b) with artificial infeasibles

(b) detail view of (a)

Figure 3.6: 1000 examples of the $95th$ and $96th$ dimension of the feasible class of the μCHP data set and 1000 artificial infeasible examples: a) resampled distribution surrounded by artificial infeasible examples (dark blue points), b) detail view of a.

infeasible examples strongly relies on the cascade classification model requirement of clearly separable classes. Additionally the class boundaries should be clear lines in low-dimensional space. With consideration of these requirements, generation of artificial infeasible examples near the class boundaries can be applied as a second data preprocessing method after selection of feasible examples. The algorithm works as described in Algorithm 2 for all low-dimensional training subsets of the cascade classifier. The low-dimensional feasible examples X are perturbed with Gaussian noise $\mathcal{N}(\mu,\sigma)\cdot\alpha$ and yield a new data set Y. Then the distance between the examples in Y and their nearest feasible neighbors in X is computed. A value in Y belongs to the set of artificial infeasible examples Γ if the distance to the nearest feasible neighbor δ_b is larger than or equals a certain value ϵ_b. To receive enough infeasible examples around the feasible class, the above described procedure is repeated with a perturbation of all examples in Γ instead of the examples in X until the set Γ contains a sufficient number of examples. The algorithm employs the parameter for the minimal distance between infeasible examples and their nearest feasible neighbors ϵ_b and the Gaussian distribution $\mathcal{N}(\mu,\sigma)\cdot\alpha$. The parameters depend on the distri-

Algorithm 2 Sampling of infeasible examples

```
Require: 2-dimensional data set $\mathbf{X}$ with $n$ feasible examples, where the distance between infeasibles and their feasible nearest neighbors $\delta_b \gg \epsilon_b$ in about 95% of all cases
$\mathbf{Y} = \mathbf{X} + \mathcal{N}(\mu,\sigma)\cdot\alpha$
calculate euclidean distance $\delta_b$ of all examples in $\mathbf{Y}$ to their nearest feasible neighbors in $\mathbf{X}$
if $\delta_b \geq \epsilon_b$ then
    append example to $(\boldsymbol{\Gamma})$
end if
repeat
    $\mathbf{Y} = \boldsymbol{\Gamma} + \mathcal{N}(\mu,\sigma)\cdot\alpha$
    calculate euclidean distance $\delta_b$ of all examples in $\mathbf{Y}$ to their nearest neighbors in $\mathbf{X}$
    if $\delta_b \geq \epsilon_b$ then
        append example to $\boldsymbol{\Gamma}$
    end if
until number of examples in $\boldsymbol{\Gamma}$ is sufficient
shuffle $\boldsymbol{\Gamma}$
```

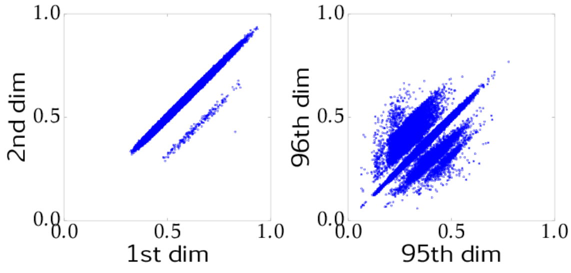
bution of feasible examples, mainly the distance between feasible nearest neighbors ϵ. Therefore ϵ_b has to be chosen carefully in such a way, that $\epsilon_b \gg \epsilon$ form the instance selection algorithm. The ϵ_b value should be at least so high, that at least 95% off all generated artificial infeasible examples lie outside the region of the feasible class. As far as the true class boundary is not known, the percentage of real feasible examples among the artificial infeasible ones has to be approximated. If the distance δ_b between generated infeasible examples and their nearest feasible neighbors is $\delta_b \gg \epsilon_b$ for at least 95% of all generated infeasible examples, then most of the generated infeasible examples are actually infeasible ones. The approximation relies on the requirement, that the examples of the feasible class are representatively and homogeneously distributed. But data sets usually employ less representative and less homogeneous data distributions. In such cases a more convenient data distribution can be achieved with the proposed algorithm for selection of feasible examples, see Algorithm 1.

The closer the infeasible examples are located to the class boundary, the greater is the improvement of classification specificity. But the closer the infeasible examples are located to the class boundary, the higher is the probability, that these artificial infeasible examples could be located in the region of the feasible class. False artificial infeasible examples can hamper classification improvement. Therefore a careful parametrization of the algorithm is necessary. All in all the minimum distance ϵ_b between infeasible examples and their nearest feasible neighbors should be as small as possible and as large as necessary. For further parametrization suggestions see the evaluation chapter, Sect. 4.3.2.

Noise for the generation of potentially infeasible examples should scatter in all directions without a drift. Therefore the Gaussian distribution is chosen with a mean value of $\mu = 0$. The larger ϵ_b the larger may be the standard deviation σ. A good initial choice is $\sigma = 0.01$. The range in which perturbed values can be found can be stretched with the factor α. The default value is $\alpha = 1$.

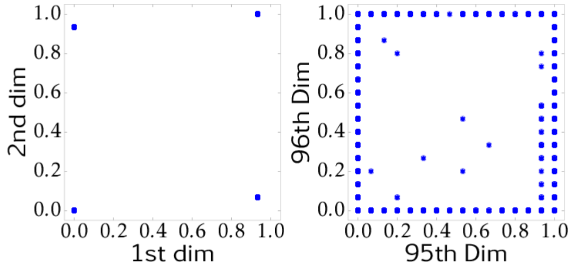
3.5 GENERALIZED CASCADE CLASSIFICATION MODEL

The presented adaptations for the basic cascade classification model in Sect. 3.3 and data preprocessing, in Sect. 3.4 enable high classification precision, but they have problems in handling inconvenient data structures, like e.g., see Fig. 3.7 or Fig. 3.8. The basic cascade classifier requires classification tasks with two clearly separable



(a) 1st, 2nd dimension (b) 95th, 96th dimension

Figure 3.7: 2-dimensional plots of the unfavorable data structure of feasible global operation schedules of 5 μCHPs.



(a) 1st, 2nd dimension (b) heat pump

Figure 3.8: 2-dimensional plots of the unfavorable data structure of feasible heat pump operation schedules.

classes, where the small and interesting class consists preferably of only one concept and has clear boundaries. Furthermore the small interesting class should be surrounded by the other class. These requirements enable precise decision boundaries on the low-dimensional subsets of the baseline classifiers. Violations of these requirements lead to imprecise decision boundaries. Classification errors on the low-dimensional data sets accumulate in the overall classification result.

High-dimensional classification tasks with imbalanced classes, which do not fulfill these requirements are e.g., global operation schedules of coalitions of μCHPs or heat pump operation schedules. Global operation schedules of a coalition consisting of μCHPs, see Fig. 3.7 employ different data structures in the low-dimensional subsets. The data subsets consists of different numbers of concepts with different shapes. A further example of inconvenient data structures are heat pump operation schedules, see Fig. 3.8. In low-dimensional heat pump data subsets, the small and interesting class surrounds the large infeasible class. Furthermore the feasible class employs several very small concepts.

A generalization of the basic cascade model can overcome the problems resulting from inconvenient data structures. Generalization can be achieved by ensemble approaches. Ensemble approaches promise a higher classification precision than single classifiers. The advantage of ensembles over single classifiers is due to the averaging of better and worse classifiers in the ensemble. Among the various ensemble classifiers presented in literature see e.g., Rokach 2010; Elish et al. 2013; Whalen and Pandey 2013; Jurek et al. 2014, there is one for common time series classification tasks, called *transformation based ensembles*, see Bagnall, L. M. Davis, et al. 2012. The ensembles are built on feature transformations of the original time series, because discriminatory features are more or less distinct in different time series representations. The *transformation based ensembles* classifier has been extended to a *collective of transformation based ensembles* (COTE) in Bagnall, J. Lines, et al. 2015, where a collective of classifiers is built on each transformation in the ensemble. Bagnall *et al.* showed in Bagnall, J. Lines, et al. 2015 that COTE outperformed most of the common classifiers on different time series classification tasks.

Transformation based ensembles and COTE work both with common binary classifiers, while the cascade classification model deals with high-dimensional time series classification tasks with severely imbalanced classes. The latter require special classifiers, because common binary classifiers cannot handle these data properties.

With respect to the *transformation based ensembles* classifier, both the introduction of ensembles and the transformation of the time series are promising ideas to generalize the cascade classification model. Ensembles promise higher classification accuracy, while time series transformations can lead to time series representations with data structures that can be classified more easily. The *transformation based ensembles* classifier is introduced in the following subsection, Sect. 3.5.1. A Generalization of the cascade classifier is a chived by building the *transformation based* *ensembles* classifier around the basic cascade classification model in Sect. 3.5.2.

3.5.1 Transformation-Based Ensembles

The *transformation based ensembles* classifier was developed to improve time series classification precision on various time series classification tasks compared to the precision of a single classifier, see Bagnall, L. M. Davis, et al. 2012. The approach achieves heterogeneity among the ensemble members by applying time series transformations. As far as discriminatory features are more or less distinct in different time series representations, classification in the time domain does not necessarily yield good results. Similarity among time series can be observed in time, in shape and in change domain, see Bagnall, L. M. Davis, et al. 2012. Therefore the training set time series ($\mathbf{X}$) from the temporal representation ($\mathbf{X} = \widehat{\mathbf{X}}_1$), are transformed with a Fourier transformation ($\mathit{fft}(\mathbf{X}) = \widehat{\mathbf{X}}_2$), that represents the spectral domain. Additionally they are transformed with the autocorrelation function ($\mathit{acf}(\mathbf{X}) = \widehat{\mathbf{X}}_3$), that represents similarity in change and a further transformation, principle component analysis (*PCA*) that yields the principle components of the time series ($\mathit{PCA}(\mathbf{X}) = \widehat{\mathbf{X}}_4$). A separate classifier is built on each data set ($\widehat{\mathbf{X}}_k, k \in \{1, \ldots, 4\}$). In Bagnall, L. M. Davis, et al. 2012, binary nearest neighbor (NN) classifiers (1-NN with euclidean distance and 1-NN with dynamic time warping (DTW)) are applied to all data sets $\widehat{\mathbf{X}}_k$ due to their good performance in time series classification tasks. New test instances are classified by all ensemble members and their predictions are combined with a weighting scheme, equal weighting or weighting based on cross-validation accuracy.

Transformation based ensembles have been extended to a collective of transformation-based ensembles (COTE) in Bagnall, J. Lines, et al. 2015. In contrast to the *transformation based ensembles* model, COTE employs several different classifiers on each transformed data set $\widehat{\mathbf{X}}_k$ instead of a single classifier. COTE enables different aggregation and weighting possibilities for the predictions of new instances and achieved a greater classification accuracy than common classifiers and the *transformation based ensemble* classifier on many different time series data sets, see Bagnall, J. Lines, et al. 2015.

As a first step only *transformation based ensembles* and not COTE is considered to generalize the cascade classification model.

3.5.2 Generalized Cascade Classification Model

The nearest neighbor classifier in the *transformation based ensembles* approach is replaced with the cascade classification model because nearest neighbor classifiers suffer from hubness. Hubness makes distinction between relevant and irrelevant points difficult, see Tomašev and Mladenić 2013; Tomašev, Buza, et al. 2015 and is even severe for imbalanced classes, see Radovanović et al. 2010. Beside the classifier exchange some further adjustments have to be conducted.

First of all s appropriate time series transformations have to be chosen, that represent preferably each of the three domains, time, shape and change. The cascade classifier requires such transformations, where the small interesting class in the low-dimensional training data subsets forms one cluster, that is well separable from the other class and can be learned easily by a baseline classifier.

Assuming there are s transformations, that fulfill the requirements and comprise at least one transformation from time ($\widehat{\mathbf{X}}_1$), shape ($\widehat{\mathbf{X}}_2$) and change domain ($\widehat{\mathbf{X}}_3$), the training set ($\mathbf{X}$) and the validation data set are transformed into the s representations. In the following one transformation is utilized from the three domains, $s = 3$. After that a cascade classifier is built from each transformed data set $\widehat{\mathbf{X}}_k$, $k \in \{1, 2, 3\}$ and the respective validation set. These classifiers yield predictions for new time series, transformed into the s different representations. The predictions of the low dimensional classifiers of the s cascade models allow different aggregations to overall classification results. Two different aggregation schemes can be applied. The first aggregation scheme deals with the results of each cascade classifier, while the second aggregation scheme deals with the intermediate results of the low-dimensional classifiers of the cascade classifiers. The first aggregation scheme aggregates the predictions of each cascade classifier $\widehat{F}_k$ for a new time series t with weightings w_k to an overall result $\mathbb{G}(t) = g(\widehat{F}_k, w_k)$, see Fig. 3.9(a).

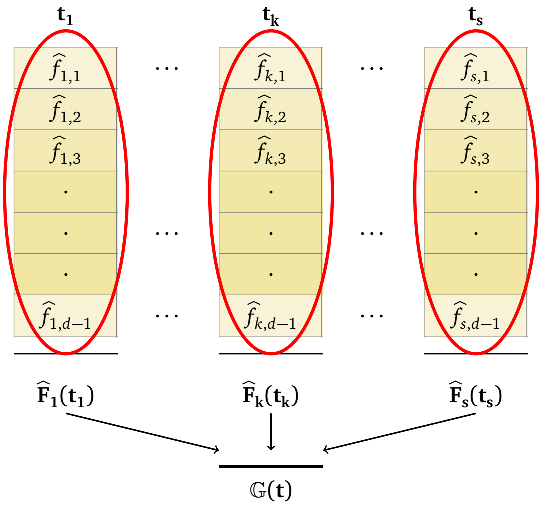
The second aggregation scheme operates on the predictions $\widehat{f}_{k,j}$ with $j \in \{1, \ldots, d-1\}$ for each 2-dimensional classifier of each of the cascade classifiers, see Fig. 3.9(b). All s predictions $\widehat{f}_{k,j}$ with $k \in 1, 2, \ldots, s$ are aggregated to a new prediction $\widehat{f}_j(t) = h(\widehat{f}_{k,j}, w_{k,j})$ with the weights $w_{k,j}$. This aggregation yields a prediction for each low-dimensional segment of the new time series t. These intermediate results are similar to the result of one ordinary cascade classifier and they are aggregated to an overall result $\mathbb{H}(t)$ like the intermediate predictions of the ordinary basic classifier, see (3.3).

Both aggregation schemes can employ different weightings, like e.g., equal weighting or validation performance based weighting. Equal weighting means, depending on the aggregation scheme the predictions of either the cascade classifiers or the small classifiers get the same weight $w = 1/s$. Validation performance based weighting is based on the validation performance α of each cascade classifier c_k (weight w_k), (3.5) for aggregation Scheme 1

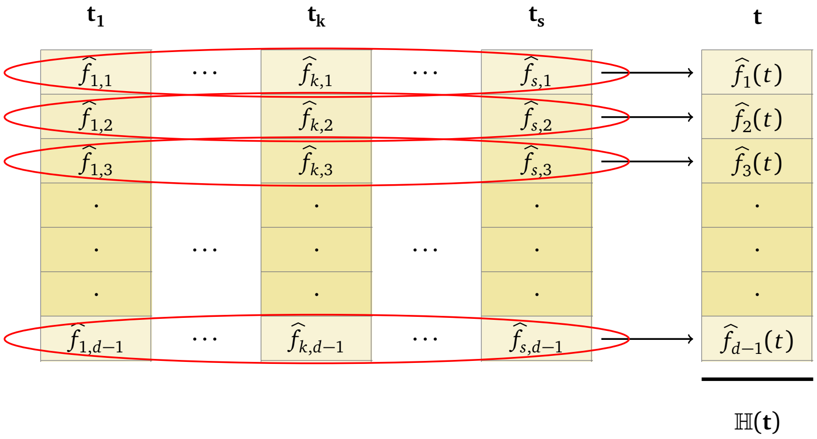
$$w_k = \frac{\alpha(c_k)}{\sum\limits_{k=1}^{s}(\alpha(c_k))} \tag{3.5}$$

or each of the small classifiers $c_{k,j}$ (weight $w_{k,j}$), (3.6) for aggregation Scheme 2.

$$w_{k,j} = \frac{\alpha(c_{k,j})}{\sum\limits_{k=1}^{s}(\alpha(c_{k,j}))} \tag{3.6}$$



(a) aggregation Scheme 1



(b) aggregation Scheme 2

Figure 3.9: Aggregation schemes for the generalized cascade classification model presented for one test schedule t with d time steps and $d-1$ low-dimensional classifiers and their predictions $\widehat{f}_{k,j}$, $j \in 1, \ldots, d-1$. The ensemble consists of s cascade classifiers.

3.6 SUMMARY

Since the cascade classification model was developed for a category of classification tasks, see Tab. 3.1 and should yield precise results for different classification tasks, the model needs to be adaptable.

The cascade classification model was proposed in a basic version in Sect. 3.2. Adaptations to given data sets and the inherent data structure of the feasible class were presented in Sect. 3.3 and Sect. 3.4. Furthermore a generalized ensemble version of the cascade classification model was proposed that can handle complex data structures.

In Fig. 3.10 a process model for the application of the cascade classification model is shown and suggests when to apply which adaptation. Cascade classification model parameters are listed in Tab. 3.2. Further application and parametrization suggestions can be found in the evaluation chapter, see Chapt. 4.

parameter	parameter range	
Model Fitting		
dimensionality	$2 \leq p < d$	see Sect. 4.2.3
baseline classifier	OCSVM, knn, etc.	see Sect. 4.2.2
baseline classifier parametrization	$2 \leq p < d$	see Sect. 4.3.1
Data Preprocessing		
minimal distance between feasibles	$\epsilon_b > 0$	see Sect. 4.3.2
minimal distance between feasibles and infeasibles	$\epsilon_b > \epsilon$	see Sect. 4.3.2
gaussian distribution	$\mathcal{N}(\mu, \sigma) \cdot \alpha$ ($\mu = 0$, $\sigma > 0$, $\alpha > 0$)	

Table 3.2: Cascade classification model parameters.

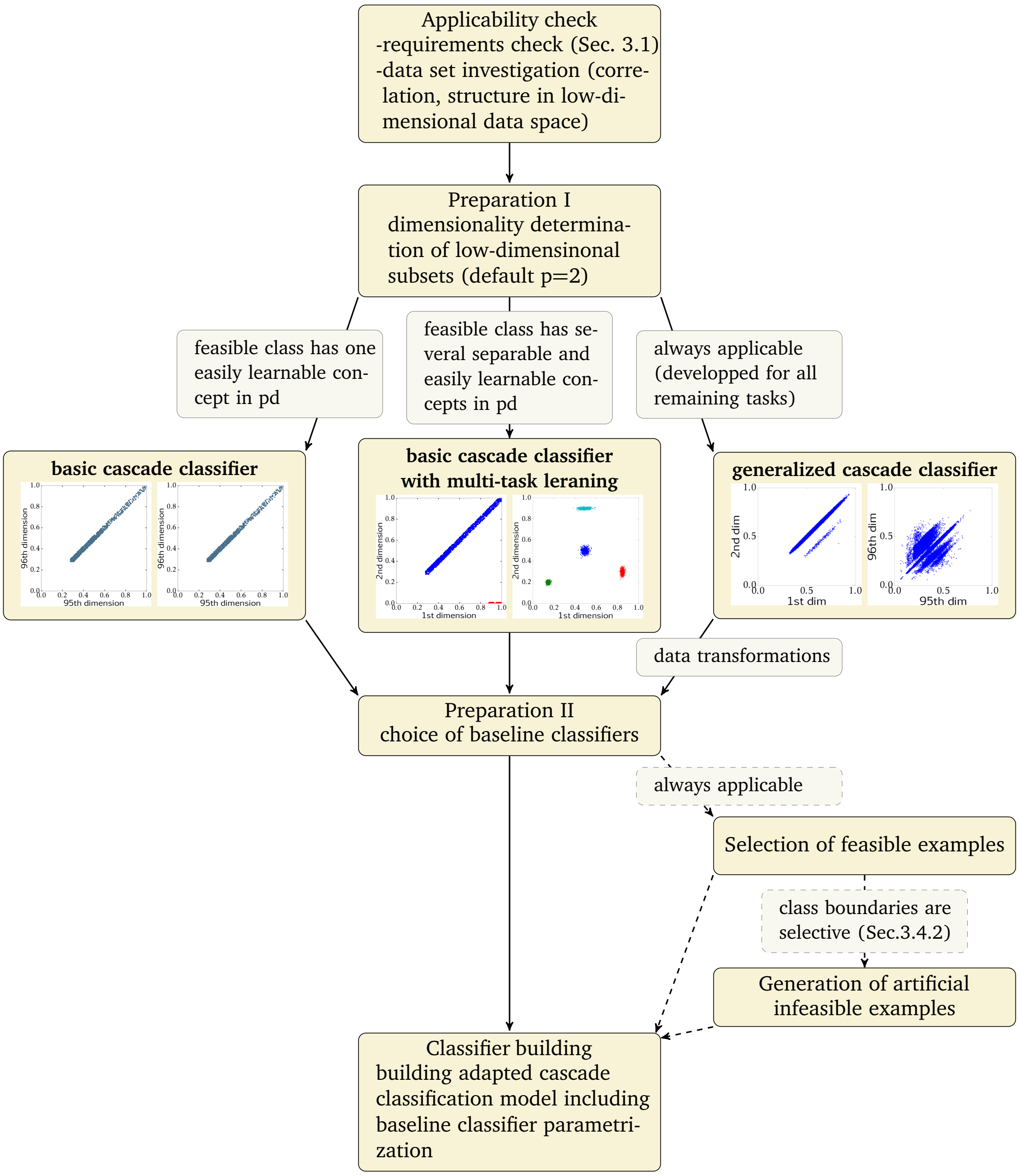

Applicability check
-requirements check (Sec. 3.1)
-data set investigation (correlation, structure in low-dimensional data space)
Preparation I
dimensionality determination of low-dimensinonal subsets (default p=2)
feasible class has one easily learnable concept in pd
feasible class has several separable and easily learnable concepts in pd
always applicable (developped for all remaining tasks)
basic cascade classifier
96th dimension
95th dimension
basic cascade classifier with multi-task leraning
2nd dimension
1st dimension
generalized cascade classifier
2nd dim
1st dim
96th dim
95th dim
data transformations
Preparation II
choice of baseline classifiers
always applicable
Selection of feasible examples
class boundaries are selective (Sec.3.4.2)
Generation of artificial infeasible examples
Classifier building
building adapted cascade classification model including baseline classifier parametrization

Figure 3.10: Process model: application and adaptation of the cascade classification model.

4 EVALUATION OF THE CASCADE CLASSIFICATION MODEL

The developed cascade classification model is an adaptable classifier, that can be used as a search space description in the optimization in VPP scheduling. The search space description is here also called flexibility description. Since this flexibility description should be abstract, consistent and precise, the cascade classification model is evaluated according to the classifier performance on energy time series of energy units and artificial data sets. The classifier performance is considered with respect to precision, parameter sensitivity and temporal complexity. Then model fitting and parametrization suggestions are derived from the evaluation results. Furthermore the cascade classification model is compared to a SVDD used in the flexibility description in Bremer 2015 and a common one-class SVM (OCSVM) which is comparable to the SVDD.

Some of the evaluation results are published in Neugebauer et al. 2015; Neugebauer et al. 2016; Neugebauer, Bremer, et al. 2016; Neugebauer et al. 2017. This chapter consists of a description of the basics of the evaluation experiments in Sect. 4.1, the evaluation of the classifier precision in Sect. 4.2, the parameter sensitivity in Sect. 4.3, the temporal complexity evaluation in Sect. 4.4 and a summary of the evaluation results in Sect. 4.5.

4.1 Basics of the Evaluation Experiments

The influence of model adaptations and model parameters on the model behavior strongly depend on the given classification task properties and the data set properties. Therefore different evaluation scenarios are built from energy time series- and artificial data sets with different data structures and different data set complexity. The data sets are introduced in Sect. 4.1.1 and the evaluation scenarios in Sect. 4.1.2.

The experimental setup of the evaluation experiments is presented in Sect. 4.1.3, the employed baseline classifiers and their default parametrization in Sect. 4.1.4, experimental evaluation in Sect. 4.1.5 and implementation details and pseudocode in Sect. 4.1.6.

4.1.1 Data

Data sets for the evaluation experiments are energy data sets from the application domain and artificial data sets with special properties to test the cascade classification model behavior. In the following the different data set are briefly introduced with regard to data set generation and their properties.

The energy data sets comprise operation schedules of single energy units (μCHP, heat pump) and global operation schedules of coalitions of energy units (homogeneous μCHP coalitions, mixed coalitions).

The artificial data sets comprise a *Hypersphere* and a *Hyperbanana* data set. The *Hypersphere* data set is used due to its correlation behavior that is different from the energy time series correlation behavior. The *Hyperbanana* data set is assumed to yield representative results, because banana shaped classes are difficult to learn. For this reason data sets with banana shaped classes are often used to test new classifiers. Furthermore the *Hyperbanana* data set is used in this thesis to test the selectivity of the cascade model, because infeasible examples are available for different distances to the class boundary. Altogether two *Hyperbanana* data sets are used, which are based on the same classes, but with different distributions of the feasible and infeasible examples.

All data sets, energy data sets and artificial data sets are simulated with models and they are no real measurements. Simulated data sets are advantages for the classifier evaluation, because arbitrary numbers of examples can be generated and data set properties like the ratio of generated feasible and infeasible examples can be influenced. Furthermore the distribution of the generated examples can be influenced by the choice of a sampling procedure. The most important sampling requirement is to generate feasible, respectively infeasible examples that represent the volume of the whole class. Additionally it would be advantages to have representative samples that are more or less homogeneously distributed in the volume of the respective class. To achieve these sample requirements different sampling, respectively data set

Property	Characteristic
classification task properties:	
greatest correlation length	2
correlation decrease	rapid decrease (within one time step)
balance of classes	feasible $\ll$ infeasible
data structure:	
number of feasible concepts	2 (middle- + edge concept)
shape of concepts	smooth boundaries, different shapes
data properties:	
feasible examples	$1,100,000$
infeasible examples	$1,100,000$
distribution:	
feasibles	representative, not homogeneous
infeasibles	representative, homogeneous (hardly any examples near the class boundary, due to the severely imbalanced classes)

Table 4.1: Overview of the μCHP classification task and data set properties

generation techniques are used to generate the different data sets.

The energy time series data sets are generated from energy unit simulation models of single energy units. Feasible global operation schedules are derived from single feasible operation schedules of all coalition members. Infeasible global operation schedules have to be estimated with a heuristic. Artificial data sets are sampled from functional descriptions. In contrast to the energy unit simulation models, the functional descriptions allow a specific generation of infeasible examples near the class boundary. These infeasible examples help to test the classifier selectivity.

In the following, all data sets and their modifications used for the evaluation scenarios are briefly introduced. The data sets of single energy units are presented in Tab. 4.1 and Tab. 4.2. Since energy unit coalition data sets of homogeneous μCHPs and mixed coalitions of μCHPs and heat pumps employ similar properties independent of the number of coalition members, only homogeneous μCHP coalitions will be considered and their properties are presented in Tab. 4.3. After that the artificial data sets are presented in Tab. 4.4 and Tab. 4.5.

Property	Characteristic
classification task properties:	
greatest correlation length	2 (3 also high)
correlation decrease	fast decrease (within two time steps)
balance of classes	feasible $\ll$ infeasible
data structure:	
number of feasible concepts	1 (surrounds the infeasible class)
shape of concepts	non smooth boundaries, not easy to learn
data properties:	
feasible examples	$1,000,000$
infeasible examples	$1,000,000$
distribution:	
feasibles	not representative, not homogeneous (due to the simulation model sampling procedure)
infeasibles	representative, homogeneous

Table 4.2: Overview of the *Heatpump* classification task and data set properties

Property	Characteristic
classification task properties:	
greatest correlation length	2
correlation decrease	rapid decrease (within one time step)
balance of classes	feasible $\ll$ infeasible
data structure:	
number of feasible concepts	several ones
shape of concepts	no smooth boundaries, different shapes
data properties:	
feasible examples	$30,000$
infeasible examples	$30,000$
distribution:	
feasibles	representative, more or less homogeneous
infeasibles	representative, homogeneous (imbalanced classes: hardly any examples near the class boundaries)

Table 4.3: Overview of the μCHP coalition classification tasks and data set properties.

Property	Characteristic
classification task properties	
greatest correlation length	low correlation between all dimensions
correlation decrease	no decrease
balance of classes	feasible $\ll$ infeasible
data structure	
number of feasible concepts	4 (4 hypersphere-shaped concepts)
shape of concepts	smooth boundaries, easy to learn
data properties	
feasible examples	$1,000,000$
infeasible examples	$1,000,000$
distribution:	
feasibles	representative, homogeneous
infeasibles	representative, homogeneous (for the region near the class boundaries)

Table 4.4: Overview of the *Hypersphere* classification task and data set properties

All these data sets are introduced with regard to some classification task properties, the data structure and properties resulting from the data set generation. For the evaluation of the classification experiments and to judge the classifier selectivity the distribution of feasible and infeasible examples is very important. The examples can represent the whole volume of the respective class (representative distribution) or not. Furthermore these examples can be distributed homogeneously or inhomogeneously in the sampled region of the respective class. Since the infeasible classes are much larger than the feasible ones, most infeasible examples are located far way from the class boundary.

More detailed descriptions off all data sets (data set properties and sample generation) can be found in the appendices, Sect. A for single energy units, Sect. B for artificial data sets and Sect. C for global operation schedules of coalitions of energy units.

Property	Characteristic
classification task properties	
greatest correlation length	2
correlation decrease	fast decrease (within one time steps)
balance of classes	feasible $\ll$ infeasible
data structure	
number of feasible concepts	1
shape of concepts	smooth boundaries, difficult to learn
data properties	
feasible examples	$1,000,000$
infeasible examples	$1,000,000$ (for each contour region)
distribution:	
feasibles (HB1)	less representative, not homogeneous
feasibles (HB2)	representative, not homogeneous
infeasibles	representative, homogeneous (in the corresponding contour region)

Table 4.5: Overview of the *Hyperbanana* classification task and data set properties. The only difference between the two data sets HB1 and HB2 consists in the distribution of the feasible examples.

4.1.2 Evaluation Scenarios

The evaluation scenario categories are designed according to classification task and data set properties, that are important for the cascade classification model and its adaptations. The main property is the data structure of the feasible class which is characterized by the number of concepts and the shape of theses concepts. The scenarios A and B employ different numbers of concepts with shapes, that are easy to learn. The scenarios C employ arbitrary number of concepts with hard to learn shapes.

Scenarios in the class A, see Tab. 4.6 employ one feasible concept with a more or less easy to learn shape. Such data sets are modifications of the μCHP and the *Hypersphere* data set and the unmodified *Hyperbanana* data set. Even though the *Hyperbanana* data sets are a bit difficult to learn, they belong to the scenarios A due to only one feasible concept and the clearly separable classes.

Scenarios A	Data set
A-CHP	μCHP (middle concept is feasible, edge concept and infeasible class are infeasible)
A-HS	*Hypersphere* (middle concept is feasible and remaining concepts and infeasible class are infeasible)
A-HB1	*Hyperbanana* set 1 (unmodified)
A-HB2	*Hyperbanana* set 2 (unmodified)

Table 4.6: Overview of the evaluation scenarios A with one feasible concept and a more or less easy to learn shape.

Scenarios in the class B, see Tab. 4.6 employ several feasible concept with an easy to learn shape. Such data sets are the μCHP and the *Hypersphere* without any modifications.

Scenarios in the class C, see Tab. 4.8 employ an arbitrary number of feasible concept with a hard to learn shape. Such data sets are the heat pump data set and data sets of coalitions of homogeneous energy units e.g., μCHPs and coalitions of mixed energy units e.g., μCHPs and heat pumps.

Scenarios B	Data set
B-CHP	μCHP (unmodified)
B-HS	*Hypersphere* (unmodified)

Table 4.7: Overview of the evaluation scenarios B with several feasible concept with more or less easy to learn shapes.

Scenarios C	Data set
C-HP	*Heat pump* (unmodified)
C-CHP	homogeneous coalition of μCHPs (unmodified global operation schedules)
C-CHP-HP	mixed coalition of μCHPs and heat pumps (unmodified global operation schedules)

Table 4.8: Overview of the evaluation scenarios C with arbitrary numbers of feasible concept and a hard to learn shape.

4.1.3 Experimental setup

The evaluation experiments are all setup in a similar way. If nothing else is indicated the high-dimensional data sets ($dim = 96$) are split into low-dimensional subsets with ($p = 2$).

Basic experimental setup

The basic setup for the basic cascade classification model is as follows:

1. Preparation
    a) training set of N feasible examples
    b) validation set of N feasible examples
    c) test set of 10000 feasible and 10000 infeasible examples
2. Learning
    a) parametrization of the low-dimensional classifiers with gridsearch and the validation set according to the validation results (TP values)

   b) storing the best classifier parameters or the best classifiers

3. Prediction
   a) prediction of all low-dimensional subsets of the test data set (intermediate results)
   b) aggregation of the intermediate results to an overall result

Experimental setup with data preprocessing

If data preprocessing is applied the basic setup changes a little. Some steps have to be done before the "Preparation" and the "Learning" phase changes. Since data preprocessing can consist of the selection of feasible examples (case 1) or also generation of artificial infeasible examples (case 2).

Case 1 Only selection of feasible examples is applied to the feasible raw data before the preparation steps.

Case 2 Selection of feasible examples and generation of artificial infeasible examples is applied to the feasible raw data.

- **1 a)** training set of N feasible examples (and N infeasible ones for binary baseline classifiers)
- **1 b)** validation set of N feasible examples and N infeasible ones
- **2 a)** parameter optimization according to the validation accuracy

Experimental setup for the generalized model

The application of the generalized cascade classification model requires some changes of the basic setup. The raw data has to be transformed before the "Preparation" phase and the aggregation in the "Prediction" phase has to be adopted.

Appropriate transformations have to be chosen in pre-tests according to the effect on the data structure. Most of the transformations can be directly applied to the complete raw data set. But transformations like the principle component analysis that rely on the whole data set and not on single time series have to be computed at first on the feasible raw data for the training and validation set. Then the transformation with the fitted coefficients can be applied to the test data set. If low-dimensional infeasible training and or validation data sets are available they can be also transformed afterwards.

Now data preprocessing could be applied as usual. The next changes occur in the "Prediction" phase, where changes are required after the computation of the intermediate aggregation results for all ensemble classifiers.

3 b) aggregation of the intermediate ensemble results according to the aggregation schemes, see Fig. 3.9)

4.1.4 Baseline Classifiers and their Default Parametrization

The experiments are conducted with different one-class and binary classifiers. The applied one-class classifiers are a classic One-Class Support Vector Machine (OCSVM), a k nearest neighbor classifier, called kmeans and a mixture of Gaussian models (MOG). The applied binary classifiers are a binary Support Vector Machine (SVM) and a common k nearest neighbor classifier (knn). For more details about the applied classifiers, see Sect. 2.4. For the different baseline classifiers and the different data sets some parameter combinations, respectively parameter ranges are used, see Tab. 4.9. These parameters result from pre-tests and parameter ranges are mainly not sampled homogeneously. But they are sampled with higher densities near the expected best parameter value.

4.1.5 Experimental Evaluation

The different classification experiments are mainly evaluated according to the classifier precision. The overall precision is considered as well as the precision of the single classifiers of the cascade. Precision of the low-dimensional subsets is assessed in terms of a visualization of the decision boundaries. The precision of the complete classifier cascade for the high-dimensional data set is assessed as true positive rates (TP) and true negative rates (TN). TP and TN values are computed from the confusion matrix of the test set predictions and the true labels of the test examples, see Sect. 2.5.1.

Name	Parameter values and ranges
OCSVM-param-1	kernel=rbf, $\nu \in \{0.0001, 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.1, 0.2\}$, $\gamma \in \{0.1, 1, 10, 50, 100, 150, 200\}$
OCSVM-param-2	kernel=rbf, $\nu \in \{0.0001, 0.0005, 0.001, 0.002, \ldots, 0.009, 0.01\}$, $\gamma \in \{50, 60, \ldots, 200\}$
OCSVM-param-3	kernel=rbf, $\nu \in \{0.0001, 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.1, 0.2\}$, $\gamma \in \{5, 10, 20, 30, 40, 50\}$
OCSVM-param-4	kernel=rbf, $\nu \in \{0.0001, 0.0005, 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025, 0.05\}$, $\gamma \in \{90, 100, \ldots, 250\}$
OCSVM-param-5	kernel=rbf, $\nu \in \{0.0001, 0.0005, 0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009, 0.01\}$, $\gamma \in \{0.1, 1, 10, 50, 100, 150, 200\}$
OCSVM-param-6	kernel=rbf, $\nu \in \{0.0001, 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.1, 0.2\}$, $\gamma \in \{1, 5, 10, 20, 30, 40, 50\}$
OCSVM-param-7	kernel=rbf, $\nu \in \{0.0001, 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.1, 0.2\}$, $\gamma \in \{2, 5, 10, 20, 30, 40, 50\}$
OCSVM-param-8	kernel=rbf, $\nu \in \{0.0001, 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.1, 0.2\}$, $\gamma \in \{10, 50, 100, 150, 200\}$
kmeans-param-1	$k \in \{1, 2, \ldots, 20\}$, $\epsilon_t \in \{0.01, 0.05, 0.1, 0.15\}$
MOG-param-1	$k_t \in \{5, 6, \ldots, 15\}$, $\epsilon \in \{0.01, 0.05, 0.1, 0.15, 0.2\}$
SVM-param-1	kernel=rbf, $C \in \{1, 10, 50, 100, 500, 1000, 2000\}$, $\gamma \in \{1, 5, 10, 15, 20\}$
SVM-param-2	kernel=rbf, $C \in \{1, 10, 50, 100, 500, 1000, 2000\}$, $\gamma \in \{1, 10, 20, 30, 40, 50, 60, 70, 80\}$
knn-param-1	$k \in \{1, 2, \ldots, 26\}$

Table 4.9: Parametrization of the different (baseline) classifiers for the evaluation experiments.

4.1.6 Implementation

I have implemented the cascade classification model with simple for loops calling the baseline classifier for all low-dimensional data subsets. The implementation is based on *python 2.7* in combination with *numpy 1.8.2*, T. E. Oliphant 2006, *scipy 0.13.3*, E. Jones et al. 2001, *matplotlib 1.3.1*, Hunter 2007, *sklearn 0.14.1*, Pedregosa et al. 2011 and *matlab R2014.b* with *pymatlab 0.2.3*, a python interface to matlab, *PRTools 5.0*, D. Tax 2013b and *Dd_tools 2.0.0*, D. Tax 2013a.

All classifiers, used in this thesis, see Sect. 2.4 are taken from different packages. The OCSVM, the SVM and the knn classifier are taken from *sklearn 0.14.1*, Pedregosa et al. 2011. The MOG and the kmeans classifier are implemented in *Dd_tools 2.0.0*, D. Tax 2013a. For comparison also the SVDD classifier applied in Bremer 2015 is applied. Here the respective python implementation of that SVDD is used from the Bachelor thesis of Graaff 2015, that is based on *tilitools*, see Görnitz 2015.

Below, pseudocode of the basic cascade classification model is presented.

Pseudocode of the Basic Cascade Classification Model

Pseudocode of the basic cascade classification model with an OCSVM baseline classifier.

```
\\ parameters
dim = ...    \\ dimensionality of the data set (default 96)
p = ...      \\ dimensionality of data subsets (default 2)
N = ...      \\ number of feasible training and validation examples
n_test = ... \\ number of test examples from each class
\\ OCSVM parameters
nu_values = [...]
gamma_values = [...]

\\ feasible training and validation data [n examples, dim features]
data_f = load(feasible_examples)

\\ test data [2 * n_test, dim features]
test_f = load(feasible_test_examples)
test_inf = load(infeasible_test_examples)
```

```
Test = [test_f[:n_test, :], test_inf[:n_test, :]]
\\ labels: 1:feasible, -1:infeasible
y_test = ones(2*n_test,1)
y_test[n_test:] = -1

result = zeros(2*n_test, dim-(p-1)) \\ intermediate results

for k in (1, 2,..., dim -(p-1):
    \\ training- and validation data
    X_train = data_f[:N, (k: k+(p-1))]
    X_val = data_f[N: 2N, (k: k+(p-1))]
    y_val = ones(N, 1)  \\ labels

    tp = 0 \\ initial value of validation TP

    \\ parametrization of the baseline classifier with gridsearch
    for i in nu_values:
        for j in gamma_values:
            clf = OCSVM(kernel=rbf, nu=i, gamma=j)
            clf.train(X_train)         \\ training
            pred = clf.predict(X_val)  \\ predict validation set

            \\ prediction evaluation and storing best parameters
            CM = confusion_matrix(pred, y_val)
            tp_new = CM[0,0] / (CM[0, 0] + CM[1, 0])
            if tp_new > tp:
                tp = tp_new
                nu_best = i
                gamma_best = j

    \\ prediction of test data
    clf = OCSVM(kernel=rbf, nu=nu_best, gamma=gamma_best)
    clf.train(X_train)
    result[:, k] = clf.predict(Test[:, (k: k+(p-1))])

\\ aggregation of intermediate results
\\ smallest predicted label of each test example
pred_test = min(result, axis=1) \\ axis 1 = dim-(p-1) features
```

4.2 Precision and Model Fitting

After the introduction of the basics of the classifier evaluation the first evaluation part focuses on classification precision. First off all the precision of the basic cascade classification model is evaluated in terms of input data dimensionality in Sect. 4.2.1, the baseline classifier choice in Sect. 4.2.2 and the dimensionality of the low-dimensional classifiers in Sect. 4.2.3. Then the impact of data preprocessing on the classification precision is evaluated in Sect. 4.2.4. After that the impact of the generalized cascade classification model is evaluated in terms of ensemble preparation in Sect. 4.2.5 and the aggregation of the intermediate results of all ensemble members to an overall result in Sect. 4.2.6.

These studies of classification model details are followed by a classifier selectivity study in Sect. 4.2.7 and a comparative study of the different cascade classification model versions and adaptations in Sect. 4.2.8. This study also includes a comparison to a commonly applied classifiers, a classic OCSVM and a SVDD classifier.

In all precision evaluation and comparison experiments, precision is measured in terms of true positive rates (TP) and true negative rates (TN) and in some cases also the learned decision boundaries are considered.

All results are summarized in Sect. 4.2.9. Finally model fitting suggestions are derived from the evaluation results, taking into consideration classification task and data set properties.

4.2.1 Scalability of the basic cascade classification model

Scalability is considered in terms of the the number of training examples and the dimensionality of the data set. The analysis is conducted on a μCHP data set like A-CHP but with 384 time steps instead of 96. The classification experiments are done on subsets of the data set with the dimensionality $dim \in \{48, 96, 144, \dots, 384\}$. These experiments are done with the default setup, with $N = \{1000, 2000, \dots, 10000\}$ training examples and the baseline classifier *OCSVM-param-2*.

The resulting TP and TN values depend on the number of training examples N and the dimensionality of the data set dim, see Fig. 4.1. Since the TN values are always equal to 1, they are not plotted.

As expectable, the TP values increase with increasing numbers of training examples

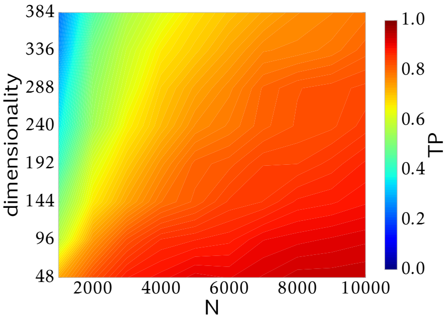


Figure 4.1: TP values for classification tasks of increasing dimensionality and increasing numbers of training examples (N). The classification was done with the basic cascade model and $p = 2$.

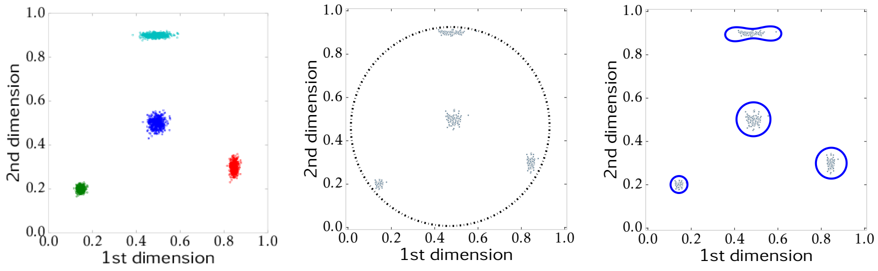
N and decrease with increasing dimensionality of the data set. In the experiments still more than 75% of the 384 dimensional examples were predicted correctly for high values of N, here $N \geq 8000$. In the application domain 384 dimensions, respectively 384 time steps usually correspond to a temporal horizon of three days with 15 min resolution. Scheduling of longer temporal horizons is not advisable due to great uncertainties resulting from uncertainties in the predicted training operation schedules, e.g., weather forecast, head demand prediction, etc..

4.2.2 Baseline Classifier Choice

After these scalability considerations now the effect of the different model adaptations is analyzed starting with the baseline classifier choice.

The cascade classification model allows the choice of baseline classifiers according to the structure of the low-dimensional data subsets. If the feasible class consists of only one concept, one baseline classifier is needed. Many classifiers lose precision, if the feasible class consists of more than one concept. This problem could be overcome with multi-task learning. Multi-task learning means learning each concept separately and merging the classification results to an overall result, see Fig. 4.2.

In this subsection the classification precision of different baseline classifiers is studied on data sets with a more or less simple data structure. More precisely a first



(a) class with four concepts (b) OCSVM decision boundary (c) OCSVM with multi-task learning decision boundary

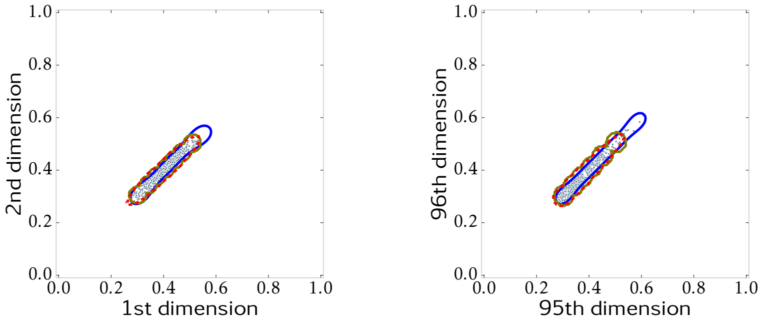
Figure 4.2: Binary *Hypersphere* classification example where the feasible class consists of four separable concepts (clusters). (a) All concepts are colored differently. (b) Decision boundary learned with an OCSVM. (c) Decision boundaries learned with separate OCSVMs for each concept.

study with different baseline classifiers is done with data sets, where the feasible class employs only one concept with a simple shape (A-CHP, A-HS). After that the effect of baseline classifiers and multi-task learning is studied in a second study with data sets with several feasible concepts and easy to learn shapes (B-CHP, B-HS).

Baseline Classifier Choice when the Feasible Class Employs One Concept

In this first study the impact of the baseline classifier choice on the classification precision is studied for a feasible class with one concept. The default experimental setup is used in combination with different one-class baseline classifiers (OCSVM, kmeans, MOG). The training set size for the reduced μCHP data sets (A-CHP) is $N = \{1000, 2000, \ldots, 15,000\}$ and for the reduced *Hypersphere* data sets (A-HS) $N = \{1000, 2000, \ldots, 10,000\}$. The experiments are conducted with the default setup and the same parametrization for both data sets: *OCSVM-param-1*, *kmeans-param-1* and *MOG-param-1*.

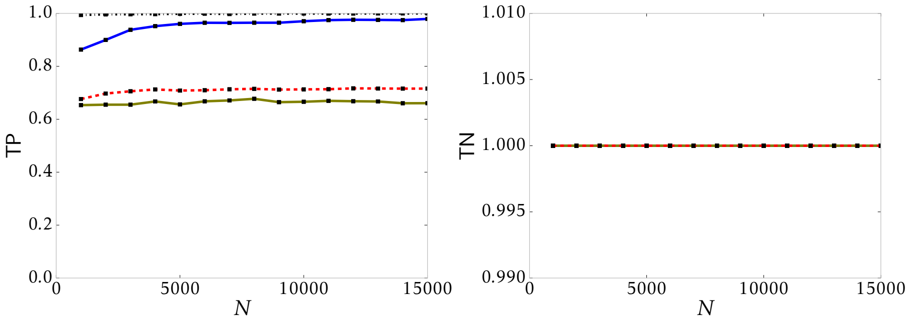
The resulting decision boundaries of the data set A-CHP are shown in Figure 4.3 for the first and the last subset of the cascade. Especially for the 2-dimensional



(a) $2dim$-boundaries on dim. 1/2

(b) $2dim$-boundaries on dim. 95/96

Figure 4.3: Decision boundaries on A-CHP, OCSVM (solid blue), kmeans (solid olive) and MOG (dashed red). Gray scattered points indicate 500 of $N = 5000$ training examples



(a) TP values (A-CHP)

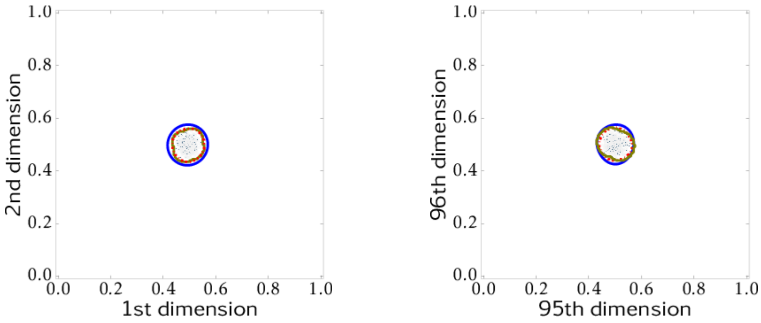
(b) TN values (B-CHP)

Figure 4.4: TP and TN values on A-CHP. The lines indicate cascade classification with OCSVM (solid blue), kmeans (solid olive), MOG (dashed red) and classic OCSVM (dashed dotted black). All TN values have a value of 1.

CHP data set the 2-dimensional OCSVM classifiers learn larger regions of feasible schedules than the MOG and the kmeans classifiers. As a result the OCSVM cascade predicts more feasible test schedules as feasible than the MOG and the kmeans classifier, see Fig. 4.4(a). The OCSVM baseline classifier achieves the highest TP values of all cascade model baseline classifiers. Therefore the OCSVM will be used in the following as baseline classifier for this data set. But a classic OCSVM achieves even higher TP values than the cascade classification model.

Furthermore all baseline classifiers and the classic OCSVM lead to TN values of 1, see Fig. 4.4(b). But these high TN values for the μCHP data set do not necessarily mean, that new infeasible test time series examples are classified correctly. The applied infeasible test examples are taken from the whole volume of the large infeasible class and therefore most of the examples are not located near the class boundary.

Next the results of the reduced *Hypersphere* data set are shown in Fig. 4.5 as decisison boundaries of the low-dimensional subsets and as TP and TN values in Fig. 4.6. The decision boundaries of the different baseline classifiers differ less than the decision boundaries on the A-CHP data set in Fig. 4.3, but the achieved TP values are similar. But the TN values differ for the Hypersphere data set, where the infeasible test examples are located near the class boundary, the OCSVM cascade has the lowest TN values and the classic OCSVM the highest, see Fig. 4.6(b). Increasing

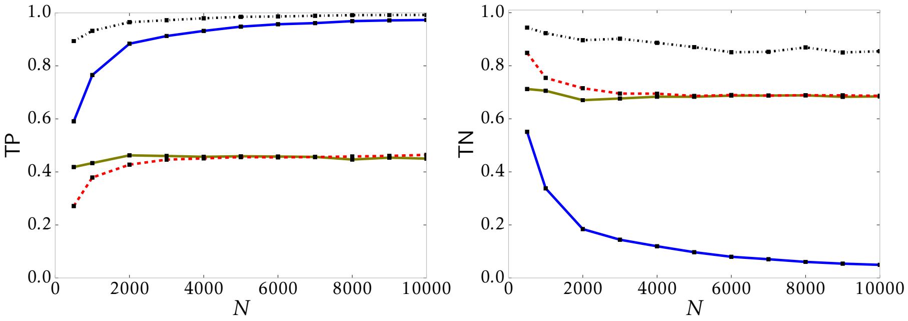


(a) 2dim-boundaries on dim. 1/2

(b) 2dim-boundaries on dim. 95/96

Figure 4.5: Decision boundaries on A-HS, OCSVM (solid blue), kmeans (solid olive) and MOG (dashed red). Gray scattered points indicate 500 of $N = 5000$ training examples

values of N lead to increasing TP values and decreasing TN values for the cascade approach, see Fig. 4.6. The more feasible examples the classifier predicts as feasible, the wider is the learned class boundary and the more infeasible examples near the class boundary lie inside the learned region of the feasible class. All in all no baseline classifier yields promising results on this A-HS data set. The classic OCSVM achieves definitely the best results.



(a) TP values (A-HS)

(b) TN values (A-HS)

Figure 4.6: TP and TN values on A-HS. The lines indicate cascade classification with OCSVM (solid blue), kmeans (solid olive), MOG (dashed red) and classic OCSVM (dashed dotted black).

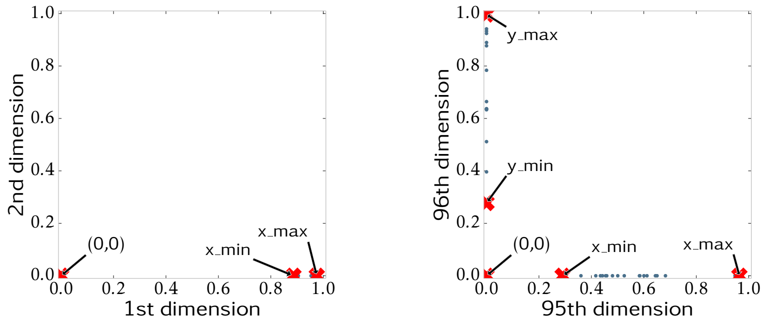
On both considered data sets (A-CHP, A-HS) the classic OCSVM achieves higher TP values and on the *Hypersphere* data set even higher TN values than the cascade approach. For these two reduced data sets with a simple data structure and only one feasible concept the cascade approach cannot keep up with the classic OCSVM.

This first baseline classifier choice study where the feasible class employs only one concept with an easy to learn shape is followed by the second study where the feasible class employs several concepts with easy to learn shapes.

Baseline Classifier Choice when the Feasible Class Employs Several Concepts

In this second study the impact of the baseline classifier choice for data sets with different feasible concepts is studied with a similar experimental setup and the same parametrization as the baseline classifier choice for data sets with one concept in the previous subsection. The big difference between these two studies is the application of multi-task learning in this second study. The training set size for the μCHP data sets (B-CHP) is $N = \{1000, 2000, \ldots, 15,000\}$ and for the *Hypersphere* data sets (B-HS) is $N = \{1000, 2000, \ldots, 10,000\}$. Additionally the list of baseline classifiers from the first study is extended for this second study by a min-max classifier.

This min-max classifier is designed for the 2-dimensional CHP edges concepts of B-CHP with $x = 0$ and or $y = 0$. The classifier learns whether there is a feasible example at $(0,0)$ and it learns the minimum > 0 and the maximum for the x and the y axis, see Fig. 4.7. New examples are classified as feasible, when they are located on the x or y axis between the minimum and the maximum or when they are located at (0,0) and this was a feasible training example. (Due to some CHP hard constrains concerning the CHP operation mode, there can be no feasible examples between 0 and $min > 0$, see Bremer et al. 2010.)



(a) $2dim$-boundaries on dim. 1/2

(b) $2dim$-boundaries on dim. 95/96

Figure 4.7: Min-max classifier for the μCHP data set. Red crosses mark the minimum (> 0) and maximum for both axes as well as the origin $(0,0)$, which can be feasible. The blue points indicate training examples

Depending on the correlation behavior and the structure of the data set as well as the separability of the feasible concepts there are two ways to combine the basic cascade classification model with multi-task learning. The feasible concepts may either be better separable in low-dimensional space or in high-dimensional space.

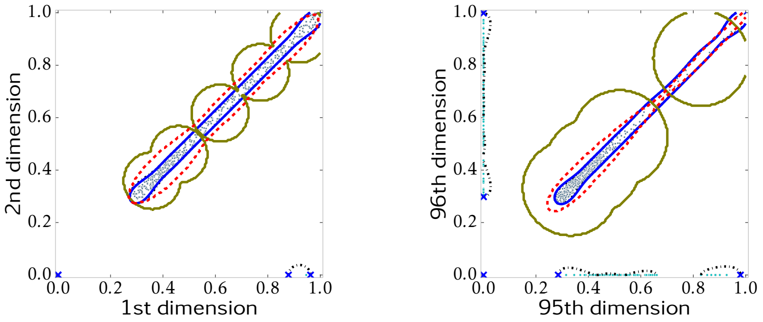
- First case: The concepts are better separable in low-dimensional space. In this case the data set is split into p-dimensional data subsets (default p=2) with overlapping features as for the cascade classification model. These p-dimensional data subsets are split according to the concepts based on knowledge about the concepts or based on clustering. In contrast to the basic cascade approach where one classifier is trained per cascade step, now one classifier is trained for each concept per cascade step. The intermediate results of the different classifiers of the cascade on the test set are combined as usual.

- Second case: The concepts are better separable in high-dimensional space. The high-dimensional data set is split into subsets according to the high-dimensional concepts. Each concept is learned with a cascade approach. The data sets of the concepts are treated as if they were different classification tasks. The cascade models yield results for each concept. To receive an overall result, the confusion matrices of all concepts are calculated and aggregated.

While the middle μCHP concept of the reduced data set A-CHP is separated in high-dimensional space from the edge concept, both concepts of B-CHP are separated in low-dimensional space in this subsection for multi-task learning. The separation in low-dimensional space leads to a slightly different middle concept than in A-CHP. Here the middle concept comprises more examples than in A-CHP.

The four feasible *Hypersphere* concepts are always separated in high-dimensional space for A-HS as well as for multi-task learning in this subsection.

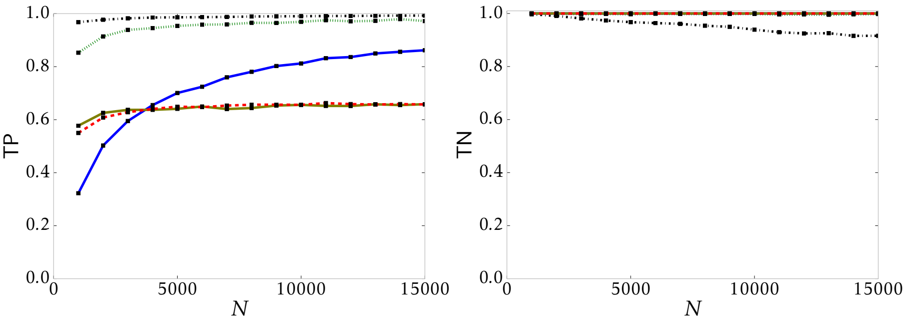
After the description of the experimental setup the results are presented for the two data sets one after another. First of all the results of the B-CHP data set are presented in Fig. 4.8 as decision boundaries and in Fig. 4.9 as TP and TN values. The OCSVM without multi-task learning and the OCSVM with multi-task learning (OCSVM + min-max) achieve similar decision boundaries for the concept in the middle (black line is hidden by the solid blue line). But the decision boundaries of the edge concepts are different (black line and blue crosses).



(a) $2dim$-boundaries on dim. 1/2

(b) $2dim$-boundaries on dim. 95/96

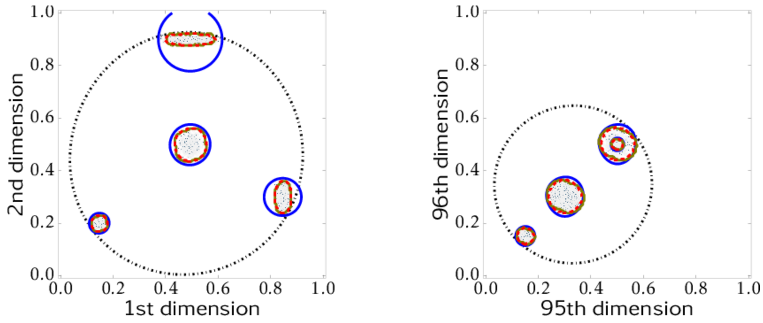
Figure 4.8: Decision boundaries on B-CHP, concept in the middle with OCSVM (solid blue), kmeans (solid olive) and with MOG (dashed red), crosses mark min-max boundaries and OCSVM without multi-task learning (dashed dotted black). The gray and cyan points indicated 500 of $N = 5000$ training examples.



(a) TP values of CHP data with multi-task learning

(b) TN values of CHP data with multi-task learning

Figure 4.9: TP and TN values of B-CHP with classic OCSVM (dashed dotted black) and cascade approach: OCSVM + min-max (solid blue), kmeans + min-max (solid olive), MOG + min-max (dashed red) and a single OCSVM (dotted green).



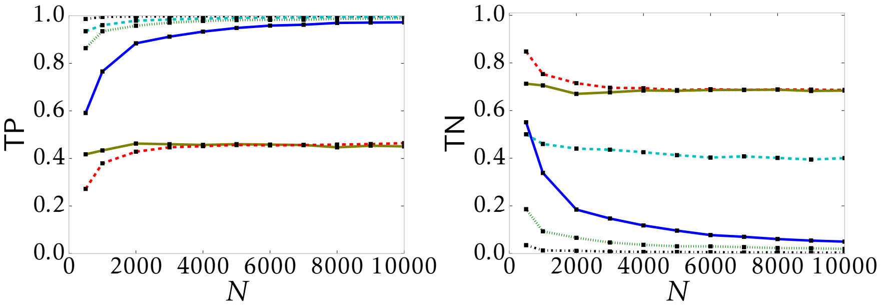
(a) $2dim$-boundaries on dim. 1/2

(b) $2dim$-boundaries on dim. 95/96, concepts overlap in this projection

Figure 4.10: Decision boundaries on B-HS, with multi-task learning OCSVM (solid blue), kmeans (solid olive), MOG (dashed red) and OCSVM without multi-task l. (dashed and dotted black). The gray points indicate 500 of $N = 5000$ training examples.

For higher numbers of training examples N, the TP values of the OCSVM cascade with multi-task learning approaches the TP values of the classic OCSVM and the TP values of the OCSVM cascade without multi-task learning. The TN values of the CHP classification is near 1 for all classifiers, except from the classic OCSVM where the TN values decrease for increasing values of N, see Fig. 4.9(b). The classic OCSVM achieves the highest TP values because the classifier overestimates the feasible regions at the cost of misclassified infeasible examples. The fact, that the infeasible test examples are not located near the class boundary shows, the classic OCSVM overestimates the feasible class to a great degree. Therefore the cascade classification model with an OCSVM and a min max baseline classifier is more suitable for B-CHP than a classic OCSVM.

Next the classification results of the complete *Hypersphere* data set (B-HS) are presented in Fig. 4.10 as decision boundaries and in Fig. 4.11 as TP and TN values. But a classic OCSVM and the classic OCSVM with multi-task learning achieve the highest TP values, see Fig. 4.11(a). But due to the overestimation of the feasible class by the classic OCSVM, see Fig. 4.10 most of the infeasible training examples are misclassified. This leads to low TN values for the classic OCSVM, see Fig. 4.11(b). The classic OCSVM with multi-task learning achieves higher TN values. The cascade



(a) TP values of *Hypersphere* data with multi-task learning

(b) TN values of *Hypersphere* data with multi-task learning

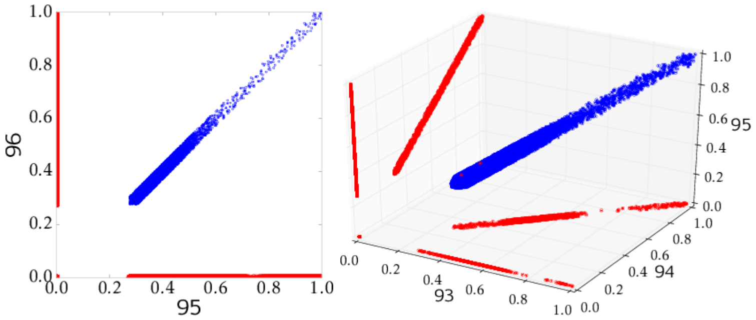
Figure 4.11: TP and TN values of B-HS with classic OCSVM without multi-task learning (dashed dotted black), with (dashed cyan) and cascade approach with multi-task learning: OCSVM (solid blue), kmeans (solid olive) and MOG (dashed red).

approach with OCSVM, MOG and kmeans classifiers achieve a bit higher TN values than the classic OCSVM without multi-task learning at the cost of lower TP values. The cascade classifiers show increasing TP values for increasing values of N. For high values of N the TP values approach values near to 1, except for the MOG and the kmeans classifier, which are therefore not of interest. The TN values of all classifiers decrease with increasing values of N. Since the infeasible examples are located near the class boundaries, the overestimation of the feasible class should not be very large. Altogether the cascade classification model reveals less precise results than the classic OCSVM with multi-task learning.

All in all multi-task learning improved the decision boundaries on both data sets. On the B-CHP data set the cascade classification model with multi-task learning yielded the highest precision. But on the B-HS data set a classic OCSVM with multi-task learning yielded a higher precision than the cascade classification model with multi-task learning. Beside the application of multi-task learning the choice of the baseline classifiers can greatly influence the classification precision.

4.2.3 Dimensionality of the Low-Dimensional Data Subsets

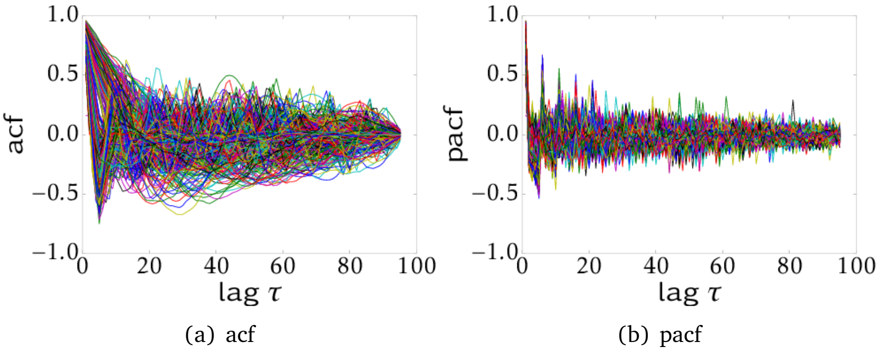
Beside the baseline classifier choice also the dimensionality of the low-dimensional subsets can be adapted. The optimal dimensionality of the data subsets (p) can be adopted according to the correlation behavior of the data set. The optimal dimensionality of the low-dimensional subsets mainly depends on the correlation length (τ) of the highest correlation between features (time steps) and the correlation decrease for features with increasing distance. Furthermore the data shape of the feasible class in the low-dimensional data subsets can differ with the dimensionality. For example the number of feasible concepts in the data subsets differs in the μCHP data set B-CHP with the dimensionality (p) of the subsets. For increasing dimensionality of the subsets, the number of feasible concepts increases, see Fig. 4.12.



(a) feasible concepts in 2d (b) feasible concepts in 3d

Figure 4.12: Feasible concepts of the μCHP data set (B-CHP) in different low-dimensional data subsets.

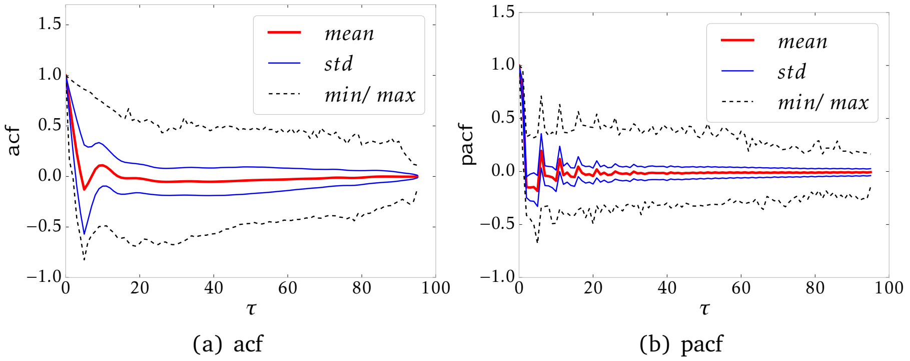
The correlation behavior is assessed with the autocorrelation function (acf) and the partial autocorrelation function (pacf), see Sect. 3.1. Since acf and pacf refer to single time series, an overall correlation behavior of the whole data sets has to be derived. Therefore acf and pacf have to be computed for all time series of the feasible class. Then the mean and the standard deviation can be computed as a measure of the overall correlation behavior. Based on the mean correlation behavior the correlation length (τ) of the highest correlation can be identified. Additionally the correlation decrease for increasing distances between the time steps can be judged. In case of a slow correlation decrease it is advantages to chose p larger than the correlation length of the highest correlation. In this case larger p values minimize an information loss for the classification. The correlation behavior of the μCHP data set (B-CHP) is exemplarily shown in Fig. 4.13 for single feasible time series and in Fig. 4.14 as mean values, the standard deviation and minimum and maximum values of several feasible time series.



(a) acf (b) pacf

Figure 4.13: Correlation behavior of the feasible class of B-CHP plotted for 500 time series.

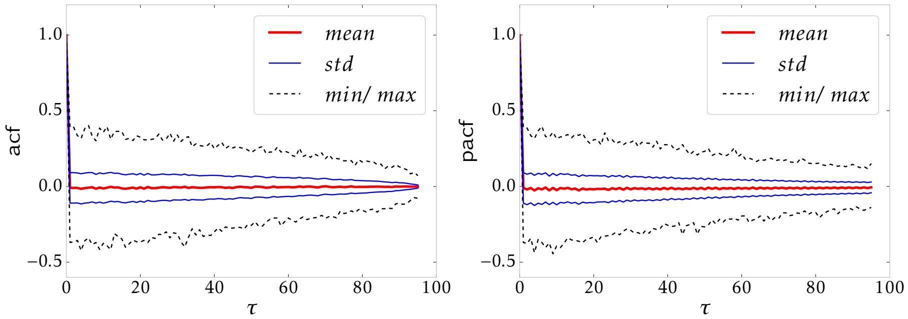
The cascade classification model is designed for time series with a high correlation between nearby time steps and a rapid correlation decrease for time steps with longer distances (τ). If the correlation behavior differs from this behavior splitting of high-



(a) acf (b) pacf

Figure 4.14: Overall correlation behavior of the feasible class of B-CHP plotted as mean and standard deviation of acf (a) and pacf (b) of 100,000 time series.

dimensional data sets into low-dimensional subsets can decrease the classification precision due to an information loss. Such an information loss is here also called projection error. An example of a data set with a different correlation behavior is the *Hypersphere* data set. The mean correlation is about 0 for all values of τ, see Fig. 4.15.

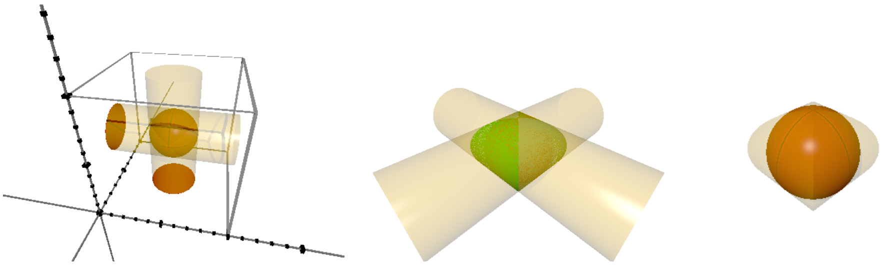


(a) acf (b) pacf

Figure 4.15: Overall correlation behavior of the feasible class of A-HS plotted as mean and standard deviation of acf (a) and pacf (b) of 100,000 feasible time series.

The projection error is exemplarily visualized in Fig. 4.16 for a 3-dimensional sphere. The classification of the sphere with the help of 2-dimensional data subsets results in an overestimation of the 3-dimensional sphere, see Fig. 4.16(c).

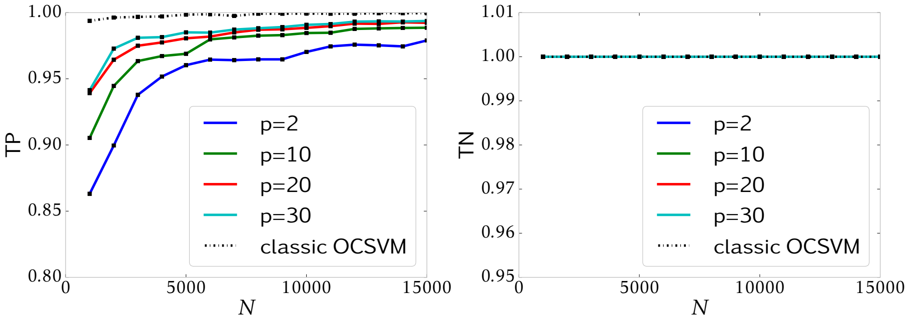
In the following the effect of the dimensionality p on the classifier precision is studied experimentally on the reduced μCHP data set (A-CHP) and the complete *Hypersphere* data set (B-HS). Further more the resulting precision is compared to the precision of a classic OCSVM. The experiments on A-CHP are done with the default setup and the experiments on B-HS are done with the default setup for the basic cascade classification model but with multi-task learning. Furthermore the experiments are done with the baseline classifier *OCSVM-param-1* on both data sets and a classic OCSVM with the same parametrization. Increasing values of p are chosen for A-CHP as $p \in \{2, 10, 20, 30\}$ and for B-HS as $p \in \{2, 20, 40, 60, 80\}$. The training set size (N) is chosen as $N = \{1000, 2000, \dots, 15000\}$ for the A-CHP data set and as $N = \{1000, 2000, \dots, 10000\}$ for B-HS.



(a) $2dim$-projections of $3dim$-sphere (b) intersection of 2 cylinders represents cascade classification (c) original sphere in intersection space of (b)

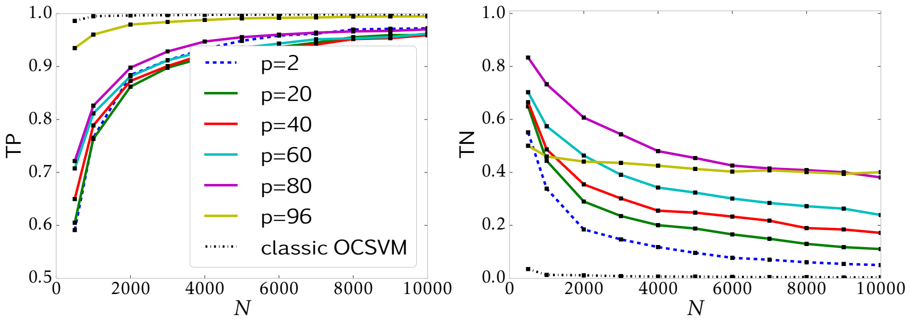
Figure 4.16: Geometric example of a 2-dimensional-cascade classification of a $3d$ sphere. This example shows a systematic classification (projection) error. The figures are plotted with POV-Ray, see Buck and Collins 2004.

First the classification results of the A-CHP data set are presented. The classification of the CHP middle concept shows increasing TP values for increasing values of p, see Fig. 4.17 and TN values of about 1 for all considered classifiers. The classic OCSVM with $p = 96$ achieves the highest TP values on A-CHP. Since TN is equal to 1 in all experiments, the classic OCSVM yields more precise results on A-CHP than the cascade classification model.



(a) TP values of CHP classification, varying p (b) TN values of CHP classification, varying p

Figure 4.17: TP and TN values of the 96-dimensional reduced CHP data set. Results are indicated for OCSVM with p-dimensional cascades, resp. classic OCSVM.



(a) TP values of *Hypersphere* classification with multi-task learning and varying values of p

(b) TN values of *Hypersphere* classification with multi-task learning and varying values of p

Figure 4.18: TP and TN values of the 96-dimensional *Hypersphere* data set. Results are indicated for OCSVM with p-dimensional cascades, resp. classic OCSVM. Legend of (a) is also valid for (b)

Next the results on B-HS are presented. On the *Hypersphere* data set, the TP values increase with increasing values of N and more or less with p, see Fig. 4.18. The higher the value of p, the slower converge the TP values. This leads to intersections of the TP value lines. The classic OCSVM without multi-task learning achieves the highest TP values and at the same time the lowest TN values. Increasing values of N lead to decreasing TN values for all values of p. Furthermore increasing values of p lead to increasing TN values. Also multi-task learning with a classic OCSVM ($p = 96$) achieves relatively high TN values compared to the other results, while a classic OCSVM without multi-task learning achieves the lowest TN values. This means a classic OCSVM achieves the worst precision on B-HS and a classic OCSVM with multi-task learning achieves the best precision. Thus the cascade classification model results (with multi-task learning) lie in between the results of a classic OCSVM with and without multi-task learning.

In summary increasing values of p showed a precision increase in all experiments. This means higher values of p than the p values resulting from the correlation considerations increase the classification performance. This observation was expectable, because projection errors decrease with increasing values of p. Additionally p values resulting from the correlation analysis are the smallest values of p which are expected to yield good classification results. Therefore p underlies a kind of trade-off between projection errors and errors resulting from the classification of high-dimensional data sets (curse of dimensionality). Unfortunately the experiments were done on data set where the feasible class employs one or more feasible concepts with an easy to learn shape and on such data sets a classic OCSVM performs better than the cascade classification model, see e.g., Sect. 4.2.2 or Sect. 4.2.8. Therefore the trade-off is not as good visible as it would be the case for data sets with a more complex data structure.

4.2.4 Data Distribution Improvement due to Data Preprocessing

Beside the adaptation of the baseline classifiers and the dimensionality of the low-dimensional subsets also the data sets can be improved to achieve the best classification results. There are mainly two possibilities to improve the training and the validation data sets. The first possibility is instance selection that means choosing representative examples from a given data set. The second data distribution improvement possibility consist in the generation of artificial infeasible outliers which supplement the feasible training examples. Furthermore artificial infeasible examples allow the application of binary baseline classifiers instead of one-class baseline classifiers. Two respective data preprocessing examples for the cascade classification model were presented in Sect. 3.4.

In this subsection the precision improvement due to data preprocessing is studied for the scenarios A-CHP and A-HB1. Both data sets employ a different distribution of the feasible class. The feasible class of the μCHP data set is more representative and the examples are distributed a bit more homogeneous than the examples of the more complex *Hyperbanana* data set. Altogether three classification experiments are conducted on both data sets. The first experiment is done without preprocessing (no prepro.), the second with selected feasible examples (fs) and the third with selected feasibles and artificial infeasible examples (fs + infs). For all experiments a one-class baseline classifier (OCSVM) is used. The third experiment is also done with binary baseline classifiers (SVM, knn).

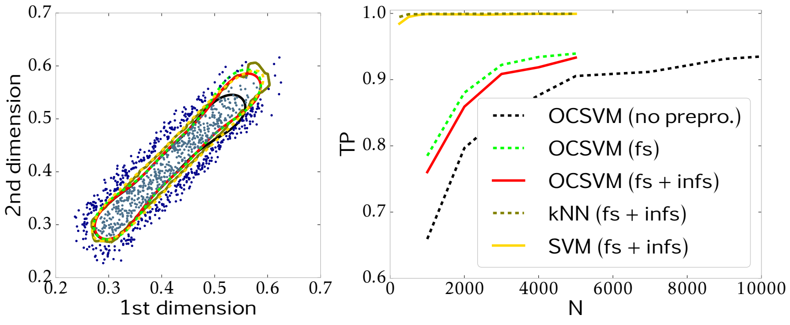
In the following theses experimental studies are presented. After that an other study shows limitations of the application of data preprocessing methods.

First of all the three experimental studies, showing the functionality of data preprocessing are presented. All these experiments are conducted with the default experimental setup for training sets of $N = \{1000, 2000, \dots, 10,000\}$ examples. The data preprocessing parameters were adapted in pre-tests. The pre-tests were conducted with different minimal distances ϵ and ϵ_b and evaluated according to the number of resulting feasible examples sn and their distribution in the 2-dimensional data subset. For the μCHP data set instance selection is parametrized as follows, the minimal distance between feasible examples is set to $\epsilon = 0.001$ and the number of new examples used for each iteration t is set to $t = 1000$. Generation of artificial infeasible examples is parameterized with $n = 15000$ initially feasible examples disturbance $= \mathcal{N}(0, 0.01) \cdot \alpha$ with $\alpha = 1$ and a minimal distance between infeasible examples and their nearest feasible neighbors $\epsilon_b = 0.025$. For the *Hyperbanana* data set the instance selection parameters are set to $\epsilon = 0.002$, $t = 1000$ and parameters for generating artificial infeasible examples are set to $n = 20000$, disturbance $= \mathcal{N}(0, 0.02) \cdot \alpha$ with $\alpha = 1$ and $\epsilon_b = 0.025$.

Classification of the differently preprocessed data sets is done with one-class and binary classifiers. On the A-CHP data set *OCSVM-param-2*, *SVM-param-1* and *knn-param-1* are applied. For the *Hyperbanana* data set (A-HB1) *OCSVM-param-5*, *SVM-param-2* and *knn-param-1* are applied.

Below the experimental results are presented for the three studies for the two data sets. In the classification experiments the proposed data preprocessing methods, selection of feasible examples and generation of artificial infeasible examples show an increase in classification precision of the cascade classifier. On both data sets A-CHP and A-HB1 data preprocessing leads to more precise decision boundaries than without data preprocessing, see Fig. 4.19(a) for the CHP results and Fig. 4.20 for the *Hyperbanana* results. This can be also seen in the TP and TN values of the classification results, see Fig. 4.19(b) for the CHP results and Fig. 4.21 for the *Hyperbanana* results.

For the μCHP data set, all three preprocessing experiments lead to TN values of 1, therefore only the TP values are plotted in Fig. 4.19(b). The first experiment without data preprocessing (no prepro.) yields the lowest TP values of all experiments for all numbers of training values N and the second experiment with selection of feasible examples (fs) leads already to higher TP values. The third experiment with selection of feasible examples and artificial infeasible examples (fs + infs) leads to different results with the OCSVM baseline classifier and the binary SVM and kNN

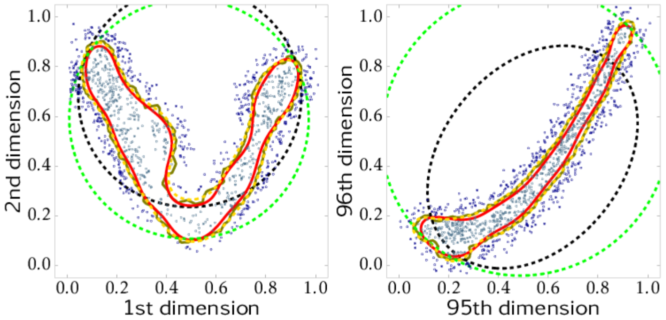


(a) decision boundaries (b) TP values

Figure 4.19: Decision boundaries (a) and TP values (b) on A-CHP. The legend in (b) is also valid for (a). The gray points in (a) indicate 500 selected feasible examples and the blue points 500 artificial infeasible examples.

baseline classifiers. While the OCSVM(fs + infs) achieves slightly lower TP values than OCSVM(fs) in the second experiment, the binary baseline classifiers SVM(fs + infs) and kNN(fs + infs) achieve TP values near 1 even for small numbers of training examples N. This means preprocessing does not only improve the classification precision on the A-CHP data set, but also allows the application of binary baseline classifier. These binary baseline classifiers yield a higher precision than the one class classifier.

Next the results of the *Hyperbanana* data set with a more complex data structure are presented. Data preprocessing influences the decision boundaries, see Fig. 4.20 and the TP values, see Fig. 4.21(a) and the TN values, see Fig. 4.21(b) of the classification results. In the first experiment (no prepro.) and second experiment (fs) the classification achieves relatively high TP values and at the same time the lowest TN values of all experiments due to an overestimation of the feasible class, see Fig. 4.20. This time all TN values are relatively low because the infeasible test examples are all located close to the class boundaries. The third experiment (fs + infs) revealed an opposed behavior of the OCSVM baseline classifiers. The OCSVM(fs + infs) achieves lower TP values than the OCSVM in the previous experiments but also the highest TN values of all experiments. SVM and kNN baseline classifiers with (fs + infs) achieve the highest TP values of all experiments and at the same time lower TN values than

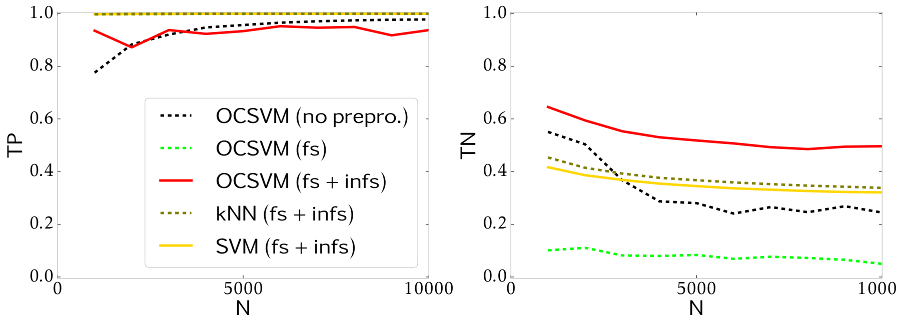


(a) $2d$-boundaries on dim. 1/2 (b) $2d$-boundaries on dim. 95/96

Figure 4.20: Decision boundaries on the *Hyperbanana* data set trained with $N =$ 1000 feasible (+ 1000 infeasible) training examples, no prepro. (dashed black), fs (dashed green), OCSVM(fs + infs) (red), kNN(fs + infs) (olive) and SVM(fs + infs) (yellow). The gray points indicate 500 of the selected feasible training examples and the blue points 500 of the artificial infeasible examples.

the OCSVM(fs + infs). This means data preprocessing increases the classification precision. This time the one class baseline classifier yields more precise results than the binary baseline classifiers.

In summary, data preprocessing increases the classification precision of the cascade classifier on both data sets. While selection of feasible examples increases the classification precision, artificial infeasible examples can lead to an even greater increase depending on the data set and the baseline classifier choice. Furthermore artificial outliers allow the application of binary classifiers and they can increase the classification precision once more compared to the one-class baseline classifiers. On the more complex *Hyperbanana* data set with infeasible test examples near the class boundaries, the impact of data preprocessing is visible in the TP and the TN values. Even though data preprocessing greatly improves the classification precision in these experiments, this is not always the case, especially not for data sets with a very inhomogeneous distribution of the feasible class. If selected nearest feasible neighbors have a large distance between each other, especially at the concept edges, generation of artificial infeasible examples does not increase classifier precision and introduces errors.

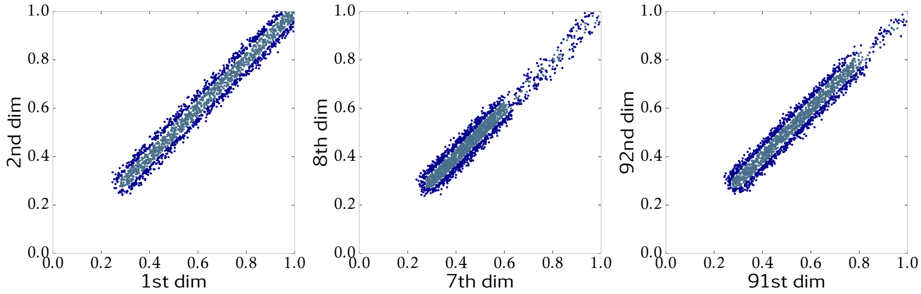


(a) TP values (A-HB)

(b) TN values (A-HB)

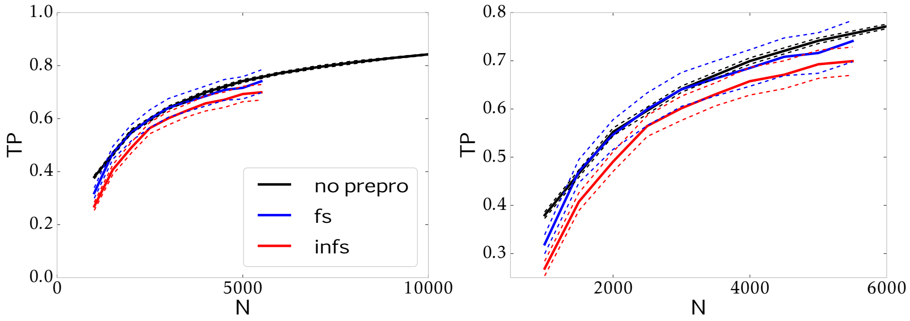
Figure 4.21: TP and TN values on the *Hyperbanana* data set for different preprocessing steps and different baseline classifiers. The legend in (a) is also valid for (b). The green line of OCSVM(fs) in (a) is covered by the olive and the yellow line.

Limitations of the application of data preprocessing is presented in the following at an example. Erroneous artificial infeasible examples occur in the μCHP middle concept, when both concepts are separated in low-dimensional space, see Fig. 4.22. This data set is different from A-CHP where both concepts are separated in high-dimensional space. Especially in Fig. 4.22(b) and Fig. 4.22(b) artificial infeasible examples are generated in the region of the feasible class. Next the classification experiment and the respective results are presented for this data set. The B-CHP data set is classified as in Sect. 4.2.2 with multi-task learning (*OCSVM-param-1* + min-max classifier) and additional the middle concept is preprocessed with $\epsilon = 0.001$, $\epsilon_b = 0.007$ and $t = 100$. The resulting classification precision is shown in Fig. 4.23 as mean TP values of 10 experimental runs. Even a few problematic subsets with wrong artificial infeasible examples, like in this example result in a worse classification precision with artificial infeasible examples than without them, see Fig. 4.23. The mean TP values of the experiment without preprocessing (no prepro) and the experiment with selection of feasible examples (fs) are pretty similar. The classification precision with selection of feasible examples is probably not higher than without preprocessing, because selection of feasible examples cannot homogenize the feasible examples much. Therefore the selected feasible examples are still



(a) B-CHP middle dim 1/2 (b) B-CHP middle dim 7/8 (c) B-CHP middle dim 91/92

Figure 4.22: If the two μCHP concepts are separated in 2-dimensional space, the feasible examples of some middle concept subsets are distributed very inhomogeneously. Data preprocessing with $\epsilon = 0.002$, $\epsilon_b = 0.007$, $t = 100$ yields the light blue feasible examples and the dark blue points are artificial infeasible examples.



(a) TP values (B-CHP) (b) detail view of (a)

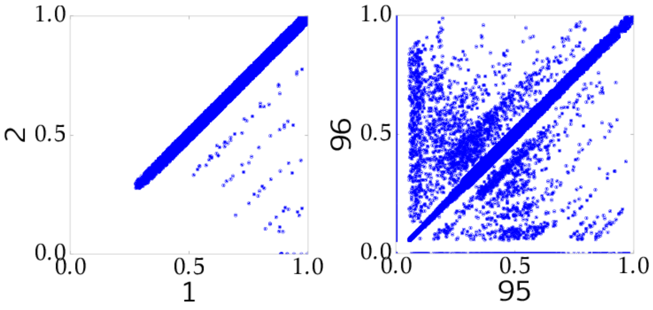
Figure 4.23: TP values on B-CHP for different kinds of data preprocessing. Mean values are indicated with solid lines and standard deviation with dashed lines.

inhomogeneously distributed, see Fig. 4.22. Furthermore there might be only a very small range of optimal ϵ values and a stronger fine tuning of ϵ could probably increase the classification precision a little. Parameter sensitivity of the data preprocessing parameters is studied Sect. 4.3.2.

All in all data preprocessing can increase classifier precision after a careful parametrization. For a detailed study and more parameter fitting advice see, Sect. 4.3.2. Selection of feasible examples can be always applied and generation of artificial infeasible examples can be helpful, if the distribution of feasible examples is not very inhomogeneous. Furthermore artificial infeasible examples allow the application of binary baseline classifiers and they might yield a higher precision than one class baseline classifiers.

4.2.5 Ensemble Preparation: Transformations and Data Preprocessing

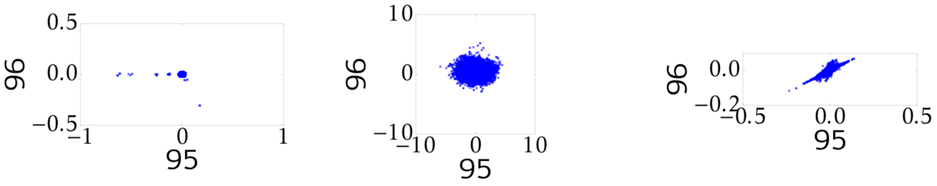
Beside adaptations of the basic cascade classification model and data preprocessing, a generalized version of the casade classification model can improve the classification precision. The generalized cascade classification model, based on ensembles was particularly developed for complex classification tasks, where the feasible class employs a complex data structure. Such data sets are e.g., global operation schedules of energy unit coalitions, see Fig. 4.24 or heat pump operation schedules.



(a) step 1 and 2 (b) step 95 and 96

Figure 4.24: The first and the last two time steps of 96-dimensional time series are plotted against each other for a coalition of 5 μCHPs. Each subfigure is plotted with 30, 000 time series segments.

Data transformations can simplify the classification task and lead to a less complex data structure. Transformation results of global operation schedules of the μCHP coalition in Fig. 4.24 are shown in Fig. 4.25.



(a) pca(X) (b) fft(X) (c) acf(X)

Figure 4.25: Transformed feasible operation schedules of a coalition of 5 μCHPs. The figures show 30, 000 operation schedule segments of the 95th and the 96th time step.

Appropriate data transformations, which simplify the classification task have to be identified in pre-tests. The transformations are preferably chosen from the three domains: temporal domain, spectral domain and change domain, see Sect. 3.5.2. These domains can be arbitrarily modified or extended e.g., by principle components. After the selection of appropriate transformations, also data preprocessing could be applied to improve the classification precision. If the feasible global operation schedules are distributed more inhomogeneously like in the data sets of energy unit coalitions (C-CHP), selection of feasible examples cannot homogenize the distribution of the feasible examples much.

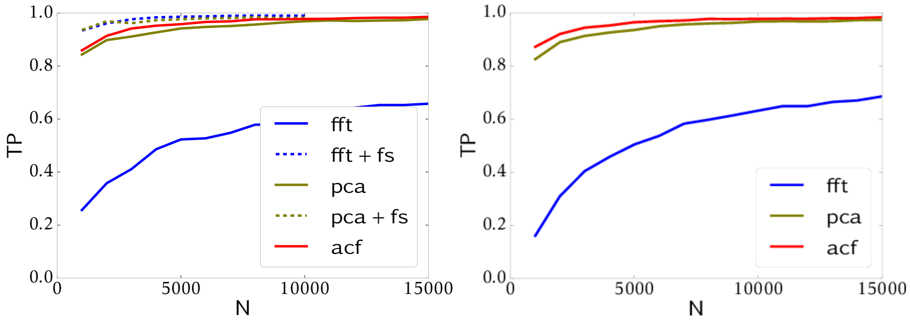
In the following the effect of data transformations and data preprocessing on the classification precision is studies for the members of the generalized cascade classification model one after another. The low-dimensional data subsets of the transformed feasible operation schedules often differ a little in shape and some still employ not so easily learnable data structures. Therefore the classification results of each transformed data set are expected to yield not very high TP values and in some cases also not very high TN values. But ensembles of these classifiers can overcome the weaknesses of the single classifiers. In this subsection not the ensemble results, but the classification results of the ensemble members are of interest. The effect of data transformations and data preprocessing (selection of feasible examples) is shown exemplarily for the heat pump data set (C-HP) and a coalition of 5 μCHPs (C-CHP). First of all the experiments with the heat pump data set are conducted. Therefore the data set is transformed with fft(X), acf(X) and pca(X) and divided into 2-dimensional and 3-dimensional subset. Then selection of feasible examples, parametrized with $\epsilon = 0.0015$ and $t = 100$, is applied to the 2-dimensional data subsets transformed with fft(X) and pca(X). The acf transformed data set is not processed with selection of feasible examples, because the data set employs such an inconvenient data distribution, where selection of feasible examples can hardly smooth the distribution. Basic cascade classifiers with OCSVM baseline classifiers (*OCSVM-param-1*) are built on all data sets (p=2 with (+fs) and without selection of feasibles), (p=3 without selection of feasibles).

After that the experiments on the μCHP coalition are conducted. The data sets are transformed with fft(X), acf(X) and pca(X). Then the transformed data sets are split into 2-dimensional subsets. The cascade classification model is applied with the default setup for the generalized model with *OCSVM-param-1* for all transformations. For comparison the untransformed data set is classified with a basic cascade classification model with *OCSVM-param-6*.

The classification results of the different transformations of the heat pump data set (C-HP) are shown in Fig. 4.26(a) in form of TP values with and without selection of feasible examples for $p = 2$. In Fig. 4.26(b) the TP values are shown for low-dimensional data subsets of $p = 3$. The TN values are always equal to 1. The TP values of the acf and pca transformation are relatively high for p=2 and p=3 compared to the TP values of the fft transformation. One reason for low TP values on the fft transformations is the neglect of the imaginary part. But selection of feasible examples strongly increases the TP values of the fft transformation.

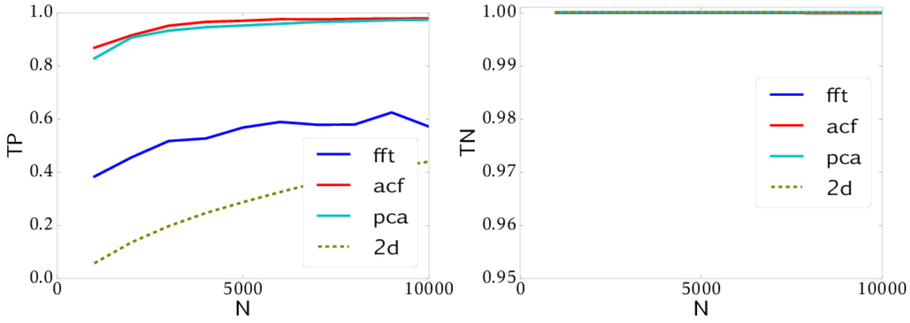
The precision gain due to appropriate data set transformations becomes even more obvious in the experiments with the μCHP coalition data set in Fig. 4.27. While all classifiers achieve TN values equal to 1, they strongly differ in the TP values. The cascade classification model achieves much higher TP values on all transformed data sets than one the untransformed one.

All in all the intermediate classification results on the heat pump and the μCHP coalition data set showed, that a careful preparation including the choice of



(a) TP values, $p = 2$ (b) TP values, $p = 3$

Figure 4.26: TP values for different transformations of the heat pump data set (C-HP) where the low-dimensional data sets consist of either p=2 (a) or p=3 time steps (b). In (a) +fs indicates the application of selection of feasible examples.



(a) TP values (b) TN values

Figure 4.27: TP and TN values for different transformations of a μCHP coalition with 5 members (C-CHP). Additionally the TP and TN values for the untransformed data set are indicated as 2d.

transformations and possibly also selection of feasible examples increase the precision of the classifiers in the ensemble. A basic cascade classification model achieves a much lower classification precision on the untransformed data set than with appropriate transformations.

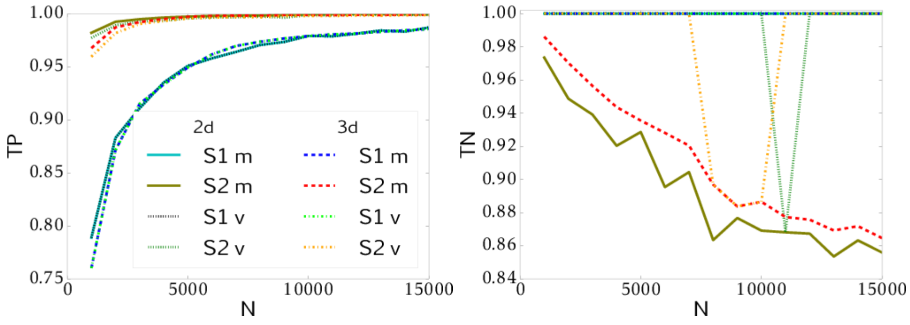
Since the transformed data sets lead to a relatively high precision, ensembles of these classifiers are expected to yield an even higher precision. The aggregation of these intermediate results to overall ensemble results is presented in the next subsection.

4.2.6 Ensemble Precision

In this subsection the classifier precision of the generalized cascade classification model is analyzed. Therefore the impact of different combinations of the aggregation schemes and weighting schemes is compared on the overall ensemble results. The experiments are done on the same data sets as the ensemble preparation experiments in the previous section, see Sect. 4.2.5, the heat pump data set (C-HP) and global operation schedules of homogeneous μCHP coalitions (C-CHP), this time consisting of different numbers of energy units. These data sets employ small differences in the data set complexity and slightly different distributions of the feasible examples. Finally the effect of the data set complexity on the classification precision is considered.

All experiments are set up as in the previous section, see Sect. 4.2.5. But this time without data preprocessing. Furthermore all μCHP coalitions are parametrized as the coalition in Sect. 4.2.5. The classification results of all ensemble members are aggregated to overall results with different combinations of the two aggregation schemes S1 and S2 and the weighting schemes equal weighting and validation precision based weighting, see Sect. 3.5.2.

First the aggregated results are presented for the heat pump data set in Fig. 4.28. While the two aggregation schemes S1 and S2 lead to different TP and TN values, the choice of the weighting scheme shows hardly any effect on the overall classification precision. Scheme S1 achieves always TN values equal to 1 and a bit lower TP values than aggregation scheme S2. But S2 leads to lower TN values than S1. The TP and TN values in Fig. 4.28 form two groups according to the applied aggregation schemes. Furthermore the dimensionality of the low dimensional classifiers, here $p = 2$ and



(a) TP (b) TN

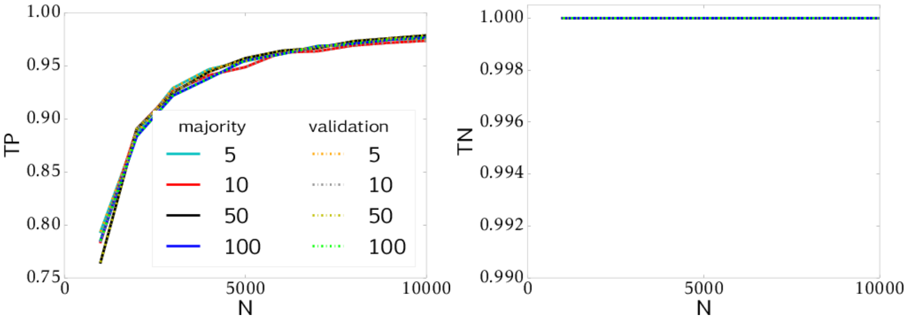
Figure 4.28: TP and TN values of the heat pump ensemble classifiers with p=2 and p=3, without data preprocessing and increasing numbers of training examples N. The two aggregation schemes are indicated as S1 and S2 and the different weighting schemes equal weighting as m and validation performance based weighting as v. The legend in (a) is also valid for (b). All hidden lines in (b) are equal to 1.

$p = 3$, shows no effect on the classification precision.

After the experiments with the heat pump data sets, the experiments on the μCHP coalition data sets are conducted. The resulting TP and TN values are shown in Fig. 4.29 for the different aggregation and weighting scheme combinations. The C-CHP data sets show similar results as the heat pump data set (C-HP), concerning the aggregation of the intermediate classification results of all ensemble members. The choice of the aggregation scheme influences the TP and TN values, while the choice of the weighting scheme shows hardly any effect.

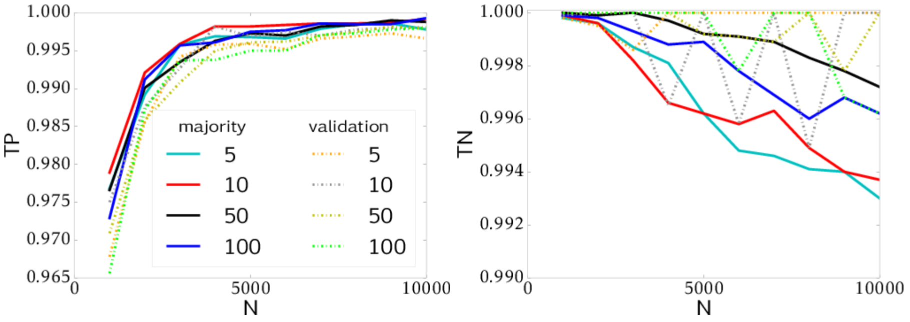
Even though an increasing number of energy units in the coalitions slightly increase the data set complexity, there is no effect visible in the classification results. Thus small changes in the data set complexity have no effect on the classification precision.

Similar experiments on μCHP coalitions were presented in Neugebauer, Bremer, et al. 2016. But these experiments were done with less representative examples of the feasible class, where most examples are distributed around the expectation value. These data sets all employ a similar data structure and also a similar data set complexity, see also Sect. C.1. Therefore these experiments are not presented in



(a) TP (S1)

(b) TN (S1)



(c) TP (S2)

(d) TN (S2)

Figure 4.29: TP and TN values of coalitions with different numbers of μCHPs. The intermediate results are aggregated according to aggregation scheme S1 in (a) and (b) and according to S2 in (c) and (d). The legend in (a) is also valid for (b) and the legend in (c) is also valid for (d).

more detail here.

Overall the results on the heat pump data set and the μCHP coalition data sets show similar results. While the data set complexity and the choice of the weighting scheme showed hardly any effect on the ensemble precision, the choice of the weighting scheme exerts a great influence on the ensemble precision. Both aggregation schemes yield similar or higher TP values than the single ensemble members and a basic cascade classification model without data set transformations, see Sect. 4.2.5. Aggregation scheme S1 results in TN values equal to one, like all single ensemble members. But aggregation scheme S2 leads to lower TN values. Since most of the infeasible test examples of both data sets are not located close to the class boundary, the TN values indicate classification mistakes. Therefore aggregation scheme S1 yields more precise results than the aggregation scheme S2.

Aggregation scheme 1 combines the results of all ensemble members for complete global operation schedules and conserves the temporal ordering of the time steps. But aggregation scheme 2 combines the intermediate predictions of the ensemble members. These intermediate predictions do not correspond to time steps, but to special frequencies, the correlation of time steps with a certain distance and so on. Since S2 aggregates the intermediate prediction results from different domains (e.g., temporal, spectral and change), S2 is not suitable for all data sets.

The results from the previous subsection Sect. 4.2.5 and this subsection showed, well prepared ensembles increase the classification precision. A more detailed comparison of the different cascade classification model version is presented in Sect. 4.2.8. But before the classifier comparison the selectivity of the cascade classification model is studied in the next section, see Sect. 4.2.7.

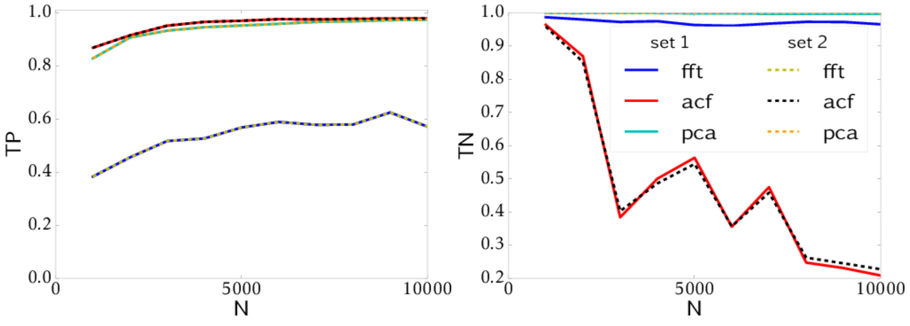
4.2.7 Classifier Selectivity

After the evaluation of the basic and the generalized cascade classifier version, the selectivity of both model version is studied with complex data sets. The classifier selectivity is measured in terms of correctly predicted infeasible test examples close to the class boundary. Since the energy data sets all employ infeasible examples, that are not located close to the class boundary, special infeasible test examples are employed. Furthermore an artificial data set is used.

The classifier selectivity experiments are conducted with the data set of a coalition of 5 identical μCHPs (C-CHP) and an artificial *Hyperbanana* data set (A-HB2). Since all infeasible default test examples of energy units are classified correctly and they are not located close to the class boundary, another set of infeasible test examples is applied, where the examples are located closer to the class boundary, see Sect. C.2.1. The *Hyperbanana* data set A-HB2 is expected to yield more representative classifier selectivity results than the μCHP coalition, because the infeasible test examples represent different "rings" around the feasible *Hyperbanana* class with different distances to the class boundary.

All experiments are conducted with the default set-up. Furthermore the cascade classifiers are adapted to the given classification tasks according to the observations in the previous experiments. The experiments with the C-CHP data set are done with the generalized cascade classification model and the same parametrization as in the experiment in Sect. 4.2.5, but the infeasible test examples are exchanged. This time the infeasible test data set consists of two data sets. Set 1 employs more similarities with feasible examples than set 2, see Sect. C.2.1. The A-HB2 data set is classified with the basic cascade classification model with knn-param-1. Furthermore the *Hyperbanana* data set is prepared with selection of feasible examples ($\epsilon = 0.005$, $t = 1000$) and generation of artificial infeasible examples ($\epsilon_b = 0.05$).

The resulting TP and TN values of all ensemble members for the μCHP coalition are shown in Fig. 4.30. Since only the infeasible test examples are different, the TP values in Fig. 4.30(a) are identical to the TP values in Fig. 4.27(a). As expectable the TN values of the infeasible examples with more similarity to the feasible test examples, see Fig. 4.30(b), are a bit worse than the TN values of infeasible examples further away from the class boundaries, see Fig. 4.27(b). This trend is also visible between the two infeasible test sets. Set 1 which employs more similarities with the feasible class achieves a bit worse TN values than set 2. The biggest TN loss compared to Fig. 4.27(b) can be observed for the infeasible examples transformed with acf. This TN loss is due to the great similarity in the correlation behavior of

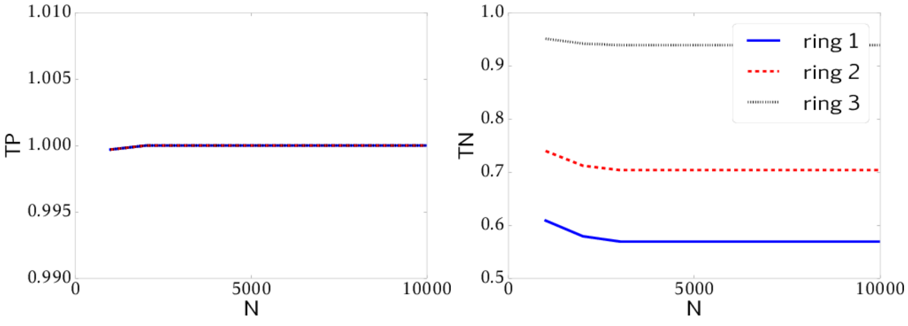


(a) TP (b) TN

Figure 4.30: TP and TN values for a coalition of 5 μCHPs with infeasible test examples near the class boundary. The experiment are identical to the experiments for Fig. 4.27, except from the employed infeasible test examples.

feasible and infeasible test examples.

Next the experiments with the *Hyperbanana* data set are conducted, where the infeasible test examples are distributed in "rings" around the feasible class. The resulting TP and TN values are shown in Fig. 4.31. While the classifier predicts all feasible test examples correctly, the prediction correctness of infeasible test examples increases with an increasing distance to the class boundary. At least more than 50% of the infeasible examples close to the class boundary (ring 1) are predicted correctly. Ring 3 already achieves more then 90% correctly predicted examples.



(a) TP (b) TN

Figure 4.31: TP and TN values for the *Hyperbanana* data set A-HB2, where the infeasible test examples are distributed in "rings" around the feasible class.

Overall the selectivity experiments showed a strongly increasing precision, here increasing TN values for infeasible test examples with an increasing distance to the class boundary. Common infeasible test examples from the energy data sets employ infeasible test examples which are not located close to the class boundary. These infeasible test examples were all predicted correctly in most experiments in this chapter. These observations confirm a more or less slight overestimation of the feasible class, depending on the classification task, as shown in Sect. 4.2.3 e.g., with the example in Fig. 4.16.

4.2.8 Comparative Experimental Study

After the evaluation of the classification precision of the basic and the generalized cascade classification model version and the respective model adaptations one after

classifier	parameters	ensemble combinations
basic cascade cl.	*OCSVM-param-6*	
generalized cascade cl.	acf: *OCSVM-param-1* fft: *OCSVM-param-1* pca: *OCSVM-param-1*	all combinations of weighting and aggregation schemes
classic OCSVM	*OCSVM-param-7*	
SVDD	$C = 1.0$, $\gamma = 1.0$	

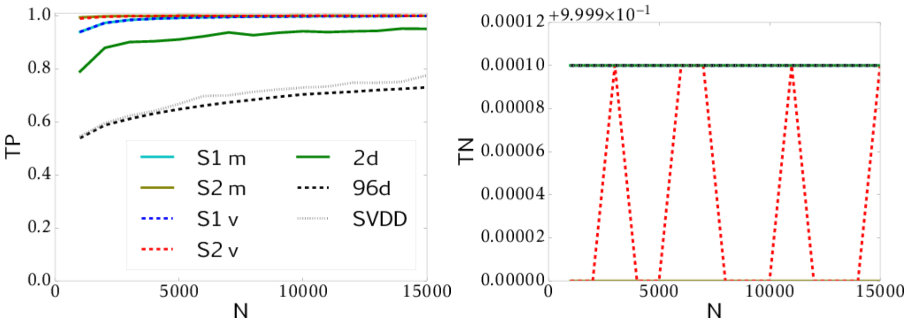
Table 4.10: Classifier parameters for the comparative experimental study on the complete μCHP data set.

another as well as the classifier selectivity, now the different cascade classification model versions are compared against each other. This comparative study is extended by a comparison to common one-class classifiers, a classic OCSVM and a comparable classifier, a SVDD.

A comparison of the basic cascade classification model and a classic OCSVM was already conducted in the baseline classifier choice section, see Sect. 4.2.2. The classic OCSVM achieves a higher precision than the basic cascade classification model, when the feasible class employs a simple data structure. But if the feasible class employs a complex structure, the cascade classification model yields a higher precision than the classic OCSVM. Therefore the classifier comparison experiments in this subsection are conducted on data sets with a more complex structure of the feasible class. The comparison experiments are conducted on two data sets of different complexity and with different distributions of infeasible test examples. These data sets are the complete data set of a single μCHP (B-CHP) and data sets of homogeneous coalitions of identical μCHPs (C-CHP).

All these comparison experiments are conducted with the default classifier set-up. The classification parameters are chosen in pre-test in such a way, that the highest TP values are achieved in combination with TN values always equal to 1. The resulting parameter values and ranges for the experiments on the B-CHP data set are indicated in Tab. 4.10. The experiments on the C-CHP data sets are parametrized in the same way as in the experiment in Sect. 4.2.5, but this time the intermediate results of the generalized classifier are aggregated. The classic OCSVM is parametrized with *OCSVM-param-6*, the basic cascade classifier without transformations (2d) is also parametrized with *OCSVM-param-6* and the SVDD classifier with $C = 1.0$ and $\gamma = 1.4$.

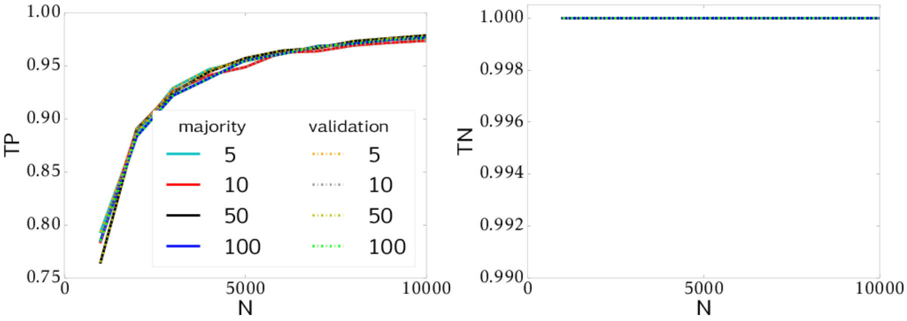
At first the experiments on the B-CHP data set with two feasible concepts are conducted. As expectable, the generalized cascade classification model yields the highest TP values and TN values equals 1, see Fig. 4.32. As already mentioned in the ensemble precision subsection, Sect. 4.2.6, the weighting schemes have hardly any influence on the classification precision, while the aggregation scheme S1 achieves better results than the aggregation scheme S2. The common one-class classifiers, the classic OCSVM (96d) and the SVDD achieve the lowest TP values of all classifiers. The classic OCSVM and the SVDD yield similar results and under certain conditions both classifiers can yield the same results, see Sect. 2.4 and D. M. J. Tax 2001. The basic cascade classification model (2d) yields TP values in between the best and the worst classifiers.



(a) TP (b) TN

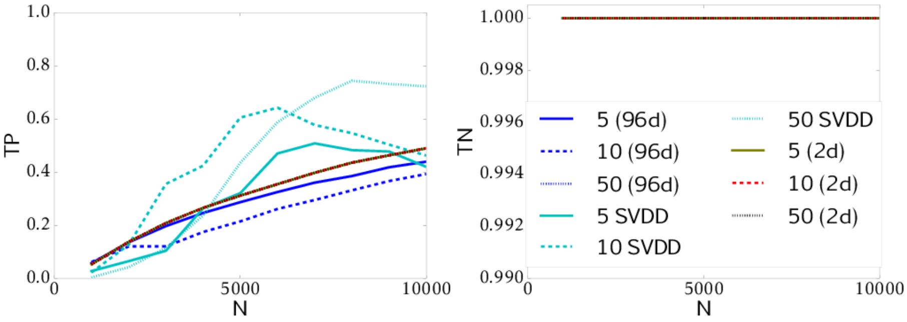
Figure 4.32: TP and TN values on the complete μCHP data set (B-CHP) for different classifiers. The basic cascade classification model is indicated as 2d and the classic OCSVM as 96d. The legend in (a) is also valid for (b).

Next the experiments with the μCHP coalition data sets (C-CHP) are presented, where the feasible class employs a more complex structure than in the B-CHP data set. This time all classifiers yield TN values of 1, because the generalized cascade classification model is only applied with the aggregation scheme S1. The generalized cascade classification model is again more precise than the basic cascade classification model again, see Fig. 4.33. But here the basic cascade classification model achieves similar TP values as the common one-class classifiers, the classic OCSVM



(a) TP S1

(b) TN S1



(c) TP further classifiers

(d) TN further classifiers

Figure 4.33: TP and values achieved on the μCHP coalitions (C-CHP) for different classifiers. In (a) and (b) the ensemble results are shown for aggregation scheme S1. In (c) and (d) the classification results of a classic OCSV (96d), a SVDD and a basic cascade classifier without transformations are shown. The experiments with the basic cascade classification model (2d) and the classic OCSVM for a coalition of 50 μCHPs (96d) yield nearly the same TP values. All TN values in (b) and (d) are equal to 1. The legend in (a) is also valid for (b) and the legend in (d) is also valid for (c). The figures (a) and (b) are identical to (a) and (b) in Fig. 4.29.

and the SVDD. Both common classifiers are expected to yield even more similar TP values, if the classification parameters would have been more fine tuned.

All in all the classifier comparison yielded similar results on both data sets. The generalized cascade classification model achieved the highest precision. On the less complex data set B-CHP the basic cascade classification model outperformed the common one-class classifiers. But on the more complex data sets C-CHP the basic cascade classification model and the common one-class classifiers achieved a similar precision.

The classifier comparison on complex data sets revealed the highest precision for the generalized cascade classification model. A basic cascade classification model achieves a similar or even a higher precision than common one-class classifiers (classic OCSVM and SVDD). Overall the cascade classification model with a careful adaptation is more precise for complex data sets than common one-class classifiers.

4.2.9 Summary

The precision evaluation section focused on the effect of the different cascade classification model adaptations and a comparison of the cascade classification model against common one-class classifiers. The evaluation results are summarized below one after another.

Effects of the Different Model Adaptations

The main effects of the different model adaptations on the classifier precision are presented below.

- **Scalability:** The classification precision of the cascade classification model increases with increasing numbers of training examples. Large numbers of training examples make a high classification precision even possible for data sets with increasing dimensionality. (Data sets with up to 384 dimensions were tested.)

- **Baseline classifier choice:** Baseline classifiers can be adapted to the given data set. An OCSVM is often a good choice as a one-class baseline classifier.

- **Dimensionality of the subsets:** The dimensionality of the low-dimensional subsets is adapted according to the correlation behavior of the feasible class. But the data structure and a potential information loss due to the splitting of the data set need to be considered. If the feasible class does not show the expected correlation behavior (a high correlation between near by time steps and a low correlation for time steps with longer distances), the classification precision can be bad.

- **Preprocessing:** The better the distribution of the training examples, the higher classification precision values can be expected. Inconvenient data distributions (less representative and inhomogeneous) can be improved with the presented data preprocessing algorithms: selection of feasible examples and generation of artificial infeasible examples. Both algorithms require a careful parametrization.

- **Ensembles:** The ensembles precision of the generalized cascade classification model relies on the ensemble preparation and the aggregation of all ensemble members. The preparation of the ensemble members consists in an appropriate choice of transformations, which can simplify a complex data structure of the feasible class and lead to a high classification precision. The overall precision of all well prepared ensemble members depends mainly on the choice of an aggregation scheme. Weighting schemes and the data set complexity showed hardly any effect on the overall precision.

- **Selectivity:**The basic cascade classification model and also the ensemble members of the generalized cascade classifier do not predict all infeasible test examples near the class boundary correctly. But with an increasing distance to the class boundary, the prediction precision increases strongly. Nearly all infeasible test examples, that are not located close the class boundary, are predicted correctly. These results indicate a slight overestimation of the feasible class of the cascade classification model.

Classifier Comparison

The precision comparison of the basic and the generalized cascade classifier to common one-class classifiers revealed a dependence on the classification task and the data set complexity. On classification tasks with a simple data structure, a common OCSVM and a SVDD perform much better than the basic cascade classification model. But on more complex data sets the basic cascade classification model achieves the same or a higher precision than two one-class classifiers. On complex data sets, the generalized cascade classification model is even more precise than the basic cascade classification model.

Conclusion of the Precision Evaluation

All in all the precision evaluation yielded a high classification precision for the cascade classification model on most classification tasks. But if the requirements for the application of the cascade classification model, see Tab. 3.1 are not fulfilled, the resulting classification precision might be bad. But if all requirements are met the following trend can be observed. The better the cascade classification model is adapted to a given classification task and the respective data set, the more precise results can be achieved.

Furthermore the data set complexity has a great impact on the achievable classification precision. While common one-class classifiers (classic OCSVM and SVDD) achieve a higher precision on data sets with a simple data structure, the cascade classification model is superior on data sets with a complex data structure.

4.3 Parameter Sensitivity and Model Parametrization

The analysis of the effect of model adaptations on the classification precision revealed different dependences. In this section some of these dependences are analyzed quantitatively. Overall baseline classifier improvements as well as data set improvements can strongly increase the classification precision. During the qualitative analysis of the cascade classification model adaptations, especially the baseline classifier parametrization and the parametrization of the data preprocessing algorithms turned out to be sensitive to the parametrization. In this section the parameter sensitivity of the baseline classifier parametrization is studied in Sect. 4.3.1 and the data preprocessing parametrization in Sect. 4.3.2.

The Sensitivity analysis (SA) is done according to the six steps, introduced in Sect. 2.5.2. Therefore the influence of the interesting parameters is considered in sensible parameter ranges. These parameter ranges are sampled equidistantly or only some very interesting values are chosen. Then the classification precision is computed in terms of decision boundaries or TP and TN values. Finally the resulting precision is evaluated graphically and parameter fitting suggestions are derived.

4.3.1 Baseline Classifier Parametrization

Already the baseline classifier choice can have a great influence on the classifier precision as shown in Sect. 4.2.2. But the baseline classifier parametrization is also very important. In general parametrization advice for potential baseline classifiers can be found in literature or machine leaning package manuals. In this section not only the effect of the parametrization on a single low-dimensional classifier, but also the effect on the parametrization of a cascade of low-dimensional baseline classifiers as in the basic cascade classification model is of interest. This study is done exemplarily for an OCSVM baseline classifier, because an OCSVM yielded the highest precision in most of the baseline classifier comparison experiments in Sect. 4.2.2.

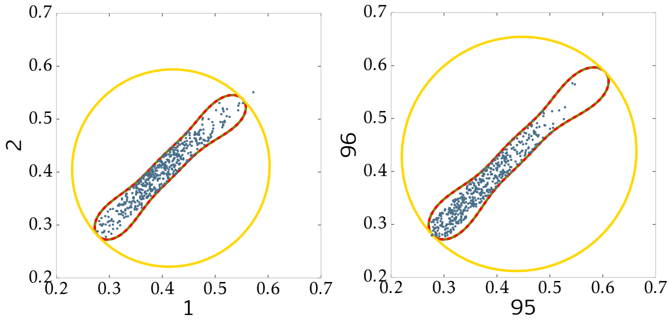
Commonly optimized parameters of an OCSVM include a kernel choice and the optimization of the kernel parameter γ and the margin parameter ν. The most common kernels are linear-, polynomial- and radial basis function kernels (rbf). The OCSVM parameters are restricted to $\gamma > 0$ and $\nu \in (0, 1]$.

Since small improvements of the low-dimensional classifiers of the cascade, like e.g., the baseline classifier choice lead to an improvement of the overall classification precision, baseline classifier parametrization is expected to have a great influence on the cascade classification model precision. The experiments are conducted with the default setup for the basic cascade classification model on the reduced μCHP data set (A-CHP) with $N = \{1000, 2000, \ldots, 15000\}$ training examples.

In a first experiment the three kernels are optimally parametrized with grid search and the resulting low-dimensional classification boundaries are compared. In a second experiment the resulting classification boundaries and the overall classification precision of a rbf kernel are compared for an optimized parametrization, a default parameterization and also a partly optimized parameterization.

Kernel Choice

The first experiments with different OCSVM kernels result in different decision boundaries on the low-dimensional subsets. In Fig. 4.34 the decision boundaries of the first and the last subset of the cascade are shown exemplarily. While the decision boundaries learned with the polynomial kernel and the rbf kernel surround the feasible



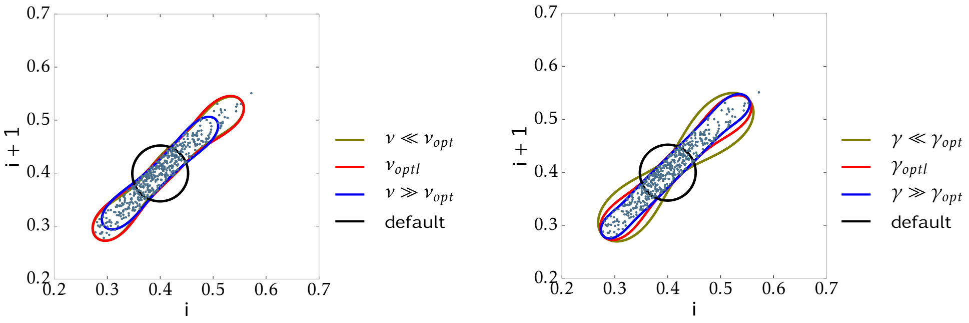
(a) dim 1/2 (b) dim 95/96

Figure 4.34: Decision boundaries for different OCSVM kernels on A-CHP for the first subset of the cascade (a) and the last subset (b). The decision boundaries are indicated with different colors: linear (yellow), rbf (red) and polynomial (olive). The olive line is hidden by the red line.

examples closely, the decision boundary learned with a linear kernel overestimates the feasible class strongly. Since the polynomial and the rbf kernel lead to similar results and the rbf kernel employs less parameters, the rbf kernel is used in the following experiments.

Parametrization of One Low-Dimensional Baseline Classifier

For the rbf kernel the parameters ν and γ have to be optimized. The parameter ν is an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors. The parameter γ is a kernel coefficient. The effect of both parameters on the 2-dimensional decision boundaries of the cascade classifier subsets is estimated experimentally and the results are shown in Fig. 4.35. In the presented example the default parametrization of the machine learning package leads to the worst decision boundaries and an optimization of both parameters yields the best ones.



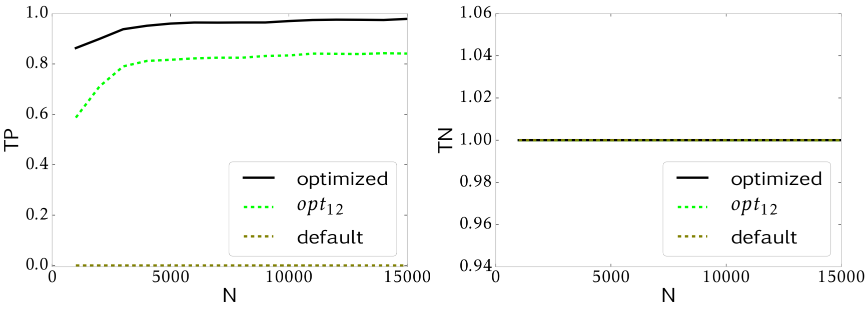
(a) ν (b) γ

Figure 4.35: 1st and 2nd time step of the reduced μCHP data set learned with the rbf kernel and different ν and γ values. The experiments were done with $N = 1000$ training examples.

Parametrization of all Low-Dimensional Baseline Classifiers of the Basic Cascade Classification Model

Next the effect of the baseline classifier parametrization for the low-dimensional subsets on the TP and TN values of the whole cascade is analyzed. The classification is done with three different parametrization sets: default parametrization (default) ($\nu = 0.5$, $\gamma = 1/n_{features} = 1/2 = 0.5$), optimization of all small classifiers (optimized) (*OCSVM-param-1*) and partial optimization of the parameters (opt_{12}) with ($\nu_{optimal} = 0.002$, $\gamma_{optimal} = 70$)). This partial optimization means the optimized parameters of the classifier of the first subset are applied to all classifiers of the cascade. Since all low-dimensional subsets employ a similar data structure they might be learnable with the same parametrization.

All parametrization sets lead to TN values of 1 and differ in the TP values, see Fig. 4.36. The default parametrization leads to the worst TP values of about 0. The



(a) TP (b) TN

Figure 4.36: TP and TN values of the basic cascade classification model on A-CHP with different parametrizations of an OCSVM baseline classifier and increasing values of N.

application of the optimized parameters of the first classifier of the cascade (opt_{12}) for all classifiers leads to relatively high TP values, but a parameter optimization for each low-dimensional classifier leads to the highest TP values.

The experimental results show, a careful parametrization of all low-dimensional classifiers (baseline classifiers) can strongly increase the cascade classification model precision in comparison to the default parameterization of the machine learning package. As already shown in Sect. 4.2.9 small improvements of the low-dimensional classifiers improve the overall classification precision.

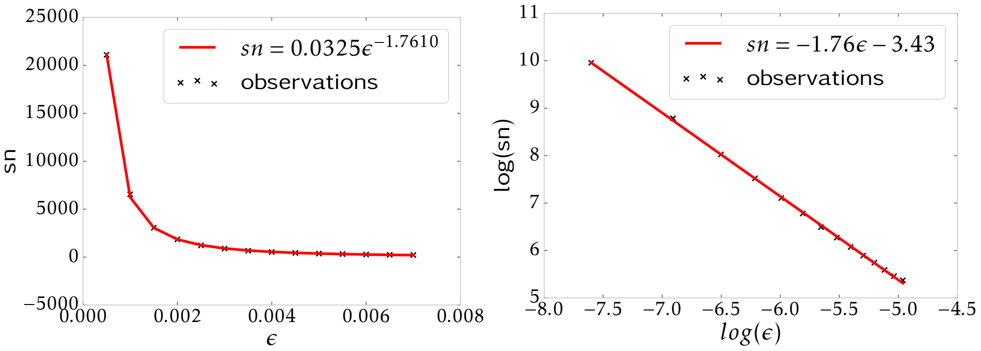
4.3.2 Data Preprocessing Parametrization

Beside classifier improvement also data improvements increase the classification precision. The qualitative effect of the data preprocessing methods, selection of feasible examples and generation of artificial infeasible examples has already been studied in Sect. 4.2.4. Depending on the data set and the parameter tuning both preprocessing methods can greatly improve the precision of the cascade classification model. Since the parametrization of the two preprocessing methods employs parameter dependencies and requires a careful tuning, the parametrization is now analyzed quantitatively. This analysis is conducted exemplarily on the reduced μCHP data set (A-CHP) with low-dimensional subsets of p=2. First off all the parametrization of the selection of feasible examples is studied. Then the parametrization of the generation of artificial infeasible examples is analyzed. After that different combinations of the dependent parameters of both methods are studied experimentally to derive parameter fitting suggestions.

Parametrization of Selection of Feasible Examples

The parametrization of selection of feasible examples depends on the given data set and the expected properties for the selected feasible examples. The given A-CHP data set with low-dimensional subsets of p=2 employ power production values of $[0,1]$ for each time step. The feasible examples are more or less inhomogeneously distributed in each low-dimensional data subset.

Since the number of selected feasible examples varies among the low-dimensional subsets and baseline classifiers require a certain number of training examples, the application context of the preprocessed data set is relevant. Here the selected feasible examples should be applied in a basic cascade classification model with a knn baseline classifier. The qualitative study in Sect. 4.2.4 yielded good classification results for $N \geq 250$ training examples. This limits the number of selected feasible examples sn, which depends on the minimal distance ϵ between selected feasible examples. Therefore the smallest number sn of all low-dimensional data subsets, resulting from the same ϵ value, is used for all low-dimensional data subsets. The resulting sn values for the A-CHP data set are shown in Fig. 4.37 for increasing values of ϵ and $t = 100$. Overall the adapted number of selected examples sn decreases for increasing values of ϵ with a power function: $sn = 0.0325\epsilon^{-1.7610}$, see Fig. 4.37.



(a) functional relation between ϵ and sn

(b) loglog plot of the functional relation between ϵ and sn

Figure 4.37: Functional relation between the minimal distance between nearest feasible neighbors ϵ and the number of selected examples sn of the subset with the fewest selected examples. The selected feasible examples have to be divided into training and validation sets or can be used only as training data when the validation examples are taken from the remaining feasible examples, that were not selected.

With respect to the minimal number of feasible training examples $N = 250$ and the constant parametrization of $t = 100$, the number of feasible training examples is chosen as $N = \{100, \ldots, 10000\}$ examples for the experimental study. Since training and validation examples should be taken from the selected feasible examples $sn = N \cdot 2$. These numbers of feasible training examples and the respective sn values correspond to $\epsilon \in \{0.001, 0.0015, 0.002, \ldots, 0.0055\}$.

Parametrization of Generation of Artificial Infeasible Examples

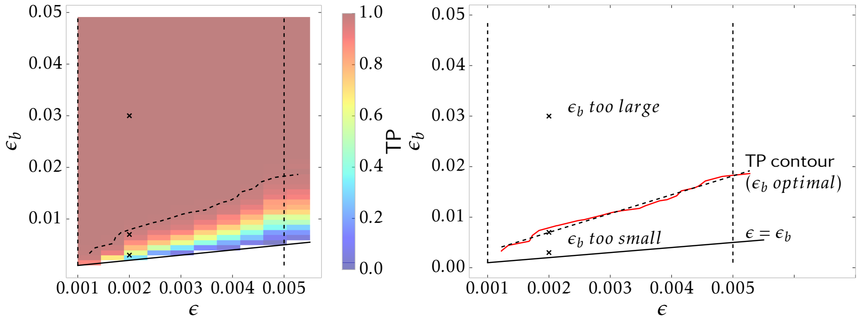
Based on the sets of sn selected feasible examples S for each low-dimensional data subset, artificial infeasible examples can be generated. Generation of artificial infeasible examples is parametrized as follows. Noise is taken from the normal distribution $\mathcal{N}(\mu, \sigma) \cdot \alpha$ with $\mu = 0$, $\sigma = 0.01$ and $\alpha = 1$. The minimal distance between feasible and artificial infeasible examples ϵ_b is increased for all data sets generated with the different ϵ values. The value for ϵ_b is chosen from $\{0.001, 0.002, \ldots, 0.05\}$ with $\epsilon_b \geq \epsilon$. Depending on ϵ_b the number of algorithm iterations is adapted until at least the same number of artificial infeasible examples are generated as the respective number of selected feasible examples sn.

Study of the Effect of Different Preprocessing Parameter Combinations

The parameter sensitivity study of classification experiments is done on the differently preprocessed data sets. The classification is done with the binary baseline classifier knn as in Sect. 4.2.4. The training and validation sets consist of $N = sn/2$ feasible and N artificial infeasible examples. Furthermore the experiments are conducted according to the basic cascade classification model setup with data preprocessing. Then the cascade classifier precision is evaluated graphically on the achieved TP and TN values according to the underlying data preprocessing distance parameter value combinations.

The cascade classifier precision yielded different TP values for the differently preprocessed data sets shown in Fig. 4.38(a) and always TN values equal to 1. The high TN values are due to the location of the infeasible high-dimensional test examples not close to the class boundary, see Sect. 4.1.1. Based on the TP values different regions of classification precision can be identified and separated by the restrictions resulting from the distance parameters, see Fig. 4.38(b). These regions of different distance parameter combinations are bounded in the ϵ range and the ϵ_b range. The ϵ range is bounded by the number of selected feasible examples. After excluding insensible ϵ values ($\epsilon < 0.001$ and $\epsilon > 0.0055$), the ϵ_b values can be divided into three groups, too small ϵ_b values, optimal ϵ_b values and too large ϵ_b values. For each of these groups the distribution of the preprocessed data sets and the learned decision boundaries are shown exemplarily in Fig. 4.39.

ϵ_b too small The region of too small ϵ_b values is bounded by $\epsilon_b = \epsilon$ at the lower bound and the TP= 0.99 contour at the upper bound. The TP contour can be approximated for this data set with a linear function via linear regression: $\epsilon_b = 3.7251\epsilon - 0.0005$. In the region of too small ϵ_b values a smaller or larger percentage of the artificial infeasible examples is generated in the region of the feasible class. This can be seen in Fig. 4.39(a) showing some artificial infeasible examples (blue points) between the gray points in the region of feasibles. This mixture of feasible and artificial infeasible examples leads to



(a) TP for different parameter combinations of ϵ and ϵ_b

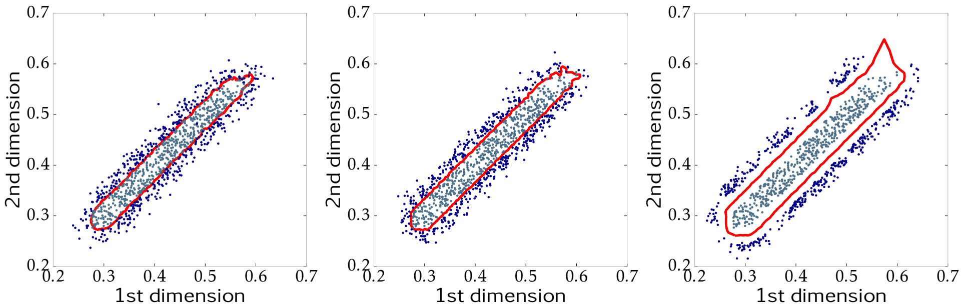
(b) regions of different parameter combinations with different effects

Figure 4.38: TP values for the different data preprocessing parameter combinations (ϵ, ϵ_b) are indicated in (a). The two vertical dashed black lines mark the range of reasonable ϵ values and the solid black line at the bottom indicates the lowest bound for ϵ_b: $\epsilon = \epsilon_b$. The black line above is the contour with TP$= 0.99$. The same boundary lines are shown in (b). Furthermore the resulting regions are indicated. The three points in both figures mark the parameter combinations used for Fig. 4.39.

non smooth class boundaries, an underestimation of the feasible class and relatively low TP values.

ϵ_b optimal The region of optimal ϵ_b values is only described by the TP$= 0.99$ contour for the A-CHP data set, because TN is equal to 1 for all parameter value combinations. If there would be infeasible test examples near the true class boundary, the TN values should decrease for increasing ϵ_b values. From these decreasing TN values a respective TN contour could be derived. This TN contour could be used in combination with the TP contour to determine the optimal value region. The corresponding artificial infeasible examples are distributed around the feasible ones with a small gap in between the classes and hardly any overlap, see Fig. 4.39(b). Due to the gap feasible and infeasible examples are clearly separable and TP and TN values are high in this region.

ϵ_b too large The region of too large ϵ_b values has a lower bound resulting from the TP contour for the μCHP data set. But the lower bound could be also



(a) to small $\epsilon_b = 0.004$ (b) optimal $\epsilon_b = 0.007$ (c) to large $\epsilon_b = 0.025$

Figure 4.39: Decision boundaries learned for the first and second dimension on A-CHP for ϵ_b values from the different regions and a fixed value of $\epsilon = 0.002$. These parameter value combinations are marked in Fig. 4.38.

determined by a TN contour, if there would be infeasible test examples near the true class boundary. If the ϵ_b value is too large, feasible and infeasible examples are clearly separable and there is a large gap between the examples of both classes, see Fig. 4.39(c). The learned decision boundary is located in the middle of this gap. The larger the gap, the further away is the decision boundary from the feasible examples and the true class boundary. This phenomenon is an overestimation of the feasible class. For this reason infeasible test examples near the true class boundary would be classified as feasible.

The sensitivity analysis showed the dependence of the two data preprocessing distance parameters. Based on the minimal distance between selected feasible examples ϵ a region of optimal minimal distance values between feasible and infeasible examples ϵ_b can be identified. Deviations from the optimal parameter value combinations lead either to an over- or underestimation of the feasible class with decreasing TN values, respectively TP values.

4.3.3 Summary

The overall precision of the cascade classification model relies on the precision of the low-dimensional classifiers. The precision of these low-dimensional classifiers can be tuned based on a classifier optimization and based on data set improvements. In this section the sensitivity of two such classification improvement methods with sensitive parameters was studied. At first the parametrization of the low-dimensional baseline classifiers was studied. After that the sensitivity of data preprocessing parameters was studied.

- **Baseline classifier parametrization:** Even a partly optimization of the baseline classifier parameters can lead to a great precision improvement compared to the default classifier parametrization of the machine learning package. The highest classification precision is achieved if the parameters of all low-dimensional baseline classifiers are optimized.
- **Data preprocessing parametrization:** The first data preprocessing method, selection of feasible examples can be always applied to improve, respectively homogenize the distribution of training examples. To allow a high classification precision, the distance parameter ϵ has to be chosen in pre-test. The second data preprocessing method, generation of artificial infeasible low-dimensional examples can be only applied, if both classes are clearly separable and the distribution of the previously selected feasible examples is relatively homogeneous. Artificial infeasible examples lead only to a precision improvement, when they are generated in the infeasible region near the true class boundary. Since ϵ_b depends on ϵ, for each ϵ value a region of too small-, too large- and optimal ϵ_b values can be determined. The highest classification precision can be achieved with optimal combinations of ϵ and ϵ_b, while too small and too large values of ϵ_b lead to an underestimation, respectively overestimation of the feasible class.

All in all the parametrization of the baseline classifiers and the data preprocessing procedures have a great impact on the classification precision. To achieve the best classification precision it is advisable to start the parametrization with pre-test.

4.4 RUNTIME COMPLEXITY

Beside the classification precision and parameter sensitivity also the runtime plays a major role for the applicability of a classifier. In the case of the cascade classification model, the runtime complexity depends on the adaptations of the model. Since all model adaptations modify and extend the basic cascade classifier, only the runtime of the basic cascade classifier is estimated. But still the basic model employs one variable term, namely the baseline classifier. Therefore the runtime complexity is estimated for arbitrary baseline classifiers and additionally for an OCSVM baseline classifier. First of all an analytic runtime estimation is done in Sect. 4.4.1 followed by an experimental estimation in Sect. 4.4.2. Then the resulting runtime estimation is analyzed with respect to time consuming parts in Sect. 4.4.3. Finally suggestions are derived to make the basic cascade classification model more efficient, also with respect to model adaptations.

4.4.1 Runtime Complexity of the Basic Cascade Classification Model

The analytic runtime estimation of the basic cascade classification model with arbitrary baseline classifiers is conducted in terms of an upper bound, see Sect. 2.5.3, Knuth 1976; Ottmann and Widemayer 2012. First of all a functional relation of the complexity of the basic cascade classifier has to be identified. Then an asymptotic upper bound of the runtime complexity is derived for arbitrary baseline classifiers. Finally the runtime complexity with an OCSVM baseline classifier is estimated.

Functional Relation: The functional relation of the basic cascade classification model is derived with the help of the pseudocode of the basic cascade classification model, see Sect. 4.1.6. The functional relation $g(n_{train}, n_{val}, n_{test}, p, dim)$ is given by

$$\left(\sum_{k=1}^{dim-(p-1)} \Big(\sum_{l=1}^{t_{iter}} z(n_{train}, n_{val}, p)\Big) + z(n_{train}, n_{test}, p)\right) + min(n_{test}, dim, p) \quad (4.1)$$

$$\Leftrightarrow \underbrace{\big(dim - (p-1)\big)(t_{iter} + 1) z(n_{train}, n_{val}, n_{test}, p)}_{g_1} + \underbrace{min(n_{test}, dim, p)}_{g_2}. \quad (4.2)$$

The functional relation contains two summands. The first summand (g_1) describes training validation and testing of the baseline classifier z. The second summand (g_2) describes the aggregation of the intermediate classification results to an overall result.

g_1: **outer loop** The outer loop iterates over all low-dimensional data subsets.

inner loop The inner loop iterates over all gridsearch parameter combinations (t_{iter}).

baseline classifier A call of $z(n_{train}, n_{val}, n_{test}, p)$ consists in building a classifier and predicting a validation or a test subset.

g_2: **minimization** The minimization of predicted class labels is an aggregation of the intermediate classification results to an overall result. The feasible class has the label 1 and the infeasible class the label -1.

This aggregation is based on the python implementation for minimization of lists:

```
// minimum of list A
min = 1
for j in (2,..., length(A)):
    if A[j] < A[min]:
        min =j
minimum = A[min],
```

see [1]. This minimization is done for each test example, thus n_{test} times. Furthermore the minimization complexity depends on the length of the list. Here each list has a length of $dim - (p-1)$.

Based on the functional description g the runtime complexity is commonly estimated in terms of the number of samples $n_{samples}$ and the number of features $n_{features}$. Here both parameters are represented by several parameters. The number of samples is represented as the number of training examples n_{train}, the number of validation examples n_{val} and the number of testing examples n_{test}. For convenience all sample size variables are set to an equal value $n_{train} = n_{val} = n_{test} = N$. From now on the number of samples is N.

---

[1] https://hg.python.org/cpython/file/2.7/Python/bltinmodule.c, last time visited 24.02.2017

Just like the number of samples, the number of features is also represented by several parameters, the dimensionality of the data set dim and the dimensionality of the low-dimensional subsets p. In the following the dependence on features is considered in terms of p. But since p and dim depend on each other with $dim-1 \geq p \geq 2$, dim needs to be considered implicitly.

Based on these assumptions and simplifications g is now

$$g(N,p,dim) = \underbrace{\Big(dim-(p-1)\Big)(t_{iter}+1)z(N,p)}_{g_1} + \underbrace{N \cdot min(dim-(p-1))}_{g_2}. \quad (4.3)$$

In the next subsections the dependence of the two summands g_1 and g_2 on N and p is analyzed separately, before the overall complexity is considered.

Runtime complexity of the first summand g_1

The summand g_1 depends on the one hand on p and on the other hand the baseline classifier z, that employs a dependence on p and N. Since this runtime estimation should be valid for arbitrary baseline classifiers, the baseline classifier dependencies are considered as $\mathcal{O}(z)$.

upper bound:
Assertion: $g_1(p) \in \mathcal{O}(p \cdot z(N,p))$
Proof: $\Leftrightarrow \exists c_1, c_2 > 0 \; \forall p \in \mathbb{N} : g_1 \leq c_1 \cdot \mathcal{O}(p \cdot z(N,p)) + c_2$

$$g_1(p, z(N,p)) = \Big(dim-(p-1)\Big)(t_{iter}+1)z(N,p) \leq c_1 \cdot \mathcal{O}(p \cdot z(N,p)) + c_2 \quad (4.4)$$

$$g_1(p, z(N,p)) = dim(t_{iter}+1)z(N,p) - (p-1)(t_{iter}+1)z(N,p) \quad (4.5)$$

$$\leq 2 \cdot dim(t_{iter}+1)z(N,p) - p(t_{iter}+1)z(N,p) \quad (4.6)$$

The assertion is proved with $c_1 = t_{iter}+1$ and $c_2 = 2 \cdot dim(t_{iter}+1)z(N,p)$, because $dim \leq p+1$ and $t_{iter} \geq 0$ ($t_{iter} = 0$: without gridsearch).

In the limiting cases $p = 2$ and $p = dim-1$ the dependence on p outside from $z(N,p)$ starts to vanish.

Runtime complexity of the second summand g_2

The minimization of lists in python has a runtime complexity of $\mathcal{O}(n_{features})$[2]. Here $n_{features}$, the length of a list correspond to $dim-(p-1)$. Sine this minimization is done N times for all test examples, $\mathcal{O}(N,p)$. In the limiting case $p = dim-1$ the dependence on p starts to vanishes.

Overall Complexity for arbitrary and OCSVM baseline classifiers

Summing up the upper bounds of the complexity estimation and neglecting the limiting cases, the basic cascade classification model employs a complexity of

$$\mathcal{O}(p \cdot z(N,p)) + \mathcal{O}(N,p) \tag{4.7}$$

for arbitrary baseline classifiers. Due to the dependence of the baseline classifiers on N and p, the complexity of the first summand g_1 is higher than the complexity of the second summand g_2. Therefore the overall complexity depends only on the first summand g_1 and can be written as

$$\mathcal{O}(p \cdot z(N,p)). \tag{4.8}$$

An OCSVM baseline classifier with the *scikitlearn* implementation employs e.g. a runtime complexity of $\mathcal{O}(n_{features} \cdot n_{samples}^2)$ to $\mathcal{O}(n_{features} \cdot n_{samples}^3)$[3]. Since this runtime complexity estimations are valid for different SVM implementations, the actual runtime complexity depends on the applied SVM algorithm and the data set. Accordingly the basic cascade classification model with such an OCSVM baseline classifier would lead to a runtime complexity of

$$\mathcal{O}(p^2 \cdot N^2) \text{ to } \mathcal{O}(p^2 \cdot N^3). \tag{4.9}$$

In the case of an OCSVM baseline classifier the number of samples N and the dimensionality of the low-dimensional subsets p seem to increase the runtime. But since p and dim depend on each other, increasing values of p do not necessarily mean an increasing runtime, see g_1 in (4.3). The baseline classifier, $z(N,p)$, here

---

[2] https://wiki.python.org/moin/TimeComplexity, last time visited 24.02.2017

[3] http://scikit-learn.org/stable/modules/svm.html#complexity, last time visited 24.02.2017

the OCSVM depends linearly on p. Therefore z can be written as $a \cdot p$ with $a \in \mathbb{R}$. Now g_1 describes a parabola which is opened to the bottom,

$$f(p) = (dim - (p-1))(t_{iter}+1)ap. \tag{4.10}$$

The worst runtime occurs at the vertex of the parabola. The vertex can be read from the parabola equation after a quadratical extension and some further mathematical transformations,

$$f(p) = (t_{iter}+1)a(-p+\frac{dim+1}{2})^2 - (t_{iter}+1)a\frac{(dim+1)^2}{4}. \tag{4.11}$$

The vertex is located on the x-axis at $\frac{dim+1}{2}$. This means p values about $dim/2$ employ the longest runtime.

These analytical runtime complexity estimations for the basic cascade classification model with an OCSVM baseline classifier are studied experimentally in the next subsection.

4.4.2 Runtime Experiments

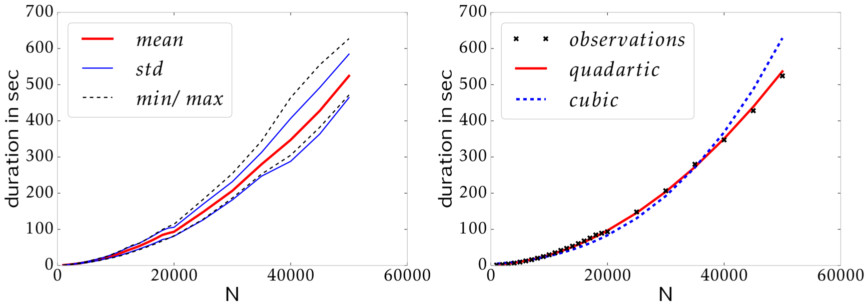
After the analytical runtime complexity estimation, the findings are compared to experimental runtime measurements. Therefore runtime experiments are conducted with respect to N and p separately. All experiments are done on the data set A-CHP with *OCSVM-param-2*. The experiments for increasing values of N are conducted with $N = \{1000, 2000, \ldots, 10{,}000, 20{,}000, \ldots, 50{,}000\}$ and $p = 2$ in 5 experimental runs for each value of N. The experiments for increasing values of p are conducted with $N = 2000$ and $p \in \{2, 5, 10, 15, \ldots, 95\}$ in 50 experimental runs for each value of p.

The runtime is measured each time on one CPU core with 2.26GHz for each classifier of the cascade ($\left(\sum_{l=1}^{t_{iter}} z\right)+z$) including training and validation during gridsearch or training and testing during the prediction stage. The resulting measurements are evaluated as mean runtime values per low-dimensional classifier of the cascade and as the runtime of the complete cascade including the aggregation of the intermediate results g_2. These results are presented at first for increasing values of N and then for increasing values of p.

Runtime Dependence on N

The experiments with increasing numbers of training examples N are evaluated according to the mean runtime of each classifier of the cascade and according to the overall runtime. The mean runtime per classifier of the cascade is shown in Fig. 4.40(a) for increasing values of N. Since the OCSVM employs a complexity between quadratic and cubic, respective regressions are conducted on the mean runtime values per classifier of the cascade. The quadratic regression fits the measurements better than the cubic regression, see Fig. 4.40(b). Thus the runtime depends quadratically on N.

The overall runtime for one basic cascade classifier is about the mean runtime per classifier in the cascade times the number of classifiers per cascade $(dim-(p-1))$ plus the runtime of the aggregation of the intermediate results. The resulting overall runtime is shown in Fig. 4.41. The overall runtime shows a quadratic dependence of the runtime on N as the mean runtime per classifier of the cascade.



(a) mean runtime per classifier (b) regression of mean runtime

Figure 4.40: Measured runtime for each classifier of the cascade t_b for increasing numbers of training examples N: mean, standard deviation and minimum and maximum values (a) mean (observations), quadratic regression and cubic regression (b).

All in all the runtime experiments show a quadratic dependence on N and confirm the analytic results.

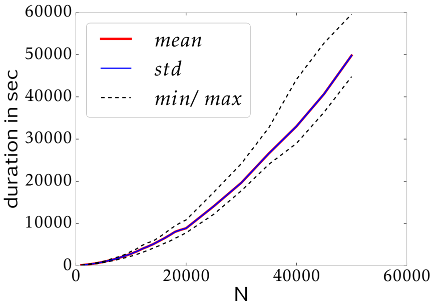


Figure 4.41: Measured mean runtime of all classifiers in the cascade t_{total} for increasing numbers of training examples N. In this figure the lines of the standard deviation cover the line of the mean values.

Runtime Dependence on p

The runtime also depends on the number of features, here indicated in terms of p. The *scikitlearn* OCSVM classifier employs a linear dependence on the number of features and therefore the dependence of the cascade classification model with OCSVM baseline classifiers is quadratic.

The mean measured runtime per low-dimensional classifier of the cascade shows a linear runtime complexity for the OCSVM baseline classifier with increasing values of p, see Fig. 4.42.

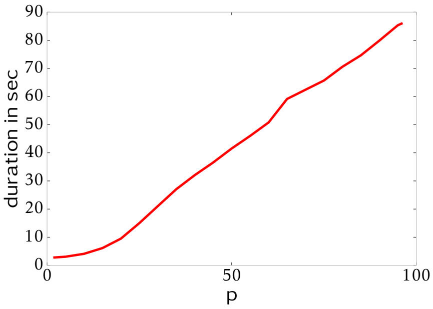
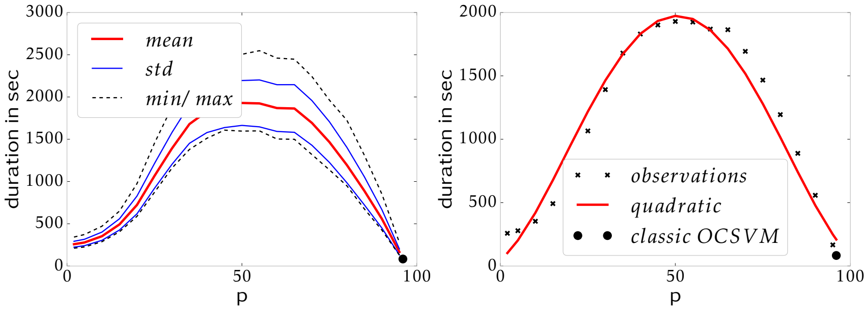


Figure 4.42: Mean runtime t_b of the runtime experiment per baseline classifier for increasing values of p. The duration t_b is a mean value of 50 experimental runs.

Since the basic cascade classification model consists of $dim - (p - 1)$ classifiers, the mean runtime per cascade has to be multiplied with $dim - (p - 1)$ to achieve the overall runtime. The resulting overall runtime depends quadratically on p, see Fig. 4.43. This means, the cascade classifier with very low- or high-dimensional subsets, compared to dim, employs the shortest runtime and subsets with p about $dim/2$ employ the longest overall runtime. Furthermore the resulting runtime for small and high values of p compared to dim is only slightly higher than the runtime of a classic OCSVM. All in all the experimental results approve the analytical results.



(a) mean overall runtime (b) mean overall runtime

Figure 4.43: Total runtime of one cascade classifier t_{total} depending on the number of features p and the runtime of a classic OCSVM (black dot). The duration t_{total} is indicated as: mean, standard deviation and minimum and maximum values (a), mean value and quadratic regression (b).

4.4.3 Summary and Runtime Improvement Suggestions

The analytical runtime complexity estimation as well as the experimental runtime measurements were only done for the basic cascade classification model, because all model adaptations modify or extend the basic model. In this section the runtime complexity results are summed up. After that runtime improvement suggestions are given especially with regard to the complete cascade classification model.

The theoretical runtime complexity estimation and the experimental measurements for increasing values of N and p are summed up below.

- **Theoretical runtime complexity:** For arbitrary baseline classifier $z(N, p)$, the runtime complexity of the basic cascade classification model is $\mathcal{O}(p \cdot z(N, p))$. The application of the *scikitlearn* OCSVM as baseline classifier results in a quadratic to cubic dependence on N and a quadratic dependence on p.

- **Experimental runtime complexity**: The runtime experiments with an OCSVM baseline classifier approve the analytic results. Increasing values of N yielded relatively high runtime lengths per classifier of the cascade, e.g., about 30 seconds for $N = 10000$ and $p = 2$. But low or high p values compared to dim lead to only a slightly higher overall runtime of the complete basic cascade classification model than a classic OCSVM. All remaining p values lead to a longer runtime with a maximum runtime for p about $dim/2$.

In summary, a high classification precision, that depends mainly on the number of training examples and the parametrization of the baseline classifiers can lead to a long runtime. Therefore parallel computing of the classifiers of a cascade is advisable. Especially for the generalized cascade classification model, where a cascade classifier is built on each transformed data set, an efficient computation is needed. Beside a parallel computation further runtime could be saved during parameter optimization of the low-dimensional classifiers. For one thing the parameter combinations that are tested with gridsearch can be limited in pre-test. For another thing gridsearch could be replaced by a more efficient parameter optimization procedure, which might be e.g., an evolutionary algorithm, see Kramer 2014. Finally the benefit of a high classification precision and the respective computational costs have to be judged for each new classification task.

4.5 Discussion of the Evaluation Results

The applicability of the cascade classification model to different classification tasks was evaluated according to the classification performance. This classification performance was assessed in terms of precision in Sect. 4.2, parameter sensitivity in Sect. 4.3 and runtime complexity in Sect. 4.4. Altogether the classification precision depends on one hand on the given classification task and the data set properties. On the other hand the precision depends on the respective adaptation of the cascade classification model. The better the model adaptations, the higher classification model precisions can be achieved.

The cascade classification model achieves a higher precision than common one-class classifiers like a classic OCSVM or a SVDD, if the classification task is complex. Only if the classification task is simple, the common one-class classifiers achieve a higher precision. Even though the cascade classification model achieves a high classification precision, the classifier selectivity experiments indicate a slight overestimation of the feasible class.

Most cascade classification model adaptations mainly require pre-tests and knowledge about the classification task and the data set. Some model adaptations have to be applied carefully due to sensitive parameters, like the baseline classifier parametrization or the data preprocessing parametrization. Furthermore some model adaptations lead to a long runtime, while the basic cascade classification model can employ only slightly more runtime than a classic OCSVM. Nevertheless the different model adaptations are necessary to allow a high classification precision for the whole class of challenging classification tasks, where the cascade classification model was developed for, see Tab. 3.1.

In summary the cascade classification model performed well on all classification tasks related to Tab. 3.1. The more complex the classification task, the better is the resulting precision compared to a classic OCSVM or a SVDD. Classification tasks with different properties than indicated in Tab. 3.1 may result in a low classification precision. After the evaluation of the cascade classification model as a classifier its applicability in VPP scheduling tasks is analyzed in Chapt. 5.

5

APPLICATION OF THE CASCADE CLASSIFICATION MODEL AS FLEXIBILITY DESCRIPTION IN VPP SCHEDULING TASKS

After the evaluation of the developed cascade classification model in terms of classification properties, now the application of this flexibility description in VPP scheduling tasks is considered. VPP scheduling refers to planing the operation of all VPP participants, which can be single energy units or coalitions of energy units, see e.g., Sect. 1.5. From a mathematical perspective, VPP scheduling is a constraint optimization problem (COP). Promising scheduling algorithms treat the COP as a distributed COP (DCOP), see Sect. 1.4. In this context machine learning surrogate models are promising constraint handling techniques, see Sect. 1.3.1. This constraint handling technique consists of machine learning surrogate models, describing the flexibilities and a respective decoder function. The machine learning surrogate models can be e.g., cascade classification models, that describe the flexibilities of all VPP participants. The flexibility of the VPP participants can be accessed via decoder functions. These decoder functions sample feasible operation schedules from the respective flexibility descriptions as possible solutions for the VPP optimization problem. A decoder can be operated according to two strategies. The first strategy consists in the generation of feasible operation schedules, that represent the whole feasible class. The second strategy consists in the search of the nearest feasible operation schedule to a given infeasible one. In this thesis the focus lies on the first strategy.

The application of the cascade classification model for constraint handling in VPP scheduling provides a machine learning surrogate model, but a decoder function is missing. Related work could possibly yield an appropriate decoder function. Such a related constraint handling approach is a decoder based on a SVDD classifier, see

Bremer 2015. But since the respective decoder function is based on high-dimensional support vectors, see Sect. 1.4.2, this decoder approach cannot be applied to the cascade of low-dimensional and dependent classifiers of the cascade classification model. Therefore a new decoder approach is required for the cascade classification model. This new decoder approach needs to handle several low-dimensional and dependent classifiers. Furthermore the adaptability of the cascade classification model leads to different flexibility descriptions. On the one hand the baseline classifier choice influences the description of the low-dimensional subsets (support vectors, density descriptions, etc.). On the other hand the application of the generalized cascade classification model multiplies the number of low-dimensional and dependent classifiers, compared to the basic cascade classifier.

Below two ideas are presented to realize an adaptable decoder approach.

- **low-dimensional and dependent classifiers:** This problem can be solved with a step-wise decoder approach. For each time step a feasible contribution range could be derived in a similar way as proposed in AVPP search space modeling, see Sect. 1.4.3. But the feasible contribution ranges depend on each other. This could be realized with a successive determination of feasible contribution ranges based on the value, chosen from the previous contribution range.

- **adaptability:** This problem resulting from the adaptability of the cascade classification model could be solved with an adaptable decoder approach. Differences in the flexibility descriptions, due to different baseline classifiers do not yield any problems for the determination of feasible ranges. But the application of the basic cascade classifier or the generalized cascade classifier require an adaptable decoder approach. A basic decoder approach according to the basic cascade classification model, see Sect. 3.2 could work according to the idea presented above for low-dimensional and dependent classifiers. For the generalized cascade classification model, see Sect. 3.5 a generalized decoder version needs to be developed. Since each cascade classifier of the ensemble is not very precise and only the ensemble off all cascades achieves a high classification precision, all cascades need to be considered in the generalized decoder version. This could be achieved by building a basic cascade decoder approach for one cascade classifier of the ensemble. After that the feasibility of the new generated time series needs to be predicted with the other cascade classifiers of the ensemble.

Based on the presented solution ideas for an adaptable decoder approach for the cascade classification model, a first attempt for a basic decoder approach is developed and presented. This attempt for a decoder approach is introduced in Sect. 5.1, including the algorithm and an experimental study. Afterwards the applicability of the developed decoder approach is discussed in the context of VPP scheduling, in Sect. 5.2.

5.1 DECODER APPROACH

In this section a first attempt for a basic decoder approach is presented in Sect. 5.1.1 followed by an experimental study and a valuation of that attempt in Sect. 5.1.2.

5.1.1 Algorithm for a Basic Decoder Approach

In a first attempt a basic version of the decoder approach was developed according to the basic cascade classification model, see Sect. 3.2. Beside the applicability of the basic cascade classification model, the feasible class of the considered task may only employ low-dimensional data subsets, which are compact sets. Otherwise a determination of feasible contribution ranges for each time step is difficult. This attempt for a basic decoder approach works for arbitrary baseline classifiers and an arbitrary dimensionality of the low-dimensional data subsets p. Below the decoder approach is presented in Algorithm 3 for $p = 2$. If $p > 2$, the first time steps of a new time series have to be determined differently. The respective algorithm modification is presented later.

The basic decoder approach works as follows for the determination of one new feasible operation schedule, arbitrary baseline classifiers and $p = 2$, see Fig. 5.1. The low-dimensional classifiers c_i (baseline classifiers) of the cascade C of a basic cascade classification model are built on a data set, scaled to values in $[0, 1]$. This scaling simplifies the determination of feasible ranges. Sampling of a new feasible example x_{new} starts with sampling of the first time step. Therefore a grid g is generated, that covers the area of possible feasible values in the interval $[0, 1]$ for the first

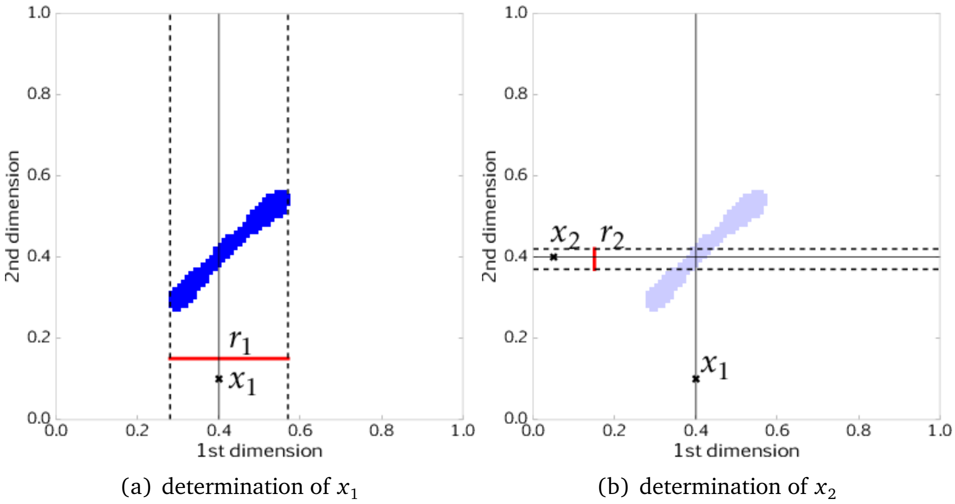
Algorithm 3 Decoder approach for the basic cascade classification model

```
Require: basic cascade classification model $C$ of low-dimensional classifiers $c_i$
         with $i \in \{1, 2, \dots, dim-1\}$ and $p = 2$,
         a baseline classifier implementation with a prediction method
Determination of the first value $x_1$ of $x_{new}$
$g = meshgrid(v, v)$, with $v = \{0, 0+\delta, \dots, 1-\delta, 1\}$ (grid generation)
reshape $g$ to (no. of examples$\times p$)
$pred_1 = c_1.predict(g)$
determine feasible range $r_1$ for the 1st time step from $pred_1$
chose $x_1$ from $r_1$
Determination of $x_{i+1}$ for $i \in \{2, 3, \dots, dim-1\}$
for $i = 1$ to $(dim-1)$ do
    $v_{i+1} = [x_i, v]$
    $pred_i = c_i.predict(v_{i+1})$
    determine feasible range $r_{i+1}$ for the time step $i+1$ from $pred_i$
    chose $x_{i+1}$ from $r_{i+1}$
end for
$x_{new} = [x_1, x_2, \dots, x_{dim-1}]$
```

and the second dimension (time step) with a resolution δ. Then the feasibility of this grid is predicted with the first classifier of the cascade c_1. Based on the resulting predictions $pred_1$ a range r_1 of all feasible values for the first time step is determined, see Fig. 5.1(a). Now an arbitrary feasible value x_1 can be chosen from r_1.

Depending on x_1 a value for the next time step is determined and so on. More general the determination of time step x_{i+1} depends on the value of time step x_i. The only difference of the determination of x_{i+1} to the determination of x_1 consists in the generation of a grid. This time the grid v_{i+1} is composed of the value from the previous time step x_i and a discretization of possible feasible values for the next time step v. Then the feasibility of v_{i+1} is predicted, the range of feasible values r_{i+1} is determined and the next value x_{i+1} is chosen, see Fig. 5.1(b). This procedure is repeated until the new feasible operation schedule x_{new} is completed. To achieve a set of new feasible operation schedules X_{new}, the decoder approach has to be applied several times.

In the following implementation hints and algorithm modifications concerning the choice of the dimensionality p and the sampling strategy for x_1 and x_{i+1} are



(a) determination of x_1 (b) determination of x_2

Figure 5.1: Step-wise determination of a new feasible example with the basic decoder approach at the example of the A-CHP data set: (a) determination of x_1 and (b) determination of x_2. The blue region indicates the grid points of g ($\delta = 0.01$), which were predicted as feasible by the first classifier c_1 of the cascade C.

presented.

Implementation Hints In some cases the approximation of the feasible range x_{i+1} can cause problems. Especially if the previous value x_i was chosen directly at the boundary of the previous range r_i. Then the following range r_{i+1} might be empty and no feasible value is available for x_{i+1}. This problem results from the approximation of the feasible range with a grid and small inaccuracies in the low-dimensional classifiers. To prevent this problem, it is advisable to chose x_{i+1} with at least a small distance to the interval boundaries of r_{i+1}.

If the problem still occurs, the easiest solution is to chose a new value for x_i. Another option is to discard all determined values of x_{new} and to start the determination of x_{new} again.

Decoder Modifications (p) Even though the basic decoder approach was introduced only for low-dimensional classifiers of dimensionality $p = 2$, it is also applicable for other dimensionality values p. But p values are only valid for the basic decoder approach, if all p-dimensional feasible data subsets are compact sets.

The choice of the dimensionality p is important for the determination of the first $p-1$ time steps. The presented grid approach for the determination of the first $p-1$ time steps cannot be applied if $p > 2$, due to the curse of dimensionality. To overcome this problem the first p time steps can be determined alternatively with the related decoder approach by Bremer 2015, see Sect. 1.4.2. Therefore a p-dimensional SVDD classifier is built on the first p time steps. After that the respective decoder approach is applied for the determination of the first p values of x_{new}. Then the basic decoder approach can be applied to determine the value for time step x_{p+1} and so on.

Beside the adaptation of p also the sampling strategy for the choice of new values x_{i+1} from r_{i+1} can be modified.

Decoder Modifications (sampling strategy for x_{i+1}) Even though the proposed decoder approach aims at the generation of representative samples of the whole feasible class, it is also possible to find the nearest feasible operation schedule for a given infeasible one. Therefore the choice of the value x_{i+1} from the feasible range r_{i+1} has to be adapted. Instead of a random value, now the closest feasible value to the corresponding infeasible value of the current time step is chosen.

5.1.2 Experimental Study with the new Decoder Approach

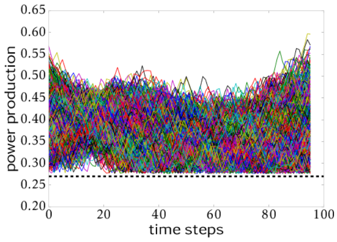
In this subsection the first attempt for a basic decoder approach for the cascade classification model is evaluated experimentally on the μCHP data set with a simple feasible class (A-CHP). Therefore the basic decoder approach is applied to low-dimensional data subsets with $p = 2$ and $p = 20$. The dimensionality $p = 2$ is a default value and $p = 20$ is chosen, because the autocorrelation of the feasible class is about zero for larger distances between the time steps.

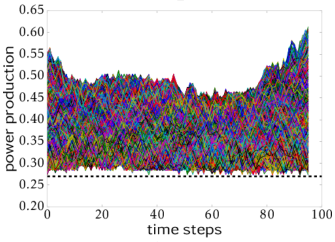
The experiments are set up with the default cascade classifier setup and $N = 1000$ feasible training examples. The experiments with $p = 2$ are parametrized with *OCSVM-param-4* and the experiments with $p = 20$ with *OCSVM-param-8*. Then the basic decoder approach is applied according to Algorithm 3 with a grid resolution of $\delta = 0.01$ to generate 1000 new time series. For $p = 2$ no adaptations are required and for $p = 20$ the first 20 time steps are determined with the python implementation, see Graaff 2015, of the decoder approach of Bremer 2015. The SVDD classifier is parametrized with $C = 1.0$ and $\gamma = 1.0$ and the respective decoder is applied with the default parametrization. This decoder approach approximates given operation schedules. Therefore the decoder experiments are initialized with 1000 different feasible operation schedules.

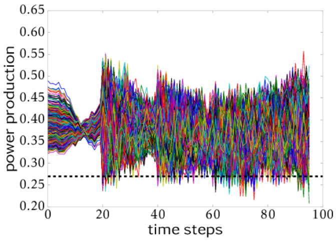
After the generation of new decoder examples their quality needs to be estimated. The simples possibility is to test the feasibility with the μCHP simulation model. But since all new decoder examples are infeasible according to the strict simulation model rules, an estimation of the errors could help to estimate the data quality. Such errors and their detection are application specific. According to the μCHP simulation model parametrization, there are amongst others three properties, that characterize the feasibility of operation schedules. First off all feasible operation schedules employ only a change of the power production of 3% of the maximal power production from one to the next time step. Secondly the minimal power production is limited by a threshold of about 27% of the maximal power production. The third characteristic property is the thermal buffer temperature, corresponding to the power production time series. The simulation model allows the computation of the corresponding thermal buffer temperature for a given power production time series. The thermal buffer may only employ temperature values between 65°C and 75°C.

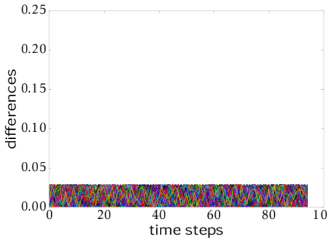
These three properties are shown in the figures in Tab. 5.1 for the training set and the new decoder examples with $p = 2$ and $p = 20$. The training examples are feasible time series and show the expected properties of feasibility.

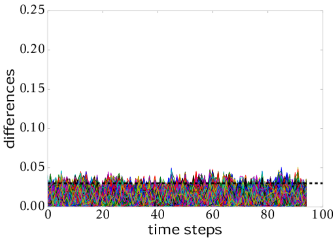
The new decoder examples of both p values deviate more or less from feasible operation schedules in the three considered properties. While $p = 2$ does not lead to any violations of the minimal power production threshold of 27%, see Tab. 5.1(b), $p = 20$ leads to violations, see Tab. 5.1(c). Concerning the other properties, power production changes between neighboring time steps and the thermal buffer temperature, the new decoder examples violate one of these properties strongly and the other property less strongly. Additionally the decoder examples generated with $p = 20$ show different properties for the first 20 time steps, than for the other time steps, see Tab. 5.1(c, f, i). This difference is due to the application of a different decoder approach for the first 20 time steps. Since only the proposed basic decoder approach for the cascade classification model should be evaluated, the first 20 time steps are ignored.

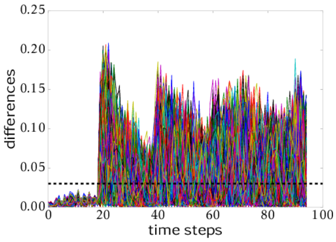
<table>
<tr><th>feasible training examples</th><th>decoder examples ($p = 2$)</th><th>decoder examples ($p = 20$)</th></tr>
<tr><td colspan="3">time series representation</td></tr>
<tr>
<td>

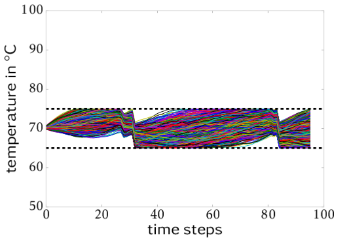
(a)</td>
<td>

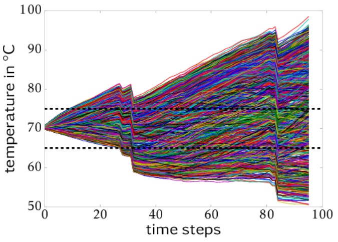
(b)</td>
<td>

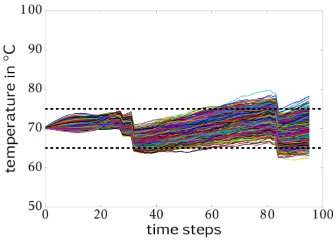
(c)</td>
</tr>
<tr><td colspan="3">power production changes between two time steps</td></tr>
<tr>
<td>

(d)</td>
<td>

(e)</td>
<td>

(f)</td>
</tr>
<tr><td colspan="3">corresponding thermal buffer temperature</td></tr>
<tr>
<td>

(g)</td>
<td>

(h)</td>
<td>

(i)</td>
</tr>
</table>

Table 5.1: Properties of new decoder examples in comparison to the training examples. The different properties show different errors in the decoder examples. Additionally the black dashed lines indicate thresholds and feasible ranges.

All together the new decoder examples generated with $p = 2$ and $p = 20$ differ far too much from feasible operation schedules. These deviations are much bigger than small inaccuracies, compared to the feasible training data set without any deviations, see Tab. 5.1(a, d, g). For a better understanding of the errors in the decoder samples an error analysis is conducted in the following subsection.

Analysis of the Experimental Results

All in all the experiments with the attempt for a basic decoder approach for both p values yield new operation schedules with more or less big errors with regard to the three considered properties. To identify the source of these errors potential error sources in the basic cascade classification model and in the basic decoder approach have to be considered. Beside these error sources also the error propagation from the classification model to the decoder is important. Both parts, the basic cascade classification model and the basic decoder approach are analyzed one after another.

Classification Errors The basic cascade classification model has mainly two error sources. First of all the model requirements, see Tab. 3.1 have to be fulfilled and secondly a good classification precision relies on a good model adaptation and parametrization.

- **Model Requirements:** If not all model requirements and assumptions are fulfilled, they can lead to a decreased classification precision, because the cascade classification model was developed for classification tasks with special properties. The main requirements are a high-dimensional classification task with severely imbalanced classes. Furthermore the feasible examples should employ a high correlation between neighboring time steps and the correlation should decrease strongly for longer distances between the time steps. This correlation requirement implies an appropriate choice of p for the dimensionality of the low-dimensional data subsets. Small values of p carry a risk to neglect important information and large values of p carry the risk to be affected by the curse of dimensionality.

- **Model Adaptations and Parametrization:** The better the basic cascade classification model is adapted to a given classification task and the respective data set, the higher is the achievable classification precision. But some pre-tests have shown, that an underestimation of the feasible class causes less errors in new decoder time series, than a parametrization according to the most precise classification results. This phenomenon is due to slight overestimations of the feasible class, see e.g., Sect. 4.2.7.

Beside these error sources also the amount and the quality of the available data influence the precision. Furthermore the length of the considered operation schedules is important, because the classification precision generally decreases with increasing operation schedule length.

Since the attempt for a basic decoder is based on the basic cascade classification model the errors from the classification model will propagate.

Decoder Errors The decoder model has mainly two error sources, the model requirements have to be fulfilled and the propagation of errors resulting from the classification model.

- **Model Requirements:** The applicability of the decoder approach relies on the applicability of the basic cascade classification model, respectively the cascade classification model requirements. Furthermore all low-dimensional feasible data subsets have to be compact sets, to allow the determination of feasible contribution ranges for all time steps.

- **Error Propagation:** If the basic cascade classification model employs inaccuracies, they influence the range of new possible values for each time step in the decoder model. Furthermore it can be expected that errors in a new decoder time series propagate towards the last time steps of the time series, which is e.g., visible in the thermal buffer temperature. In figure (h) and figure (i) in Tab. 5.1 the deviation from the feasible temperature range increases with the index of the considered time step.

Beside the error sources in the attempt for a basic decoder, also the feasibility estimation of the decoder samples can lead to errors, if the considered data properties are e.g., not representative or incomplete as feasibility measures.

Discussion of Classification and Decoder Errors The basic cascade classification model employs error sources as well as the attempt for a basic decoder approach and the errors can propagate. For a more precise estimation of the error sources and the error propagation a small classification test is conducted. For this purpose basic cascade classifiers are set up as in the experimental study in Sect. 5.1.2 but with different values of p. These classifiers predict the infeasible test set of A-CHP correctly. After that the feasibility of the decoder samples with p=2 and p=20 is predicted. Depending on the choice of p a more or less large percentage of the decoder samples is predicted as feasible. If p is equal to the p value of the decoder samples, of course all decoder samples are predicted as feasible. Nevertheless all decoder samples are infeasible, but they employ some similarities with feasible operation schedules. Since the infeasible test set from A-CHP is always predicted correctly and the test examples are not located close to the class boundary, the decoder samples have to be located close to the class boundary.

This observation corresponds to the findings in the selectivity study in Sect. 4.2.7, where a slight overestimation of the feasible class was detected. This means the basic decoder attempt requires a basic cascade classification model with a higher selectivity and correspondingly a lower overestimation of the feasible class.

So far only the basic cascade classification model was considered for a decoder approach. But there is also a generalized version of the cascade classification model, which is in general more precise than the basic version. According to the generalization of the basic cascade classification model, see Sect. 3.5, a generalized decoder approach is required. But since the first attempt for a basic decoder approach employs already too many inaccuracies, it could be helpful to solved them, before a generalized decoder approach is developed. Furthermore the basic decoder attempt does not offer a simple possibility to integrate an ensemble of cascade classifiers, based on different time series transformations. Due to the different requirements for a basic and a generalized decoder approach resulting from the basic and the generalized cascade classifier (time series transformations and the number of dependent classifiers), a generalized decoder approach could probably be more precise than a basic version. Then a basic and a generalized decoder approach could be developed separately. Finally the development of a generalized decoder approach remains open for further research as well as the development of a precise basic decoder approach.

5.2 Remarks on the Application of the Developed Flexibility Description in VPP Scheduling Tasks

The developed cascade classification model yields flexibility descriptions of single energy units and also coalitions of energy units. But the flexibilities in that descriptions are only accessible for VPP scheduling algorithms via a decoder function. This decoder function generates feasible operation schedules from the flexibility description. Since no respective decoder is available in literature for an adaptable classifier, a respective decoder had to be developed. Therefore a first attempt to a basic decoder approach was proposed according to the basic cascade classification model. This decoder function samples time series from the flexibility description as possible solutions to the optimization problem of the scheduling task.

An experimental study with the basic decoder approach yielded not a single feasible new operation schedule sample from the flexibility description. For a further estimation of the feasibility of new decoder samples their deviations from feasible operation schedules were considered. The resulting deviations are more or less big. Even a careful parametrization of the underlying cascade classifier and the decoder do not strongly decrease the errors in the decoder samples. Therefore an error analysis was conducted. The cascade classifier and the basic decoder approach employ error sources and the errors can propagate. A main error source is the selectivity of the basic cascade classification model, which is not sufficiently high. This becomes obvious in classification tests, because infeasible test examples near the class boundary are only partly predicted correctly. These results indicate an overestimation of the feasible class, see also Sect. 4.2.7. This overestimation of the feasible class results from splitting the high-dimensional classification task into several low-dimensional ones. Even though this classification task splitting can introduce errors in terms of an information loss, this splitting alleviates the effect of the curse of dimensionality. Altogether there is a trade off between an information loss and the effect of the curse of dimensionality.

Due to these problems with the basic decoder approach and problems to integrate an ensemble of cascade classifiers into the basic decoder attempt no generalized decoder version is developed according to the generalized cascade classification model.

All in all the proposed attempt for a basic decoder approach is not precise enough, because new decoder examples employ too big errors for the application in VPP scheduling tasks. Therefore a new decoder approach is required. Even though the cascade classification model performed well in the classification experiments, see e.g., Sect. 4.2.8, probably the cascade classification model needs to be adapted to a decoder. Furthermore a basic and a generalized decoder might require different cascade classification model adaptations, like e.g., the transformations in the generalized cascade classifier. Finally the application of the cascade classification model as flexibility description in VPP scheduling algorithms relies on a precise decoder to solve VPP scheduling tasks.

6 FINAL REVIEW

VPP scheduling relies on the flexibilities of the VPP participants. Therefore VPP scheduling algorithms require flexibility descriptions. Since there are no flexibility descriptions available, that are applicable to different kinds of single energy units and also coalitions of energy units a new flexibility description is required. This flexibility description should be abstract, consistent and precise according to the aim of this thesis, see Sect. 1.5.

This thesis consists in the development and the evaluation of such a flexibility description. In this chapter this thesis is completed with a final summary and a conclusion in Sect. 6.1 and a perspective in Sect. 6.2.

6.1 Summary and Conclusion

Since the flexibility description should be abstract, consistent and precise for the application in VPP scheduling tasks, machine learning surrogate models are appropriate descriptions of the quantity of all alternatively realizable operation schedules. Appropriate machine learning models depend on the given task and the given data set. The quantities of all alternatively feasible operation schedules of different kinds of single energy units and coalitions of energy units form a class of high-dimensional time series classification tasks with severely imbalanced classes. Beside these classification task properties, also data set properties play an important role in the choice of an appropriate classifier. Operation schedule data sets always contain feasible operation schedules and infeasible ones are not always available. Additionally the representativity and the distribution of the operation schedule examples of both classes can differ. Beside these observed data set properties, it is assumed that the data sets employ complete and correct labels and no noisy examples.

In literature there are only classifiers that fit to some of the identified classification task and data set properties, but no classifier is available that fits to all properties. Therefore a further analysis was conducted to identify properties which can be exploited to simplify the class of classification tasks. Two such properties could be identified. For one thing feasible operation schedules employ a special correlation behavior. Neighboring time steps employ a high correlation and the correlation decreases with increasing distance between two time steps. Since most information is encoded in neighboring time steps, the high-dimensional classification task can be split into several low-dimensional ones without a great information loss. The resulting new classification tasks are only low-dimensional time series classification tasks with more or less severely imbalanced classes. Especially one-class classifiers can solve these new classification tasks. For another thing the low-dimensional classification tasks employ a similar structure. This similarity saves classification adaptation effort, because all low-dimensional classification tasks can be solved with similar classifiers.

Based on these two classification task simplification properties a classifier, called cascade classification model, was developed. This classifier consists of a cascade of overlapping feature classifiers. The cascade classification model is adaptable to different classification tasks and data sets, due to a basic and a generalized version with several model adaptation possibilities and respective data preprocessing methods.

After the classifier development, the cascade classification model was evaluated at first as a classifier. Then the applicability as flexibility description in VPP scheduling tasks was analyzed. The respective results are summarized in Sect. 6.1.1 and Sect. 6.1.2.

6.1.1 Evaluation Results of the Cascade Classification Model

The cascade classification model was evaluated in terms of precision, parameter sensitivity and temporal complexity. Overall the cascade classification model performed well in different classification tasks, that fulfill the model requirements, see Tab. 3.1. The better the cascade classification model is adapted to a given classification task and the respective data set, the higher is the achievable precision. Furthermore the cascade classification model achieves a higher precision than common one-class classifiers, like a classic OCSVM and a SVDD classifier, if the classification task is more complex. Only classification tasks with a simple data structure lead to a higher precision of common one-class classifiers than the cascade classification model. Despite a high classification precision of the cascade classification model, the classifier selectivity indicates a slight overestimation of the feasible class. This means most classification errors occur close to the class boundary. Infeasible operation schedules that are located close to the class boundary employ only small deviations from feasible operation schedules. In VPP scheduling tasks large deviations from feasible operation schedules are worse than smaller deviations. But finally the consequences of misclassification have to be judged for each application individually.

The costs for the high classification precision are more or less sensitive model adaptations and data preprocessing methods, that require pre-tests. Already the basic cascade classification model requires a higher computation time than a classic OCSVM. The generalized cascade classification model requires even a multiple of the computation time of a basic cascade classifier, because the ensembles consists of several cascade classification models.

With regard to the aim of this thesis the classifier evaluation showed the achievement of the required characteristics of a flexibility description: abstract, consistent and precise.

abstract The cascade classification model yields an abstract description of the quantity of all alternatively realizable operation schedules. The cascade of dependent and low-dimensional classifiers describe overlapping operation schedule segments e.g., in terms of support vectors or density descriptions, depending on the baseline classifier.

consistent The cascade classification model yields a consistent flexibility description, because it describes the flexibilities of different kinds of single energy units as well as the flexibilities of coalitions of energy units. Additionally the scheduling horizon and the temporal resolution of the classification model are adaptable.

precise The cascade classification model achieved a high precision in the different experiments with data sets of single energy units, coalitions of energy units and also artificial classification tasks. The better the cascade classification model is adapted to a given classification task, the higher is the achievable precision.

Especially on complex classification tasks, the cascade classification model performs better than common one-class classifiers, like a classic OCSVM or a SVDD. Despite precise operation schedule classification results for single energy units and coalitions of energy units, the cascade classification model leads to a slight overestimation of the feasible class.

6.1.2 Applicability Results of the Developed Flexibility Description in VPP Scheduling Tasks

As the cascade classification model evaluation showed, the resulting flexibility descriptions fulfill all requirements for an application in VPP scheduling tasks. But the application of the flexibility description in VPP scheduling algorithms requires a decoder function to access the flexibility from the flexibility descriptions. Such decoder functions sample operation schedules from the flexibility descriptions as possible solutions to the optimization problem in scheduling tasks. Since no adaptable decoder approach could be found in literature, a first attempt for a respective decoder approach was developed for the cascade classification model. This decoder approach computes feasible ranges for each time step, based on the previous time steps. An experimental analysis with that basic decoder approach yielded for a single μCHP only infeasible decoder samples. The decoder samples employ too big errors, because the selectivity of the (basic) cascade classification model is not high enough, due to a slight overestimation of the feasible class. Even though the (basic) cascade classification model performs well in the classification experiments, see e.g., Sect. 4.2.8, the overestimation of the feasible class is a problem for the proposed basic decoder approach. Additionally the errors from the basic cascade classification model propagate in the decoder attempt. So far only a basic decoder approach was considered for the basic cascade classification model. But the generalized cascade classification model is generally more precise than the basic version. Nevertheless the problems of a basic decoder approach are also relevant for a generalized decoder. Furthermore the integration of an ensemble of cascade classification models based on time series transformations is problematic for a generalized decoder approach.

All in all the application of the cascade classification model as flexibility description in VPP scheduling tasks requires a new decoder approach and possibly also an adaptation of the cascade classification model.

6.1.3 Conclusion

Overall the proposed cascade classification model is an abstract, consistent and precise flexibility description for VPP scheduling tasks, where the VPP participants can be different kinds of single energy units or coalitions of energy units. For most VPP participants, the cascade classification model is more precise than common classifiers. The application of the cascade classification model as a flexibility description requires a decoder approach to make the flexibilities accessible for VPP scheduling algorithms. Since a first attempt for a decoder approach is not precise enough, the decoder development remains open for further research.

6.2 PERSPECTIVE

The main goal for future work is the development of an appropriate decoder function for the application of the cascade classification model as flexibility description in VPP scheduling algorithms. In this context a basic and a generalized decoder approach are required according to the basic and the generalized cascade classification model.

Basic Decoder A new basic decoder approach is required, which is not affected by the slight overestimation of the feasible class of the (basic) cascade classification model. Furthermore the error propagation from the basic cascade classification model to the basic decoder should be minimized. Probably also an adaptation of the basic cascade classification model is necessary.

Generalized Decoder Beside the problems with a basic decoder approach, a generalized decoder approach poses another problem. The ensemble of cascades of the generalized cascade classification model need to be integrated into a generalized decoder approach.

When these main problems with the decoder approaches are solved, the feasibility estimation of new decoder samples could be improved. Furthermore a comparative analysis to new decoder samples of related decoder approaches, like the decoder approach by Bremer 2015 and the one from AVPP scheduling, see Schiendorfer, Anders, et al. 2015 would be interesting.

A ENERGY DATA SETS OF SINGLE ENERGY UNITS

The cascade classification model is evaluated with energy data sets that are representative for VPP scheduling tasks. Such energy data sets are power production and power consumption time series of controllable producers and consumers. In this thesis the energy time series of a μCHP and a heat pump are used. Both data sets are high-dimensional with 96 dimensions and they employ severely imbalanced classes.

A.1 Micro Combined Heat and Power Plant

Combined heat and power plants (CHPs) produce thermal and electrical energy at the same time, see Thomas 2007. CHPs are available in different dimensions in order to provide thermal and electrical energy for buildings of different size.

A micro combined heat and power plant μCHP is a small CHP used in detached households. Most of the times these μCHPs are operated in a heat-controlled mode. In this case the flexibility of the μCHP can be used to optimize the electrical power production. In this thesis the μCHP power production time series are generated with a simulation model[1]. This simulation model is available in the energy division of the computer since department of the university of Oldenburg and was developed during the project INXS[2]. The μCHP simulation model includes beside a μCHP component a thermal buffer and the thermal demand of a detached house according to a normed warm warter demand, see Bremer 2015, p. 155. A μCHP can be operated in different operation modes, where the technical constraints of the μCHP, the constraints of

---

[1] Data sets are available for download on our department website http://www.uni-oldenburg.de/informatik/ui/forschung/themen/cascade/.

[2] This project, also called "VPP 2.0 - Konzeption des Datenaustauschs zwischen Anlagencontrollern und Fahrplanmanager", was conducted in collaboration with the EWE AG from 1.6.2008 to 31.12.2009

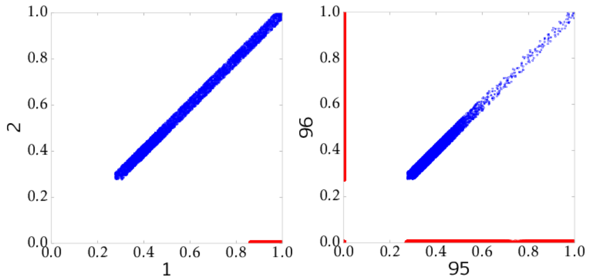
the thermal buffer and the constraints resulting from the thermal demand of the building are complied. The three main constraints are an output limitation, inertia of the μCHP modulation and the buffer temperature, see Bremer 2015, p.80. The output limitation describes the modulation of the μCHP between a minimal power output P_{min} and a maximal power output P_{max}. The inertia of the modulation limits the modulation of the power output between two time steps to $|p_i - p_{i+1}| leq \epsilon$, where ϵ depends on the μCHP. The third constraint is a limitation of the thermal power production. The thermal demand and buffering of thermal energy always have to lead to a thermal buffer temperature between an minimal and a maximal temperature.

The μCHP simulation model generates feasible power output time series and can test new time series concerning their feasibility, whether the power production of the test time series can be realized or not.

The default parametrization of the simulation model according to a Vaillant EcoPower 4.7[3] is slightly adapted: $n = 96$ time steps, an interval length $\delta_T = 900$ s and $numModes = 500$ discrete intervals that represent different CHP operation modes. Changes between the different operation modes are determined by a gradient, $gradient = 200$. This gradient corresponds to a maximal change of 3% of the maximal power production of the μCHP. The thermal demand of the household and the thermal buffer are used with the default parametrization. For classifier training and validation a set of $1,000,000$ feasible operation schedules is sampled and for testing a data set with $1,000,000$ infeasible operation schedules and $100,000$ feasible operation schedules. The simulated power output time series are scaled according to the maximal electrical power production of $P_{el} = 4700$ W to values between 0 and 1. Due to this scaling, the class of feasible and the class of infeasible power production time series form a hypercube. Due to different constraints like technical constraints, minimal and maximal water temperature of the thermal buffer or the compliance of the thermal demand of the household, the class of feasible power production time series does not form one cluster in feature space, but several clusters. These clusters are different concepts of the feasible class, see Fig. A.1. Furthermore the fragmented feasible class is much smaller than the infeasible class, because the infeasible class occupies a much larger volume in space than the feasible class, see Bremer et al. 2010. Both classes are clearly separable without a class overlap and in low-dimensional

---

[3] http://www.vaillant.de/ecopower/, last time visited 7.6.2017

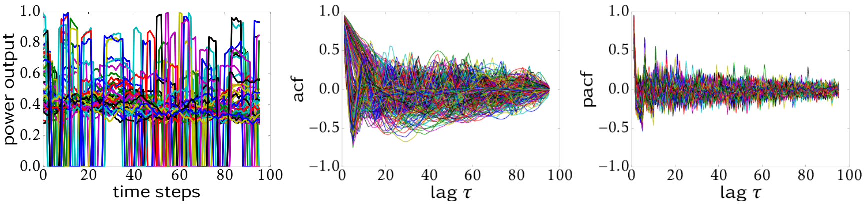


(a) 1st and 2nd dim (b) 95th and 96th dim

Figure A.1: The 1st and the 2nd time step (a) and the last two time steps (b) of the feasible μCHP power production time series show a middle concept (blue points) and an edge concept (red points).

space both concepts employ an easily learnable shapes. For some experiments the feasible class is reduced to the middle concept and the feasible edge concept is treated as infeasible.

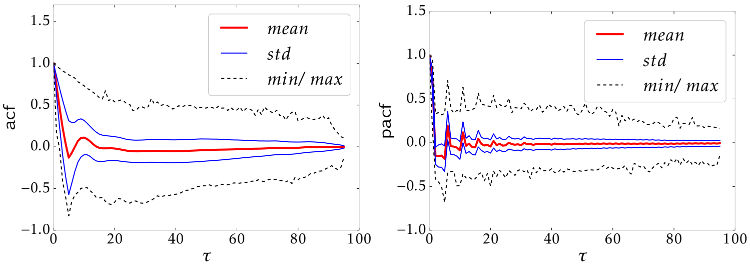
Feasible power production time series of the complete μCHP data set are plotted as time series in Fig. A.2(a) for 24 h with a resolution of 15 minutes. These time series



(a) CHP schedules (b) autocorrelation (c) partial autocorrelation

Figure A.2: Feasible μCHP power production time series: (a) plot of 50 normalized time series for 24 hours with 15 minutes resolution ($d = 96$ steps), (b) autocorrelation and (c) partial autocorrelation, each plotted for 500 time series.

(operation schedules) employ a decreasing autocorrelation and partial autocorrelation with increasing distance between the considered time steps, see Fig. A.2(b) and Fig. A.2(c). The autocorrelation and the partial autocorrelation a characteristics of one time series. The overall correlation behavior of the whole feasible class can be characterized by the mean and standard deviation of the (partial) autocorrelation, see Fig. A.3. These figures show the highest correlation between neighboring time steps and a strong correlation decrease for an increasing distance (τ) between the time steps. Even the reduced feasible class employs the same correlation behavior.



(a) autocorrelation (b) partial autocorrelation

Figure A.3: The correlation behavior is plotted for 10000 μCHP time series and increasing distances (τ) between the time steps: (a) autocorrelation and (b) partial autocorrelation.

Beside the correlation behavior and the shape of the feasible class, also the distribution of feasible and infeasible examples influences the achievable classification precision. Examples from both classes are distributed more or less homogeneously. But due to the severe imbalance, the volumes of both classes are represented with different sample densities. The feasible examples are located close to each other and the infeasible examples employ larger distances between each other. This means hardly any infeasible examples can be found close to the class boundary and infeasible test examples do not yield much information about the selectivity of the learned decision boundaries.

All in all the μCHP data set employs a simple data structure and will not yield information about the classifier selectivity.

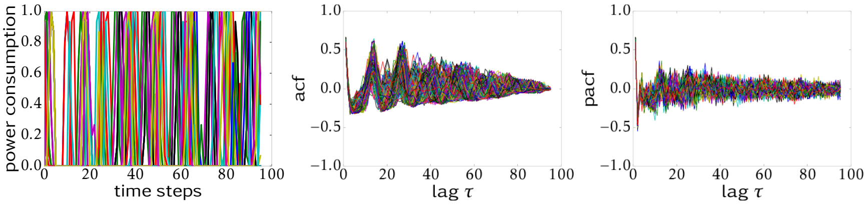
A.2 HEAT PUMP

Heat pumps are utilized as an example of controllable loads. Heat pumps use the thermal energy e.g., of the surrounding air or from an underground reservoir to meet the thermal demand of households, see e.g., Bremer 2015, p.135. Usually heat pumps are operated in combination with thermal buffers, that offer flexibility to the system in form of an adaptable electrical load of the heat pump. Heat pump power consumption time series are generated with a simulation model from the Smart Nord project, Sonnenschein and Hofmann 2015. The simulation model is available in the energy division of the computer science department of the university of Oldenburg. The simulation model imitates common heat pump types. Here the simulation model is used with the parametrer setting according to the heat pump model "Eltron WPF5" from Stiebel[4] and a default thermal buffer and a default heat demand of an apartment house with reference weather data, Wetterdienst 2010 for the 3rd of April 2010 including 2 simulated days of ahead-of-schedule work. This heat pump is a brine-water heat pump, that is not modulatable, but it can be switched on and off every minute. This "switching" allows 16 different mean power consumption values for an interval of 15 minutes. The heat pump power consumption time series are simulated for 24 h with a resolution of 15 minutes. For classification experiments $500,000$ feasible and $200,000$ infeasible power consumption time series are generated. The time series are scaled according to the maximal power consumption of 2400 W to values between 0 and 1.

Power consumption time series are plotted in Fig. A.4(a) for 24 h with a resolution of 15 minutes. These time series (operation schedules) employ a decreasing autocorrelation and partial autocorrelation for an increasing distance (τ) between the considered time steps, see Fig. A.4(b) and Fig. A.4(c). The overall correlation behavior of the feasible class is shown as mean autocorrelation and mean partial autocorrelation in Fig. A.5. The highest correlation can be found between neighboring time steps just like the μCHP operation schedules. But heat pump operation schedules employ also a high correlation between time steps with a distance $\tau = 2$ and shows a strong correlation decrease for greater distances τ.

---

[4] https://www.stiebel-eltron.de/de/home/produkte-loesungen/erneuerbare_energien/waermepumpe/sole-wasser-waermepumpen/wpf_5_7_10_13_16basic/wpf_5_basic.html, last time visited 7.6.2017

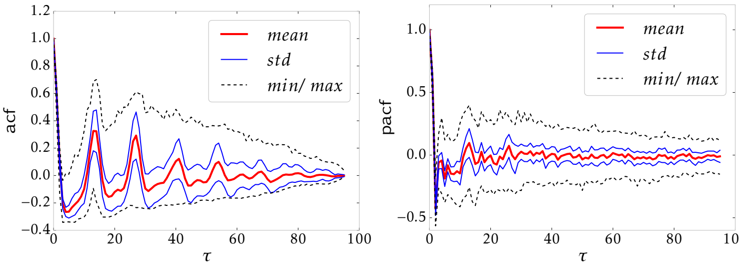


(a) heat pump schedules (b) autocorrelation (c) partial autocorrelation

Figure A.4: Heat Pump power consumption data (WPF5), (a) plot of 10000 normalized heat pump power consumption time series for 24 hours with 15 minutes resolution ($d = 96$ steps),(b) auto-correlation, (c) partial auto-correlation. The figures are plotted for 500 feasible schedules.

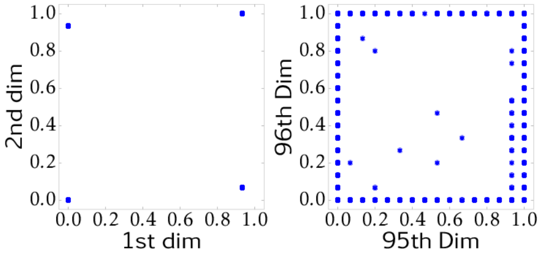
The structure of the feasible class of the heat pump operation schedules can be estimated in feature space. The feasible class consists of one concept that surrounds the infeasible class, see Fig. A.6. The shape of the low-dimensional feasible concept as in Fig. A.6 employs a hard to learn shape. Therefore the heat pump data set is transformed before classification. Some transformations like the fourier transformation, the autocorrelation function and linear principal component analysis simplify the data structure, see Fig. A.7. These transformations lead to data structures, that are easier to learn in low-dimensional space.

Independent from the data set transformations, both classes are severely imbalanced with a small feasible class and a large infeasible class. The imbalance and homogeneously distributed examples of both classes lead the hardly any infeasible examples near the class boundary. Therefore infeasible test examples do not yield much information about the selectivity of a classifier, just like the infeasible examples of the μCHP data set. But the heat pump data set yields information about the effect of the complex data structure on the classification precision.



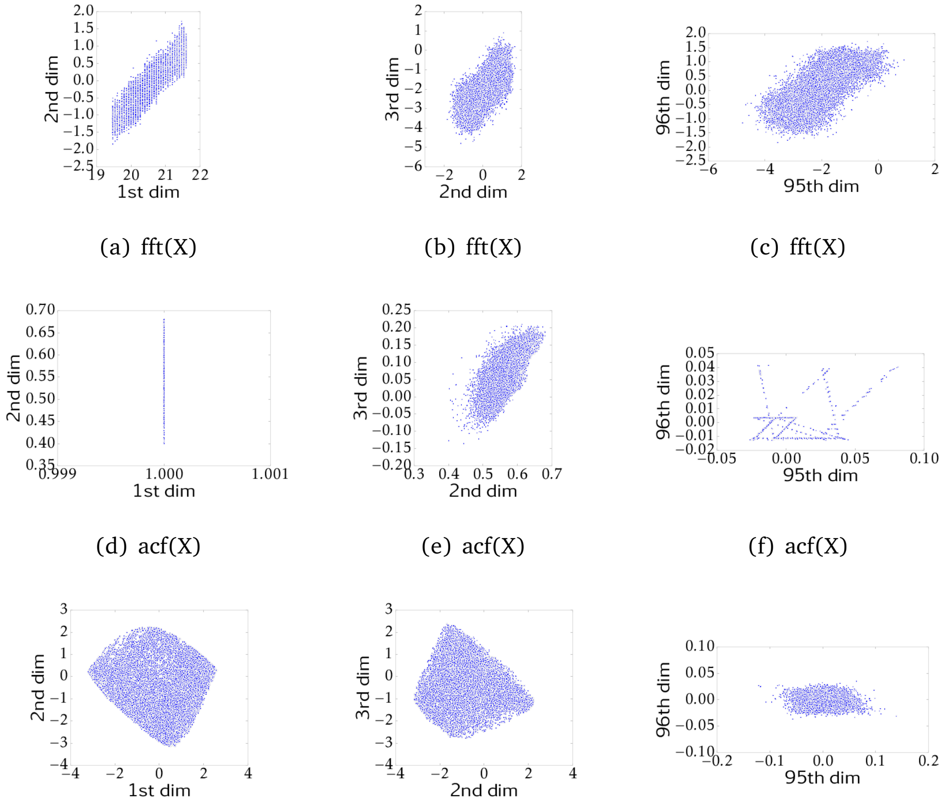
(a) autocorrelation (b) partial autocorrelation

Figure A.5: The correlation behavior is indicated as mean values, standard deviation and minimum and maximum values of a) the autocorrelation and (b) the partial auto-correlation for increasing distances between the time steps (τ). The calculations are done with 10000 feasible heat pump schedules.



(a) dim 1/2 (b) dim 95/96

Figure A.6: The 1st and the 2nd time step (a) and the last two time steps (b) of the feasible heat pump power consumption.



(a) fft(X) (b) fft(X) (c) fft(X)

(d) acf(X) (e) acf(X) (f) acf(X)

(g) pca(X) (h) pca(X) (i) pca(X)

Figure A.7: Subsets of the transformed feasible heat pump operation schedules plotted for the 1st and 2nd, the 2nd and the 3rd and the 95th and 96th time steps. The figures are plotted with 100000 feasible examples.

B ARTIFICIAL DATA SETS

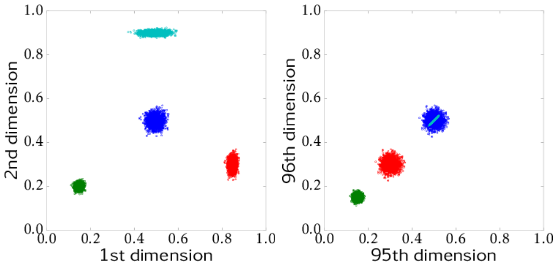
Beside energy data sets artificial data sets are used for the classifier evaluation. The artificial data sets are a *Hypersphere* data set and a *Hyperbanana* data set. These artificial data sets are high-dimensional data sets with severely imbalanced classes and employ special characteristics for classifier evaluation. The *Hypersphere* data set has a feasible class consisting of several concepts and the feasible *Hyperbanana* class has a hard to learn shape. *Hyperbanana* data is often used for classifier testing, because it is assumed to present a difficult classification task. Both data sets are presented below.

B.1 *Hypersphere* TASK

The *Hypersphere* data set consists of a feasible and an infeasible class. Both classes fill together a hypercube of side length 1. The feasible class consists of four hypersphere-shaped concepts. These four concepts are generated by stretching and translating hypersphere data simulated with Dd_tools, D. Tax 2013a. This algorithm yields representative and uniformly sampled hypersphere samples.

In feature space the shape of the feasible concepts is clearly visible, see Fig. B.1. For some experiments the feasible class is reduced to the blue middle concept. The infeasible class occupies the remaining hypercube volume. This class is sampled only near the class boundary around the feasible concepts. These infeasible examples are needed for testing the selectivity of the learned classifiers. If the feasible class is reduced to one concept, than only the corresponding infeasible examples are used.

The correlation behavior of the feasible class is analyzed in a time series-like representation separately for each concept of the feasible class, see Fig. B.5. Since the correlation behavior of does not differ among the concepts only the correlation behavior of the blue middle concept is plotted. The autocorrelation and partial au-

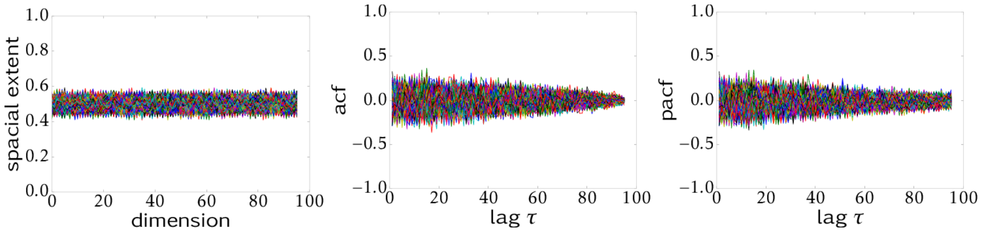


(a) 1st and 2nd dimension (b) 95th and 96th dimension

Figure B.1: 2000 feasible *Hypersphere* examples plotted in feature space for the 1st and 2nd (a) and the 95th and 96th dimensions (b). In (b) two concepts seem to be overlapping.

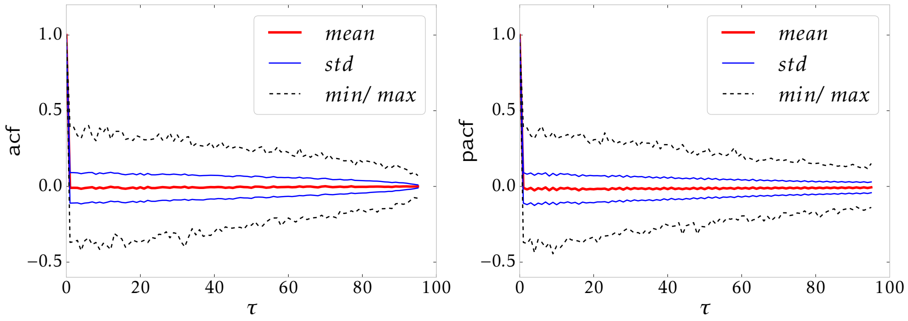
tocorrelation of single feasible examples are plotted in Fig. B.5(b) and Fig. B.5(c) for increasing distances τ between the dimensions (time steps). Mean values of the (partial) autocorrelation are shown in Fig. B.6. The feasible *Hypersphere* examples of the complete feasible class show no high correlation between near by dimensions, but a relatively low correlation for all considered correlation lengths τ. This correlation behavior deviates from the assumed correlation behavior in the cascade classification model and might lead to a low classification precision.

Even though the complete and the reduced *Hypersphere* data set are not expected to yield a high classification precision on the simple data structure, classification results will yield information about the selectivity of the classification boundaries.



(a) *Hypersphere* time series (b) autocorrelation (c) partial autocorrelation

Figure B.2: Feasible time series of the reduced *Hypersphere* data set (blue middle concept): (a) plot of 10000 96-dimensional examples,(b) autocorrelation and (c) partial autocorrelation, each plotted for 500 examples.



(a) autocorrelation (b) partial autocorrelation

Figure B.3: The correlation behavior plotted for 10000 feasible examples of the *Hypersphere* middle concept and increasing distances τ between the features: (a) autocorrelation and (b) partial autocorrelation.

B.2 *Hyperbanana* TASK

The *Hyperbanana* data set is an artificial complex data set where the small interesting class has a *Hyperbanana* shape. *Banana* and *Hyperbanana* data sets are often used to test new classifiers, because they are considered as difficult classification tasks. Therefore the results on the *Hyperbanana* data set is taken as a meaningful result. As far as there is no 96-dimensional *Hyperbanana* data set, we have generated a data set from the extended d-dimensional Rosenbrock function, (B.1),see Shang and Qiu 2006.

$$f(x) = \sum_{i=1}^{d-1} [100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2] \tag{B.1}$$

The small and interesting class, or here also called feasible class is sampled from the Rosenbrock valley with $f(x) < 100$ and the infeasible class with $f(x) >= 100$ is sampled only near the class boundary to test the sensitivity of the decision boundaries of the classifiers.

Sampling of the banana shaped valley is done by disturbing the minimum of the extended 96-dimensional Rosenbrock function with normally distributed values ($\mathcal{N}(0,1) \cdot \beta$ with $\beta \in \{40, 50, 60, 70\}$). The minima of the Rosenbrock function are presented in Shang and Qiu 2006 for different dimensionalities, but the minimum for 96 dimensions is missing. Therefore the minimum is approximated with regard to the minima of lower dimensional Rosenbrock data sets with -0.99 for the first dimension and with 0.99 for all other dimensions. The procedure of disturbing and selecting values from the Rosenbrock valley is repeated with the sampled values until enough data points are found. As far as it is difficult to sample the banana “arms” all at the same time, they are sampled separately by generating points that are $<$ or $>$ than a certain value and sampling is continued by repeating disturbance and selection with these values.

Values from all these repetitions were aggregated to one data set and shuffled. The samples generated by this procedure are not homogeneously distributed in the Rosenbrock valley and they do not represent all *Hyperbanana* “arms” equally.

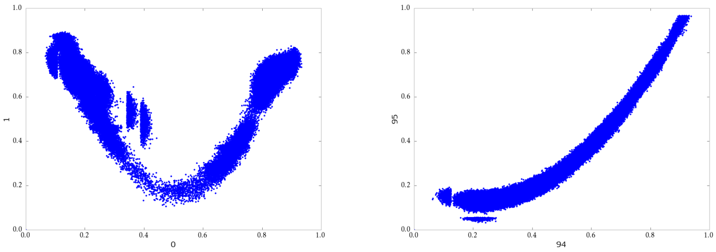
Overall two different *Hyperbanana* data sets are generated based on the Rosenbrock valley. The first data set (HB1) represents the different arms not that good as the second data set (HB2). Furthermore both data sets employ differences in the infeasible examples, see the generation of infeasible examples below.

After the generation of feasible examples, infeasible examples are generated. The infeasible class surrounds the feasible class and the infeasible class is much larger. The first data set contains 96-dimensional infeasible examples near the class boundary. These infeasible examples are sampled in the same way as the feasible ones but starting with the feasible *Hyperbanana* samples and with $100 \leq f(x) \leq 500$. The resulting infeasible examples are more or less homogeneously distributed and due to the proximity to the class boundary, such infeasible test examples yield information about the selectivity of the learned decision boundaries. Infeasible examples of the second data set are generated in the same way, but they are dived into three "rings" with an increasing distance to the class boundary. The first ring contains values with $100 < f(x) \leq 200$, the second ring contains values with $200 < f(x) \leq 300$ and the third ring contains values with $300 < f(x) \leq 500$. Data set HB1 employs only infeasible test examples from the first ring and data set HB2 employs infeasible examples from all three rings.

Finally all dimensions (features) x_i of the feasible class are scaled to values between 0 and 1 by $x_i = [x_i + (\min(x_i)+\text{offset})]/[\max(x_i)+\text{offset}-\min(x_i)+\text{offset}]$ with $\text{offset} = 0.2$. The infeasible class is scaled with the same values as the feasible class. Due to this scaling, both classes fill a hypercube volume.

Since both data sets belong to the same classification task, the corresponding properties are presented in the following only for the first data set (HB1). In feature space, see Fig. B.4 the banana shape of the feasible class is shown exemplarily for the first two features (dimensions) and the last two ones. Considering the feasible class of the *Hyperbanana* data set as time series, the correlation behavior of the single feasible examples can be computed and the results are shown in Fig. B.5. The overall correlation behavior of the feasible class is plotted in Fig. B.6 as mean and standard deviation of the autocorrelation and the partial autocorrelation. The highest correlation appears between neighboring time steps and the correlation decreases with increasing distances between the time steps.

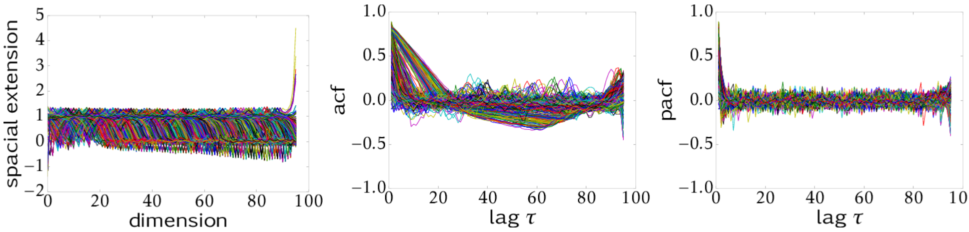
In summary the *Hyperbanana* data set is expected to yield meaningful results, because of the complex data structure and the infeasible examples near the class boundary.



(a) dim 1/2

(b) dim 95/96

Figure B.4: The 1st and the 2nd time step (a) and the last two time steps (b) of the feasible *Hyperbanana* examples.

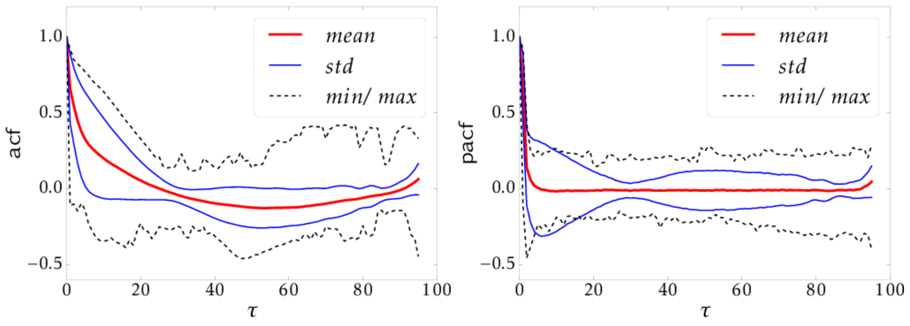


(a) *Hyperbanana* time series

(b) autocorrelation

(c) partial autocorrelation

Figure B.5: Feasible time series of the *Hyperbanana* data set A-HB1: (a) plot of 10000 96-dimensional examples,(b) autocorrelation and (c) partial autocorrelation, each plotted for 500 examples.



(a) autocorrelation

(b) partial autocorrelation

Figure B.6: The correlation behavior plotted for 500 feasible *Hyperbanana* examples and increasing distances τ between the features: (a) autocorrelation and (b) partial autocorrelation.

C ENERGY DATA SETS OF COALITIONS OF ENERGY UNITS

In the evaluation experiments coalitions of producers and coalitions of producers and consumers should be considered. The coalitions are homogeneous μCHP coalitions and mixed coalitions of μCHPs and heat pumps.

Global operation schedules of these coalitions cannot be generated directly, because they depend on simulation models of all coalition members. Feasible global schedules are generated based on feasible operation schedules of all coalition members and infeasible global operation schedules have to be estimated. Generation of feasible and infeasible global operation schedules as well as the respective data set properties are presented in Sect. C.1 and Sect. C.2. The data set of energy unit coalitions are sampled for a horizon of 24 hours with a resolution of 15 minutes.

C.1 GENERATION OF FEASIBLE GLOBAL OPERATION SCHEDULES

Generation of feasible global operation schedules and the respective data set properties are presented in this section. Feasible global operation schedules represent the overall power production, respectively consumption of all coalition members. Here only energy unit coalitions without dependences between the energy units (e.g., grid restrictions) are considered. Generation of feasible examples is done in two steps. First of all feasible global operation schedules of all coalition members are summed up to an overall power production. In the second step the representativity of the feasible global operation schedules of the feasible class is improved. These two steps are introduced in more detail below.

c.1.1 Step 1: Sample Generation

In the first step of the generation of feasible global operation schedules, alternatively feasible unscaled operation schedules of all coalition members are summed up. Below the generation of feasible global operations schedules and their representativity of the whole feasible class is presented.

Example Generation

Feasible global operation schedules sum up the unscaled power production time series, respectively consumption time series of all coalition members. This procedure is presented in Example C.1.1 for producer coalitions and in Example C.1.2 for mixed coalitions of producers and consumers.

Example C.1.1 (Generation of Feasible Global Operation Schedules of a Coalition of Energy Producers)
A feasible global operation schedule (S_g) of a coalition of m energy producers, e.g., μCHPs is composed of one unscaled feasible operation schedule S_i with $i \in 1, \ldots, m$ of each energy unit. A feasible global operation schedule is computed as

$$S_g = \sum_{i=1}^{m} S_i. \tag{C.1}$$

Resulting global operation schedules are scaled according to the maximal power production values of each energy unit Pel_{max_i} as $S_g / (\sum_{i=1}^{m} Pel_{max\_i})$.

Example C.1.2 (Generation of Feasible Global Operation Schedules of a Coalition of Energy Producers and Consumers)
A feasible global operation schedule (S_g) of a coalition of m energy producers, e.g., μCHPs and n energy consumers, e.g., heat pumps is composed of one unscaled feasible operation schedule of each energy producer Sp_i with $i \in 1, \ldots, m$ and one unscaled feasible operation schedule of each energy consumer Sc_j with $i \in 1, \ldots, n$.

A feasible global operation schedule is computed as

$$S_g = \sum_{i=1}^{m} Sp_i + \sum_{j=1}^{n} Sc_j. \tag{C.2}$$

THe resulting feasible global operation schedules can be scaled according to a maximal power production and a maximal power consumption. The m energy producers have the smallest power production $Pel_{min\_p\_i} = 0$ when they are switched off and they employ each a maximal power production of $Pel_{max\_p\_i}$. The n energy consumers have the smallest power consumption $Pel_{min\_c\_i} = 0$ when they are switched off and they employ a maximal power consumption each of $Pel_{max\_c\_i}$, with negative values.

The global operation schedules can employ values between a minimal value min and maximal value max. The $min \leq 0$ value is computed as

$$min = \sum_{i=1}^{m} Pel_{min\_p\_i} + \sum_{j=1}^{n} Pel_{max\_c\_j} = 0 + \sum_{j=1}^{n} Pel_{max\_c\_j}. \tag{C.3}$$

The $max \geq 0$ value is computed as

$$max = \sum_{i=1}^{m} Pel_{max\_p\_i} + \sum_{j=1}^{n} Pel_{min\_c\_j} = \sum_{i=1}^{m} Pel_{max\_p\_i} + 0. \tag{C.4}$$

Scaling is done as $(S_g - min/(max - min))$.

Next the representativity of the resulting feasible global operation schedules of the feasible class is studied in the next subsection.

Representativity of the Generated Feasible Examples

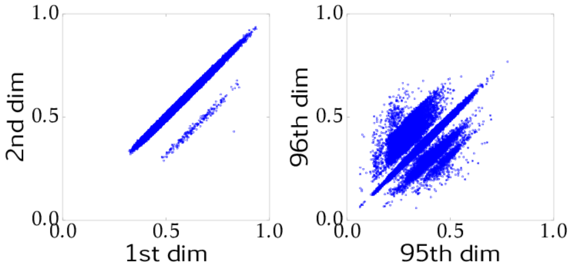
The feasible operation schedules S of all coalition members are multidimensional random variables. Summing up these random variables to global operation schedules S_g followed by a normalization yields the arithmetic mean. According to the law of large numbers, the arithmetic mean of multidimensional random variables converges towards the expectation values of all components of the random variable, see e.g., Lavine 2009; Alpaydin 2010. This convergence of the global operation schedules towards the expectation values of all components (time steps) for increasing numbers of coalition members is shown in Fig. C.1.

As far as the coalition members are identical μCHPs, they could be all operated in the same way. The corresponding feasible region in feature space for identically operated μCHPs should look like the feasible region of one μCHP in feature space, see Fig. A.1.

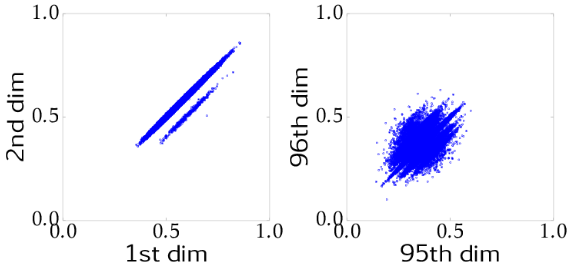
Thus the global operation schedules S_g generated in the first step and shown in Fig. C.1 do not represent the whole feasible class, the representativity needs to be increased especially for coalitions with several members.

In coalitions consisting of only a few identical members, representative feasible global operation schedules could be generated as before but this time based on the sum of specific combinations of the operation schedules of all coalition members. In a coalition of three identical μCHPs all μCHPs could be operated differently, identically or two μCHPs are operated identically and the third one in another way. The resulting global operation schedules, respectively the representation of the feasible class in feature space is shown in Fig. C.2 for this example of 3 μCHPs and coalitions of 5 and 10 μCHPs.

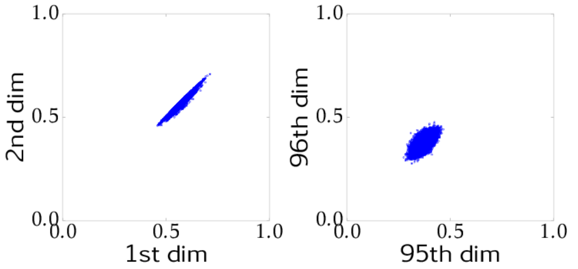
The resulting global operation schedules shown in Fig. C.2 and their representation in feature space are used in the following as references for comparison. Since the employed specific summation of operation schedules is not applicable to coalitions of different energy units as well as large coalitions, a method is needed that generates more representative feasible global operation schedules. Such a method is presented in the next section.



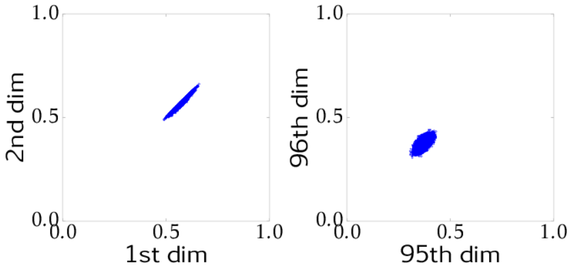
(a) 5 μCHPs step 1, 2 (b) 5 μCHPs step 95, 96



(c) 10 μCHPs step 1, 2 (d) 10 μCHPs step 95, 96

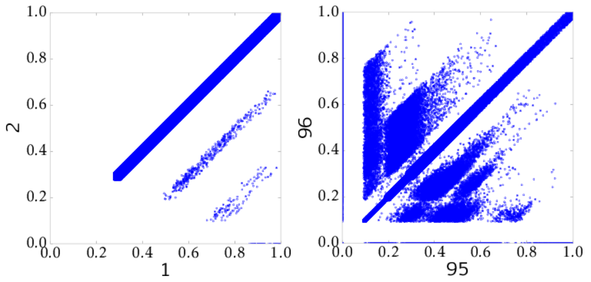


(e) 50 μCHPs step 1, 2 (f) 50 μCHPs step 95, 96

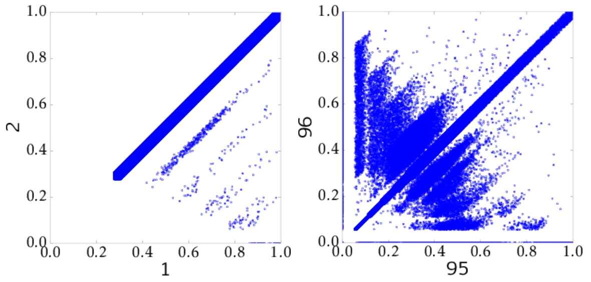


(g) 100 μCHPs step 1, 2 (h) 100 μCHPs step 95, 96

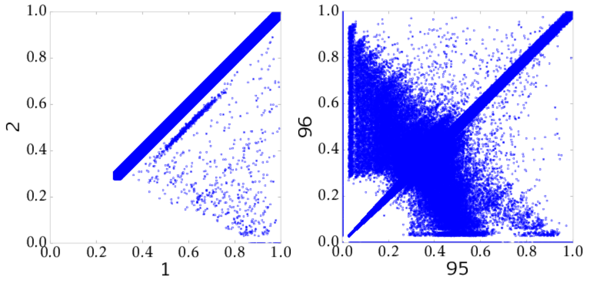
Figure C.1: The first and the last two time steps of 96-dimensional time series are plotted against each other for increasing numbers of identical μCHPs in the coalitions. Each subfigure is plotted with $50,000$ time series segments.



(a) 3 μCHPs step 1, 2 (b) 3 μCHPs step 95, 96



(c) 5 μCHPs step 1, 2 (d) 5 μCHPs step 95, 96



(e) 10 μCHPs step 1, 2 (f) 10 μCHPs step 95, 96

Figure C.2: The first and the last two time steps of 96-dimensional time series are plotted against each other for increasing numbers of identical μCHPs in the coalitions. The time series segments in each subfigure represent global operation schedules resulting from specific combinations of the operation schedules of all coalition members.

c.1.2 Step 2: Example Shifting

The second step of generating representative feasible global operation schedules aims at increasing the representativity of the generated feasible global operation schedules. Therefore the global operation schedules generated in step 1 are shifted into the different regions of the volume of the feasible class. Below the shifting method and the resulting representativity are presented after another.

Shifting of new Feasible Examples

Shifting of the feasible global operation schedules in feature space is done according to Algorithm 4. The M global operation schedules S_g are shifted away from the expectation values one after another. Therefore the k nearest neighbors of the respective global operation schedule $S_{g,i}$ are identified as $S_{g,l}$ with $l = \{1, \dots, k\}$. Next the cluster center C of the k nearest neighbors $S_{g,l}$ is computed. Then $S_{g,i}$ is shifted into the opposite direction of the cluster center with the shift vector δ and a factor

Algorithm 4 Generation of representative global operation schedules

```
Require: a set of $M$ normalized global operation schedules $S_{g,i}$ $(i = 1, \dots, M)$ from
    step 1, see Sect. C.1.1 and all underlying normalized operation schedules $S_j$ of
    all $\ell$ coalition members
    // shift of the $M$ global operation schedule $S_{g,i}$
for $i = 1$ to $M$ do
    identification of the $k$ nearest neighbors $S_{g,l}$ $(l = 1, \dots, k)$ of $S_{g,i}$
    $C = \frac{1}{k} \sum_{l=1}^{k} S_{g,l}$ ($C$ = cluster center of the $k$ nearest neighbors $S_{g,l}$)
    $\delta = S_{g,i} - C$ (shift vector)
    $S'_{g,i} = S_{g,i} + \delta \cdot x$ (shifted global schedule with factor $x$)
    // approximation of $S'_{g,i}$ by shifting the schedules $S_j$ of each coalition member
    // towards $S'_{g,i}$
    for $j = 1$ to $\ell$ do
        $\hat{S}_j = S_j$ with the smallest euclidean distance to $S'_{g,i}$
    end for
    $\hat{S}_{g,i} = (\sum_{j=1}^{\ell} \hat{S}_j) / \ell$ (normalization of the overall power production )
    $S_{g,i} = \hat{S}_{g,i}$
end for
```

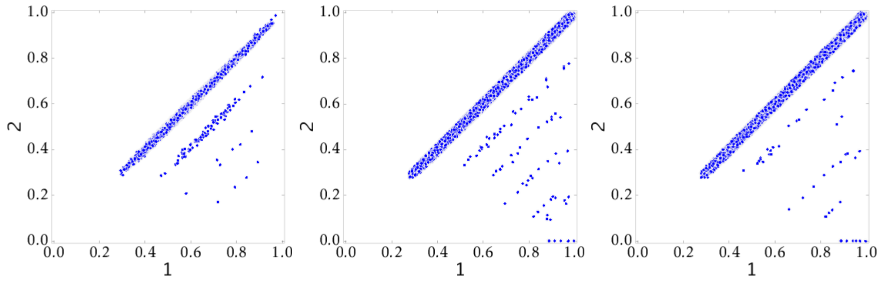
x and the resulting schedule is called $S'_{g,i}$. Since $S'_{g,i}$ does not have to be feasible, it is approximated by a nearby global feasible operation schedule. This approximation is done by drawing the feasible operation schedules S_j of all ℓ coalition members towards $S'_{g,i}$. The schedules S_j with the shortest euclidean distance to $S'_{g,i}$ are called $\hat{S}_j$. Then the ℓ schedules $\hat{S}_j$ of all coalition members are summed up and the result is normalized to values between 0 and 1. The resulting shifted operation schedules $\hat{S}_j$ form the new feasible global operation schedule $\hat{S}_{g,i}$. This shifting procedure is repeated for all remaining global operation schedules $S_{g,i}$ with $i = 2, \ldots, M$.

The shifting procedure employs two parameters, the number of nearest neighbors k and the factor x to control the shift length. The parameter k can be chosen independent from the number of coalition members and the dimensionality of the operation schedules. In pre-tests $k = 5$ performed well and is used as default value. In contrast to k the shift length parameter x depends on the number of coalition members and the number of dimensions of the operation schedules. The higher the number of coalition members and the higher the dimensionality of the operation schedules, the higher should be the value of x. The better the choice of x, the better the distribution of the shifted global operation schedules in feature space, for more details see the following subsection.

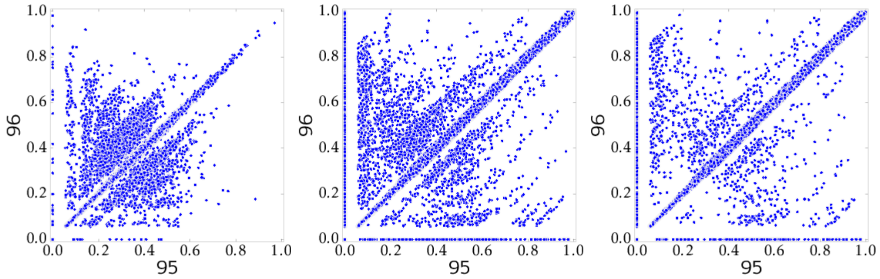
Beside the parameter choice also the implementation of Algorithm 4 benefits from fine tuning, because amongst others the nearest neighbor search can be very time consuming.

Representativity of the Shifted Examples

The shifted global operation schedules represent the whole region of the feasible class more or less, depending on the parametrization of the shifting algorithm, see Fig. C.3 exemplarily showing the effect for a coalition of 5 identical μCHPs. Optimal values of x lead in 2-dimensional feature space to the largest region covered by feasible examples which are distributed as homogeneously as possible. To small values of x reveal examples only in some regions of the volume of the feasible class and to large values reveal representative examples but with a very inhomogeneous distribution. In this subsection only shifting results for optimized values of x are considered, see Tab. C.2. The resulting distribution of feasible examples is shown in Fig. C.4. In comparison to the global feasible operation schedules without shifting, see Fig. C.1,



(a) x too small $x = 5$ (b) x optimal $x = 30$ (c) x too large $x = 100$



(d) x too small $x = 5$ (e) x optimal $x = 30$ (f) x too large $x = 100$

Figure C.3: The first and the last two time steps of shifted global operation schedules are plotted against each other for increasing shift length values x. The time series segments in each subfigure represent shifted global operation schedules for a coalition of 5 identical μCHPs.

the shifted ones represent a much larger region of the feasible class in 2-dimensional feature space. Additionally the shifted global operation schedules of coalitions of 5 and 10 identical μCHPs can be compared to the global operation schedules based on specific operation schedule combinations of all coalition members, see Fig. C.2. This comparison yields each time a similar region of the feasible class covered with examples. For this reason one can assume that the shifted global operation schedules represent the whole region of the feasible class. Furthermore the shifted global operations schedules are more homogeneously distributed than the ones based on specific combinations.

number of μCHPs	x value
5	30
10	50
50	90
100	140

Table C.1: Optimized x values for coalitions of identical μCHPs.

C.1.3 Properties of the Generated Global Feasible Operation Schedules

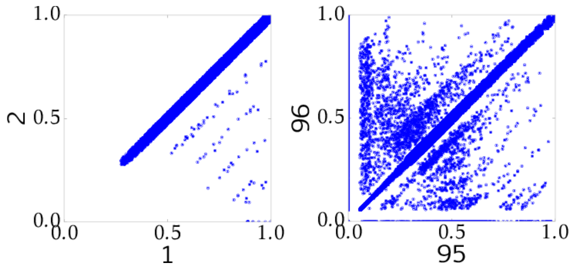
Even though the generation of feasible global operation schedules was presented in the previous subsections at the example of homogeneous μCHP coalitions, in this section also mixed coalitions of μCHPs and heat pumps are considered.

Generation of feasible global operation schedules with the two steps: sample generation and shifting leads to representative examples of the whole feasible class with a more or less homogeneous distribution. All in all the generation of global feasible operation schedules in two steps leads to representative examples of the whole feasible class with a more or less homogeneous distribution, see Fig. C.4 for homogeneous μCHP coalitions and see Fig. C.5 for mixed μCHP and heat pump coalitions. The data structure of the feasible class of all coalitions is complex with several concepts and and hard to learn shapes. Furthermore homogeneous and mixed coalitions employ a similar correlation behavior, see the autocorrelation and partial autocorrelation in Fig. C.6 for homogeneous μCHP coalitions and Fig. C.7 for mixed coalitions. The highest correlation in homogeneous μCHP coalitions can be found between neighboring time steps and for time steps with longer distances τ the correlation decreases rapidly. Mixed coalitions show a similar correlation behavior and they employ additionally a high correlation between time steps with a lag $\tau = 2$.

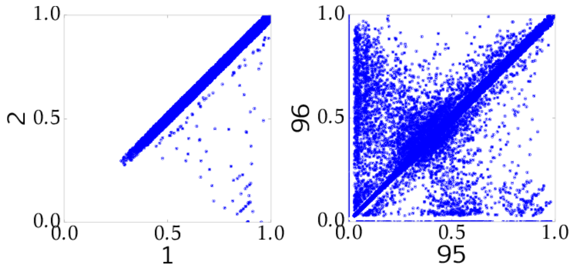
Due to the similarities in the data structure and the correlation behavior of homogeneous and mixed coalitions, similar results can be expected for all coalitions. Therefore only homogeneous μCHP coalitions with 5, 10, 50 and 100 members are considered in the following and all evaluation experiments. But even the data structure and the correlation behavior of these considered homogeneous μCHP coalitions is very similar.

To achieve good classification results on these data sets, the data structure can be simplified with time series transformations. In pre-tests the time series transformations with the autocorrelation function *acf*, the fourier transformation *fft* and the principal component analysis *pca* were identified as suitable. These transformations lead to at best only one feasible concept with an easy to learn shape, see Fig. C.8.

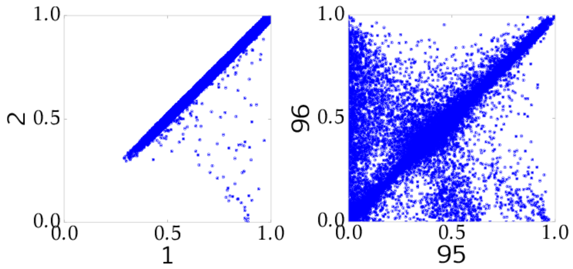
Beside feasible global operation schedules which are necessary for building classifiers and testing them also infeasible global operation schedules are required to test classification models. In the next section the generation of global infeasible operation schedules is presented.



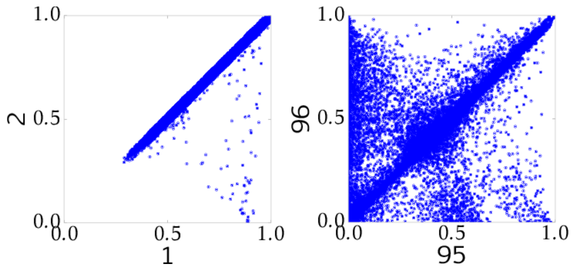
(a) 5 μCHPs step 1, 2 (b) 5 μCHPs step 95, 96



(c) 10 μCHPs step 1, 2 (d) 10 μCHPs step 95, 96

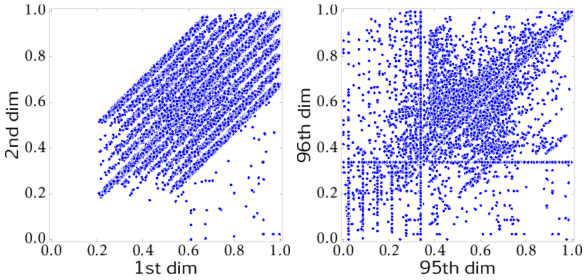


(e) 50 μCHPs step 1, 2 (f) 50 μCHPs step 95, 96

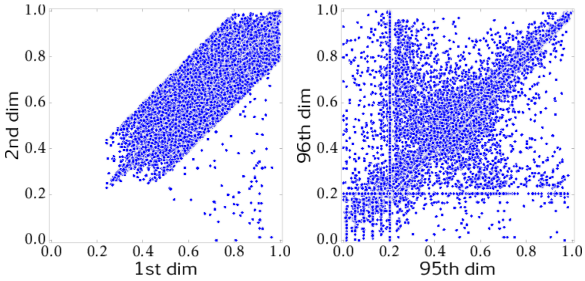


(g) 100 μCHPs step 1, 2 (h) 100 μCHPs step 95, 96

Figure C.4: Feasible global operation schedules of the same coalitions as in Fig. C.1 where the global operation schedules were shifted in feature space. Each subfigure is plotted with 30,000 time series segments of the first and the last two time steps of 96-dimensional time series.

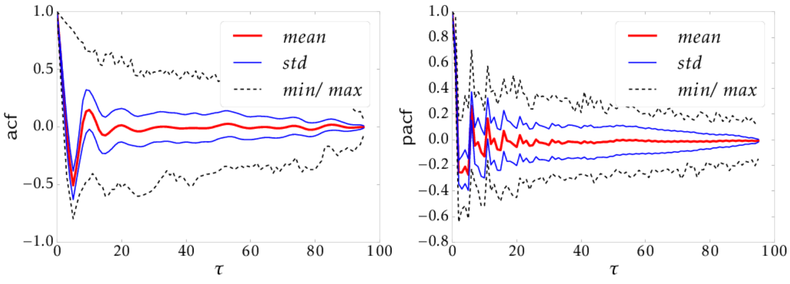


(a) 5 μCHPs, 5 heat pumps step 1, 2 (b) 5 μCHPs, 5 heat pumps step 95, 96

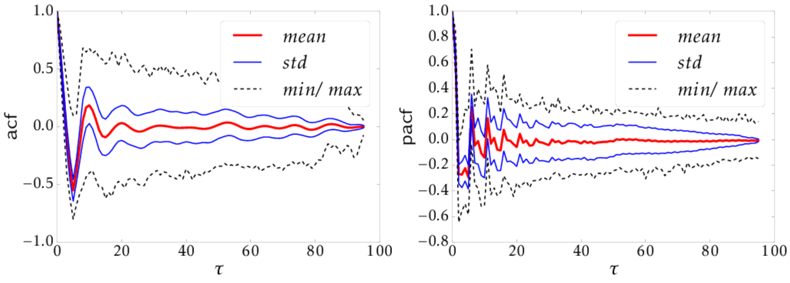


(c) 10 μCHPs, 5 heat pumps step 1, 2 (d) 10 μCHPs, 5 heat pumps step 95, 96

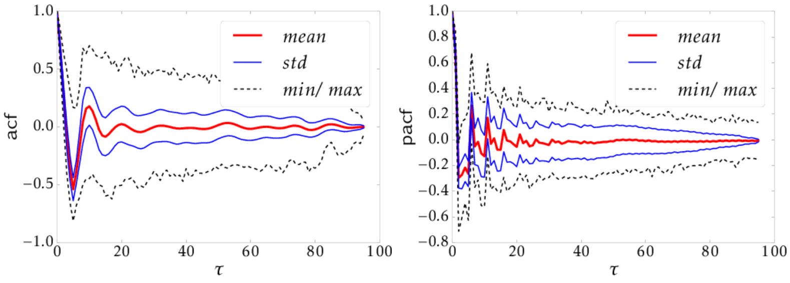
Figure C.5: The first and the last two time steps of 96-dimensional time series are plotted against each other for coalitions of different numbers of μCHPs and heat pumps. Each subfigure is plotted with $30,000$ time series segments.



(a) 5 μCHPs (b) 5 μCHPs

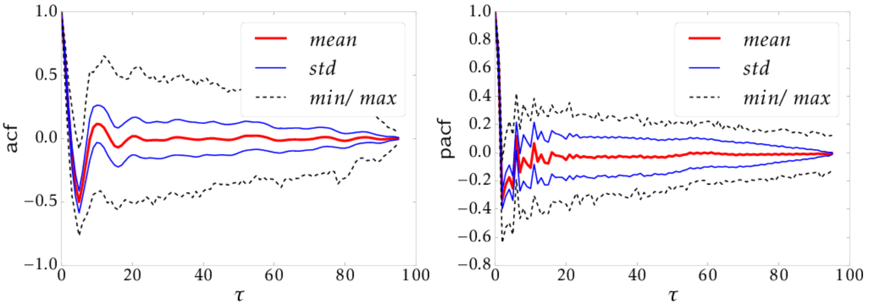


(c) 10 μCHPs (d) 10 μCHPs

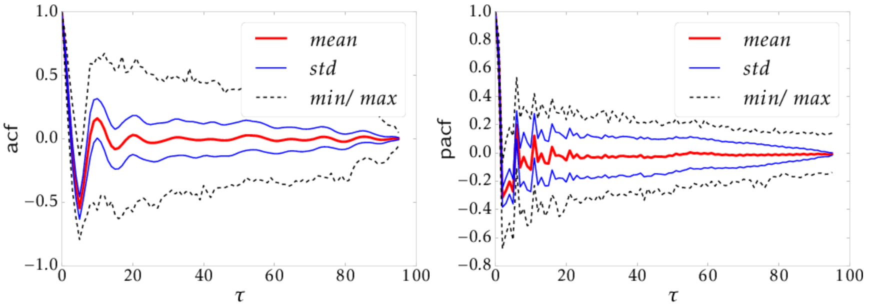


(e) 50 μCHPs (f) 50 μCHPs

Figure C.6: Autocorrelation and partial autocorrelation of homogeneous μCHP coalitions with different numbers of energy units. The figures are calculated based on the 30,000 feasible global operation schedules shown in Fig. C.4.

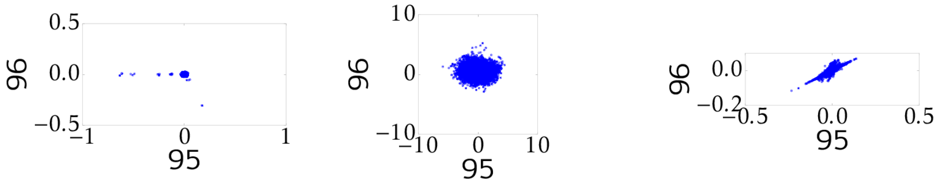


(a) 5 μCHPs 5 heat pumps (b) 5 μCHPs 5 heat pumps

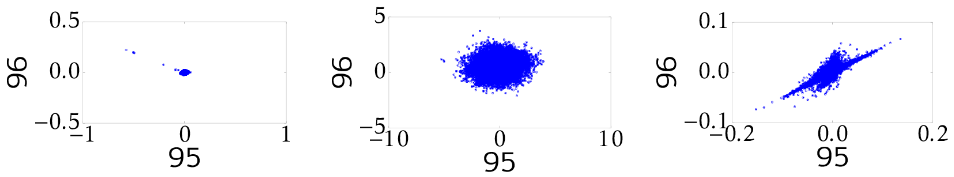


(c) 10 μCHPs 5 heat pumps (d) 10 μCHPs 5 heat pumps

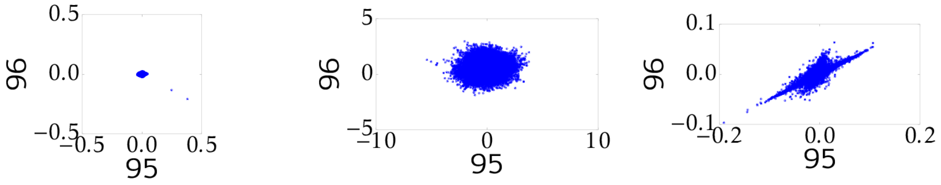
Figure C.7: Autocorrelation and partial autocorrelation of mixed μCHP and heat pump coalitions. The figures are calculated based on the 30,000 feasible global operation schedules shown in Fig. C.5.



(a) 5 μCHPs pca(X) (b) 5 μCHPs fft(X) (c) 5 μCHPs acf(X)



(d) 10 μCHPs pca(X) (e) 10 μCHPs fft(X) (f) 10 μCHPs acf(X)



(g) 50 μCHPs pca(X) (h) 50 μCHPs fft(X) (i) 50 μCHPs acf(X)

Figure C.8: Subsets of differently transformed feasible global operation schedules of the μCHP coalitions, see Fig. C.4. The figures are plotted with 30,000 points.

C.2 Generation of Infeasible Global Operation Schedules

After the generation of feasible examples in the previous section, now the generation of infeasible examples is presented. Since infeasible global operation schedules cannot be generated directly, they have to be estimated. Even though infeasible operation schedules of all coalition members are available, they cannot be combined to global operation schedules in the same way as feasible ones. Combinations of infeasible operation schedules might be also realizable by combinations of feasible operation schedules.

Therefore a method for estimating infeasible global operation schedules is presented below in two steps. First of all potentially infeasible global operation schedules are generated and in a second step their feasibility is estimated.

C.2.1 Sample Generation

Infeasible operation schedules should represent the volume of the infeasible class as good as possible with homogeneously distributed examples. Additionally infeasible examples near the class boundary would be helpful to test the selectivity of the built classifiers. In the following the generation of these two different potentially infeasible data sets is presented.

Due to the severe imbalance of the volume of the feasible and infeasible class of single energy units, a severe imbalance can be expected between the feasible and infeasible class of global operation schedules. Scaled global operation schedules employ values between 0 and 1 at each time step. This means all time series lie inside a hypercube. Sampling of new time series could be done by drawing random samples from this hypercube. Due to the severe imbalance of the classes most hypercube samples have to be infeasible depending on the dimensionality of the sampled time series. Since the infeasible examples should represent the volume of the infeasible class as good as possible the samples are drawn from a uniform distribution on the interval $[0, 1]$ for each time step. The resulting time series are stationary and similar to white noise, where all time steps are uncorrelated random variables with a constant mean and a constant variance, see Müller 1990.

Even though these examples are representative they are mainly not located near the class boundaries, due to the imbalanced class volumes. Therefore a second set of potentially infeasible global operation schedules with examples closer to the class boundary would be helpful. Such a data set should employ similar properties as feasible examples. In this case such properties could be the correlation behavior and the overall power production of each global operation schedule. Additionally the range of maximal changes between two time steps could be interesting, but in the case of the considered μCHP coalitions with several numbers of members, the power production could employ changes between 0% to 100% of the power production within two time steps.

Potentially infeasible time series with the above mentioned properties can be generated with the help of an autoregressive process (AR). An autoregressive process of order p is defined as

$$AR(p) = X_t + a_1 X_{t-1} + a_2 X_{t-2} + \cdots + a_p X_{t-p} \tag{C.5}$$

with the constants a_i. The characteristics of an AR(p) are a slowly decreasing autocorrelation function towards zero and a rapidly deceasing partial autocorrelation function towards zero, see Brockwell and R. A. Davis 1987; Müller 1990; Fuller 1996; Tsay 2005. Furthermore autoregressive processes A high correlation between neighboring time steps as in feasible global operation schedules is achieved with an autoregressive process of order $p = 1$ (AR(1)):

$$AR(1) = X_t + a_1 X_{t-1}. \tag{C.6}$$

But an AR(1) is only an approximation. The AR(1) neglects the weak correlation between time steps with a little longer distances than $\tau = 1$, that is present in the feasible time series, see e.g. Tab. C.2. The AR(1) process is simulated with the help of the python package *nitimes*, see Ghosh et al. 2009 and the respective parameters a_i are estimated based on Yule-Walker equations.

Since the autoregressive process is not limited to values between 0 and 1, the resulting time series need to be scaled to this value range. After scaling the time series with the same overall power production as feasible ones need to be selected. They form set 1 and the remaining infeasible examples form set 2.

After the generation of the two data sets of potentially infeasible global operation schedules the feasibility of theses samples has to be estimated.

C.2.2 Feasibility Estimation

Since there are no general feasibility criteria to decide whether a new operation schedule is feasible or not, they have to be derived. There are mainly two possibilities to derive such criteria. On the one hand various properties of the feasible and infeasible examples can be compared. Differences in the properties of feasible and potentially infeasible data sets are taken as feasibility criteria. On the other hand feasible and potentially infeasible global operation schedules can be approximated with their nearest feasible neighbor with the scheduling heuristic COHDA, Hinrichs, Lehnhoff, et al. 2014. Approximations of feasible global operation schedules should employ shorter distances to the given feasible global operation schedules than approximations of infeasible ones to the given ones. These two feasibility estimation methods are presented below in more detail at the example of homogeneous μCHP coalitions and the potentially infeasible data sets, presented above in Sect. C.2.1.

First Possibility: Feasibility Estimation Based on Data Properties

Especially feasible operation schedules employ characteristic properties and infeasible operation schedules differ in one or more properties from the feasible ones. Due to the much smaller volume of the feasible class than the volume of the infeasible class, the infeasible class should employ diverse characteristics of different properties. Some data set properties are directly linked to the feasibility, while others give only indirect hints. Therefore the properties with a direct link to the feasibility should be considered primarily. If the differences between a feasible and a potentially infeasible data set are distinct, the potentially infeasible data set can be treated as an infeasible data set.

Since the choice of helpful properties depends on the application, the properties comparison of a feasible and potentially infeasible data sets is shown at the example of a coalition of homogeneous μCHPs. μCHP coalition data sets employ more and less helpful properties concerning the feasibility estimation. A helpful properties, which is directly liked to the feasibility is e.g., the overall power production. The overall power production of feasible time series is limited by the thermal demand and temperature limits of the thermal buffer. Less helpful properties, which are not directly linked to the feasibility are the correlation behavior of the time series and the step length between two neighboring time steps. Feasible time series employ a characteristic correlation behavior with a high correlation between neighboring time steps and a decreasing correlation for time steps with longer distances. Furthermore they employ a characteristic distribution of power production changes between two neighboring time steps. Strong deviations from the characteristic behavior of feasible examples indicates infeasibility.

All these properties are shown in Tab. C.2 exemplarily for a coalition of 5 μCHPs. The two potentially infeasible data sets (hypercube samples and AR(1) samples) can be treated as infeasible, because there are more or less strong deviations in all considered data set properties with only a few similarities between the feasible and the potentially infeasible data sets. Even the part of the AR(1) samples with a similar overall power production as feasible global time series can be treated as infeasible, due to the remaining deviations.

Second Possibility: Feasibility Estimation Based on the Approximation of Global Operation Schedules with COHDA

A second possibility to estimate the feasibility of potentially infeasible time series consists in the approximation of a new global operation schedule with CODA. Then the distance between given and the approximated schedule can be compared for feasible schedules and potentially infeasible ones. COHDA was developed to approximate new overall schedules for a pool of energy units on the basis of feasible power production time series for each pool member. COHDA searches for a given global operation schedule the nearest feasible global operation schedule. The similarity, respectively distance δ_d between given global operation schedules and with COHDA realized global operation schedules is measured with the euclidean distance. The distance δ_d should be equal to zero or at least very small for the approximation of feasible global operation schedules, while δ_d should be larger for the approximation of infeasible global operation schedules.

For estimating the feasibility of the hypercube samples COHDA is employed in combination with a search space model for each energy unit in the pool, see Bremer and Sonnenschein 2013c; Bremer and Sonnenschein 2013a; Hinrichs, Bremer, and Sonnenschein 2013. For each hypercube sample three runs are performed with CO-

feasible examples	potentially infeasible set 1 hypercube samples	potentially infeasible set 2 AR(1) samples

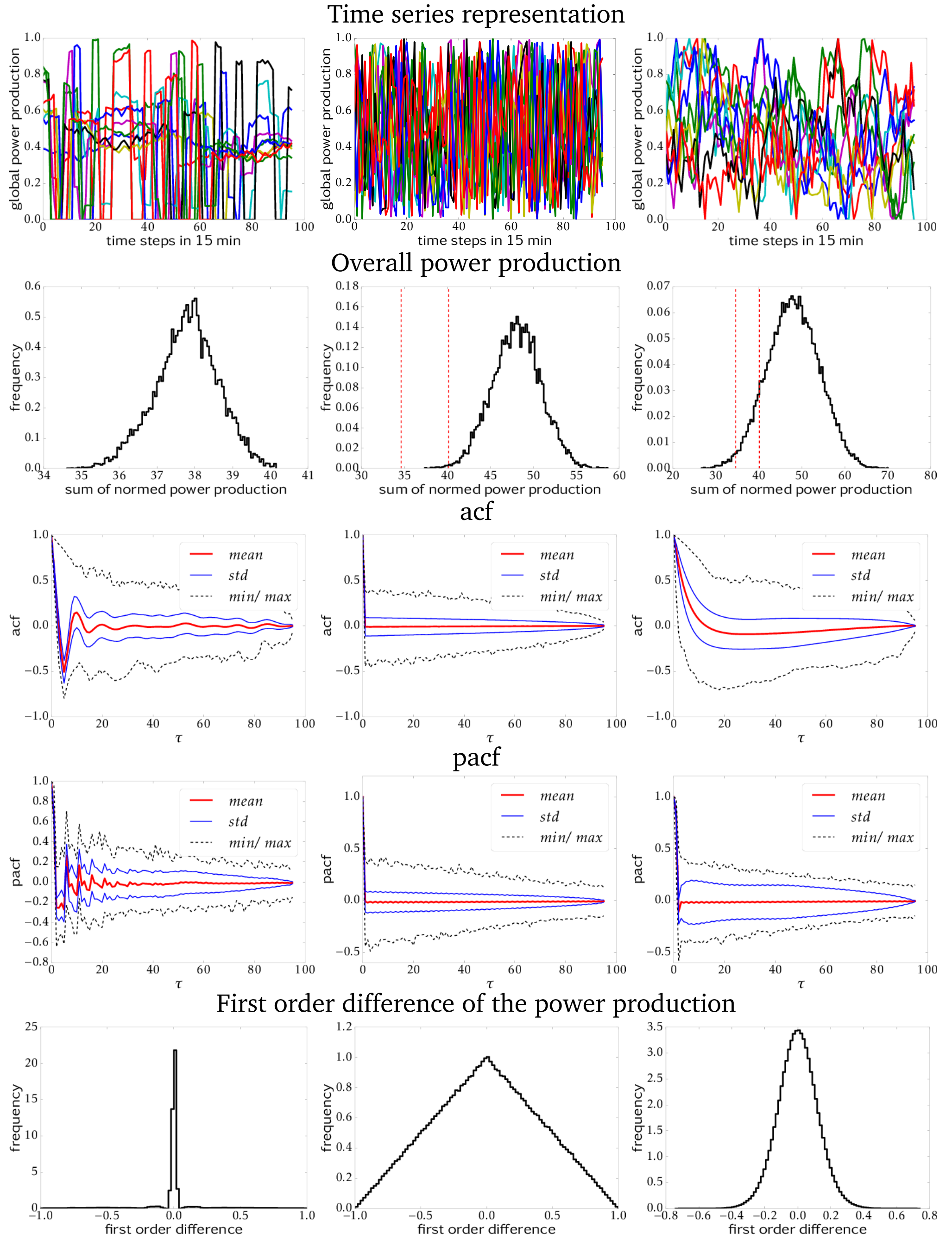
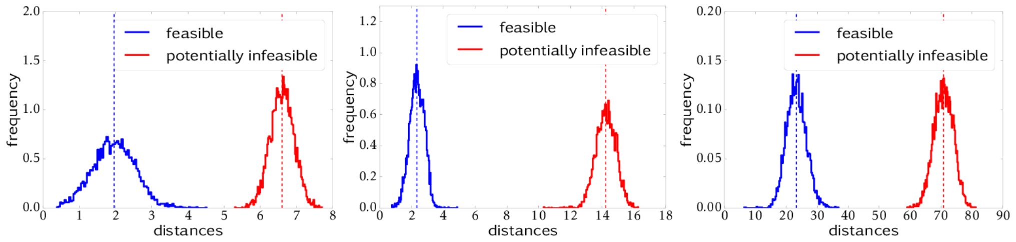


Table C.2: Properties of feasible and infeasible global operation schedules at the example of a coalition of 5 identical μCHPs.

HDA and the mean euclidean distance δ_d between the realized time series and the given global operation schedules are stored. This feasibility estimation is repeated with feasible global operation schedules to receive reference distances δ_d.

The feasibility estimation of 96-dimensional hypercube samples and feasible reference examples was done for μCHP coalitions with different numbers of members. Furthermore this estimation was done once with the global operation schedules resulting from step 1 and once with the global operation schedules resulting from step 2. In the case of feasible examples from step 1, the resulting distances δ_d for the feasible reference examples is mainly much smaller than the δ_d for the hypercube samples in all experiments. But nevertheless both distance δ_d distributions overlap a little. The results for coalitions of 5, 10 and 50 members are shown in Fig. C.9.



(a) coalition of 5 μCHPs (b) coalition of 10 μCHPs (c) coalition of 50 μCHPs

Figure C.9: Distributions of the distances δ_d between not representative feasible overall time series (summed up schedules without shifting) of different numbers of μCHPs and the approximated ones with COHDA (blue right distribution) and potentially infeasible overall time series and the ones approximated with COHDA (red left distribution). The dashed lines indicate the mean values of both distance distributions.

As far as COHDA is a heuristic, the feasibility of new overall time series can be only assumed. Since the overlap between the distance δ_d distribution for feasible and potentially infeasible global operation schedules (hypercube samples) is only very small, see Fig. C.9, all hypercube samples are taken as infeasible examples for the μCHP coalitions.

In the case of feasible operation schedules resulting from step 2, the two distance δ_d distributions overlap and a feasibility judgment is hardly possible, see Fig. C.10.

This is probably due to the parametrization of COHDA and the employed search space model. But an explanation of this observation requires more research. To judge the feasibility of global operation schedules further feasibility estimations are required.

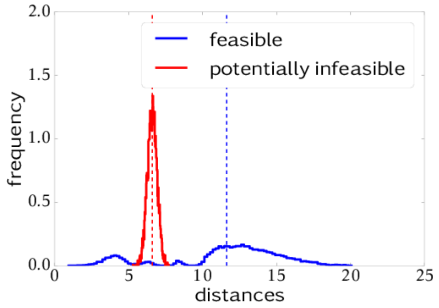


Figure C.10: Distributions of the distances δ_d between feasible overall time series of 5 μCHPs and the approximated ones with COHDA (blue left distribution) and potentially infeasible overall time series and the ones approximated with COHDA (red right distribution). The dashed lines indicate the mean values of both distance distributions.

sum of normed overall power production is similar for coalitions with different numbers of μCHPs.

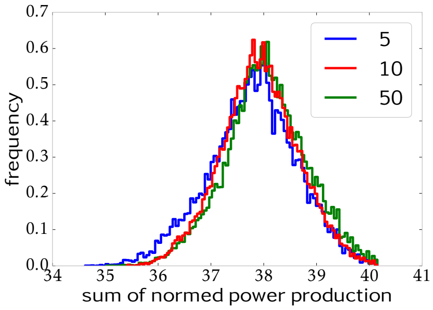


Figure C.11: Distributions of the normed power production of feasible and potentially infeasible global operation schedules. The histogram is computed with 30,000 feasible global operation schedules of each coalition.

BIBLIOGRAPHY

Aggarwal, C. C. . (2014). “Data Classification”. In: ed. by C. C. . Aggarwal. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Chapman & Hall/CRC. Chap. 17 Rare Class Learning, pp. 445–468 (cit. on p. 40)

Aggarwal, C. C., ed. (2014). *Data Classification: Algorithms and Applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery. Chapman & Hall/CRC (cit. on pp. 31, 32, 34)

Aho, A. V., J. E. Hopcroft, and J. D. Ullman (1983). *Data Structures and Algorithms*. Ed. by M. Harrison. 1st. Addison-Wesley series in computer science and information processing. Addison-Wesley Longman Publishing Co., Inc. (cit. on p. 54)

Alpaydin, E. (2010). *Introduction to Machine Learning*. 2nd ed. Adaptive computation and machine learning. Cambridge, Massachusetts, London, England: MIT Press (cit. on pp. 31, 189)

Anders, G., A. Schiendorfer, F. Siefert, J.-P. Steghöfer, and W. Reif (2015). “Cooperative Resource Allocation in Open Systems of Systems”. In: *ACM Trans. Auton. Adapt. Syst.* 10.2, 11:1–11:44 (cit. on pp. 17, 21, 26, 28)

Anders, G., F. Siefert, J.-P. Steghöfer, H. Seebach, F. Nafz, and W. Reif (2010). “Structuring and Controlling Distributed Power Sources by Autonomous Virtual Power Plants”. In: *Proceedings of the Power & Energy Student Summit 2010 (PESS 2010)*, pp. 40–42 (cit. on pp. 5, 15, 17)

Asmus, P. (2010). “Microgrids, Virtual Power Plants and Our Distributed Energy Future”. In: *The Electricity Journal* 23.10, pp. 72–82 (cit. on pp. 2, 3)

Attenberg, J. and Ş. Ertekin (2013). “Class Imbalance and Active Learning”. In: *Imbalanced Learning*. John Wiley & Sons, Inc., pp. 101–149 (cit. on pp. 40, 42)

Bagnall, A., L. M. Davis, J. Hills, and J. Lines (2012). “Transformation Based Ensembles for Time Series Classification”. In: *Proceedings of the 12th SIAM International Conference on Data Mining*, pp. 307–318 (cit. on pp. 35, 36, 38, 45, 46, 51, 56, 57, 71, 72)

Bagnall, A., J. Lines, J. Hills, and A. Bostrom (2015). “Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles”. In: *Knowledge and Data Engineering, IEEE Transactions on* 27.9, pp. 2522–2535 (cit. on pp. 36, 45, 46, 57, 71, 72)

Bak, B. A. (2015). “High-Dimensional Classification”. PhD thesis. Aarhus University (cit. on pp. 34, 37, 43)

Bak, B. A. and J. L. Jensen (2016). “High dimensional classifiers in the imbalanced case”. In: *Computational Statistics & Data Analysis* 98, pp. 46–59 (cit. on p. 43)

Bánhalmi, A., A. Kocsor, and R. Busa-Fekete (2007). “Counter-Example Generation-Based One-Class Classification”. In: *Machine Learning: ECML 2007*. Ed. by J. N. Kok, J. Koronacki, R. L. d. Mantaras, S. Matwin, D. Mladenič, and A. Skowron. Vol. 4701. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 543–550 (cit. on pp. 50, 64)

Barnabé-Lortie, V., C. Bellinger, and N. Japkowicz (2015). “Active Learning for One-Class Classification”. In: *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 390–395 (cit. on p. 45)

Bartkowiak, A. M. (2010). “Anomaly, novelty, one-class classification: A short introduction”. In: *Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference on*, pp. 1–6 (cit. on p. 44)

Batista, G., D. Silva, and R. Prati (2012). “An Experimental Design to Evaluate Class Imbalance Treatment Methods”. In: *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*. Vol. 2, pp. 95–101 (cit. on p. 42)

Beer, S. (2013). “A formal model for agent-based Coalition Formation in Electricity Markets”. In: *Innovative Smart Grid Technologies Europe (ISGT EUROPE), 2013 4th IEEE/PES*, pp. 1–5 (cit. on p. 3)

Bellinger, C., S. Sharma, and N. Japkowicz (2012). “One-Class versus Binary Classification: Which and When?” In: *Machine Learning and Applications: ICMLA, 2012 11th International Conference on*. Vol. 2, pp. 102–106 (cit. on pp. 44, 64)

Bellman, R. and R. Bellman (1961). *Adaptive Control Processes: A Guided Tour*. ’Rand Corporation. Research studies. Princeton University Press (cit. on p. 37)

Bergstra, J. and Y. Bengio (2012). “Random Search for Hyper-parameter Optimization”. In: *J. Mach. Learn. Res.* 13, pp. 281–305 (cit. on p. 51)

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc. (cit. on p. 47)

– (2006). *Pattern Recognition and Machine Learning*. Ed. by M. Jordan, J. Kleinberg, and B. Schölkopf. 1st ed. Information Science and Statistics. Springer (cit. on p. 31)

Bitsch, R., W. Feldmann, and G. Aumayr (2002). “Virtuelle Kraftwerke – Einbindung dezentraler Energieerzeugungsanlagen”. In: *etz Elektrotechnik und Automation* 123.9, pp. 16–23 (cit. on p. 2)

Blachnik, M. (2014). “Ensembles of Instance Selection Methods based on Feature Subset”. In: *Procedia Computer Science* 35. Knowledge-Based and Intelligent Information &amp; Engineering Systems 18th Annual Conference, KES-2014 Gdynia, Poland, September 2014 Proceedings, pp. 388–396 (cit. on pp. 50, 64)

Blagus, R. and L. Lusa (2010). “Class prediction for high-dimensional class-imbalanced data”. In: *BMC Bioinformatics* 11.1, pp. 1–17 (cit. on p. 43)

Blank, M. (2015). “Reliability Assessment of Coalitions for the Provision of Ancillary Services”. online availbale at http://oops.uni-oldenburg.de/2635/. PhD thesis. Carl von Ossietzky Universität Oldenburg (cit. on p. 3)

Borgonovo, E. and E. Plischke (2016). “Sensitivity analysis: A review of recent advances”. In: *European Journal of Operational Research* 248.3, pp. 869–887 (cit. on p. 52)

Boser, B. E., I. M. Guyon, and V. N. Vapnik (1992). “A Training Algorithm for Optimal Margin Classifiers”. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT ’92. Pittsburgh, Pennsylvania, USA: ACM, pp. 144–152 (cit. on p. 47)

Bouveyron, C. and C. Brunet-Saumard (2014). “Model-based clustering of high-dimensional data: A review”. In: *Computational Statistics & Data Analysis* 71, pp. 52–78 (cit. on p. 63)

Braun, M. and P. Strauss (2008). “A Review on Aggregation Approaches of Controllable Distributed Energy Units in Electrical Power Systems”. In: *International Journal of Distributed Energy Resources* 4 (cit. on p. 2)

Breiman, L. (2001). “Random Forests”. English. In: *Machine Learning* 45.1, pp. 5–32 (cit. on p. 37)

Bremer, J., B. Rapp, and M. Sonnenschein (2011). “Encoding distributed search spaces for virtual power plants”. In: *Computational Intelligence Applications In Smart Grid (CIASG), 2011 IEEE Symposium on*, pp. 1–8 (cit. on p. 18)

Bremer, J. (2015). “Constraint-Handling mit Supportvektor-Dekodern in der verteilten Optimierung”. online available at http://oops.uni-oldenburg.de/id/eprint/2336/. PhD thesis. Carl von Ossietzky Universität Oldenburg (cit. on pp. 7, 11–13, 16, 19, 20, 26, 47, 79, 90, 152, 156, 157, 169, 171, 172, 175)

Bremer, J., B. Rapp, and M. Sonnenschein (2010). “Support vector based encoding of distributed energy resources’ feasible load spaces”. In: *Innovative Smart Grid Technologies Conference Europe IEEE PES* (cit. on pp. 13, 41, 46, 47, 98, 172)

Bremer, J. and M. Sonnenschein (2012). “A Distributed Greedy Algorithm for Constraint-based Scheduling of Energy Resources”. In: *FedCSIS*, pp. 1285–1292 (cit. on p. 10)

– (2013a). “Constraint-handling for Optimization with Support Vector Surrogate Models - A Novel Decoder Approach”. In: *ICAART (2)*, pp. 91–100 (cit. on p. 206)

– (2013b). “Kommunikation von Umweltkennzahlen im Smart Grid und deren Integration in die verteilte Wirkleistungsplanung”. In: *BUIS-Tage*, pp. 35–47 (cit. on p. 25)

– (2013c). “Sampling the Search Space of Energy Resources for Self-organized, Agent-based Planning of Active Power Provision”. In: *EnviroInfo*, pp. 214–222 (cit. on pp. 25, 206)

Brockwell, P. J. and R. A. Davis (1987). *Time Series: Theory and Methods*. Springer Series in Statistics. Springer New York (cit. on p. 204)

Buck, D. K. and A. A. Collins (2004). “POV-Ray - The Persistence of Vision Raytracer”. Computer software available from http://www.povray.org/ (cit. on p. 106)

Camerra, A., T. Palpanas, J. Shieh, and E. Keogh (2010). “iSAX 2.0: Indexing and Mining One Billion Time Series”. In: *Proceedings of the 2010 IEEE International Conference on Data Mining*. ICDM ’10. Washington, DC, USA: IEEE Computer Society, pp. 58–67 (cit. on p. 36)

Chandrashekar, G. and F. Sahin (2014). “A survey on feature selection methods”. In: *Computers & Electrical Engineering* 40.1. 40th-year commemorative issue, pp. 16–28 (cit. on p. 38)

Chapman, A. C., A. Rogers, N. R. Jennings, and D. S. Leslie (2011). “A unifying framework for iterative approximate best-response algorithms for distributed constraint optimization problems”. In: *The Knowledge Engineering Review* 26 (04), pp. 411–444 (cit. on p. 12)

Cortez, P. and M. Embrechts (2011). “Opening black box Data Mining models using Sensitivity Analysis”. In: *Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on*, pp. 341–348 (cit. on p. 52)

Cortez, P. and M. J. Embrechts (2013). “Using sensitivity analysis and visualization techniques to open black box data mining models”. In: *Information Sciences* 225, pp. 1–17 (cit. on p. 52)

Detemple, J. (2014). “Portfolio Selection: A Review”. In: *Journal of Optimization Theory and Applications* 161.1, pp. 1–21 (cit. on p. 10)

Ding, H., G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh (2008). “Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures”. In: *Proc. VLDB Endow.* 1.2, pp. 1542–1552 (cit. on pp. 36, 45)

Duda, R. and P. Hart (1973). *Pattern Classification and Scene Analysis*. Wiley (cit. on pp. 47, 48)

Electricity Industry EURELECTRIC, U. of the (2014). *Terminology*. online at www.eurelectric.org/facts-terminology/terminology/networks-grids/networks. last visited 24.02.2017 (cit. on p. 4)

Elish, M. O., T. Helmy, and M. I. Hussain (2013). “Empirical Study of Homogeneous and Heterogeneous Ensemble Models for Software Development Effort Estimation”. In: *Mathematical Problems in Engineering* 2013 (cit. on pp. 46, 71)

Engel, D., L. Hüttenberger, and B. Hamann (2012). “A Survey of Dimension Reduction Methods for High-dimensional Data Analysis and Visualization”. In: *Visualization of Large and Unstructured Data Sets: Applications in Geospatial Planning, Modeling and Engineering - Proceedings of IRTG 1131 Workshop 2011*. Ed. by C. Garth, A. Middel, and H. Hagen. Vol. 27. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 135–149 (cit. on p. 38)

Evangelista, P. F., M. J. Embrechts, and B. K. Szymanski (2006). “Applied Soft Computing Technologies: The Challenge of Complexity”. In: ed. by A. Abraham, B. de Baets, M. Köppen, and B. Nickolay. Berlin, Heidelberg: Springer Berlin Heidelberg. Chap. Taming the Curse of Dimensionality in Kernels and Novelty Detection, pp. 425–438 (cit. on p. 25)

Faltings, B. and M. Yokoo (2005). “Introduction: Special Issue on Distributed Constraint Satisfaction”. In: *Artificial Intelligence* 161.1–2. Distributed Constraint Satisfaction, pp. 1–5 (cit. on p. 15)

Frey, H. C., A. Mokhtari, and T. Danish (2003). *Evaluation of Selected Sensitivity Analysis Methods Based Upon Applications to Two Food Safety Process Risk Models*. Tech. rep. Prepared for: Office of Risk Assessment and Cost-Benefit Analysis U.S. Department of Agriculture Washington, DC. Department of Civil, Construction, and Environmental Engineering North Carolina State University (cit. on p. 52)

Fu, T.-c. (2011). “A review on time series data mining”. In: *Engineering Applications of Artificial Intelligence* 24.1, pp. 164–181 (cit. on p. 36)

Fulcher, B. and N. Jones (2014). “Highly Comparative Feature-Based Time-Series Classification”. In: *Knowledge and Data Engineering, IEEE Transactions on* 26.12, pp. 3026–3037 (cit. on pp. 35, 37, 57)

Fulcher, B. D., M. A. Little, and N. S. Jones (2013). “Highly comparative time-series analysis: the empirical structure of time series and their methods”. In: *Journal of The Royal Society Interface* 10.83 (cit. on p. 37)

Fuller, W. A. (1996). *Introduction to Statistical Time Series*. 2nd ed. Wiley & Sons Ltd (cit. on p. 204)

Garcia, S., J. Derrac, J. Cano, and F. Herrera (2012). “Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.3, pp. 417–435 (cit. on pp. 50, 64)

Gatterbauer, W. (2010). “Economic efficiency of decentralized unit commitment from a generator’s perspective”. In: *Engineering Electricity Services of the Future. Springer* (cit. on p. 16)

Gemert, W. van (2012). *Flexines*. online at http://www.flexines.org/. research project from 2008 - 2012 last visited 24.02.2017 (cit. on p. 6)

Gensler, A. and B. Sick (2014). “Novel Criteria to Measure Performance of Time Series Segmentation Techniques”. In: *Proceedings of the 16th LWA Workshops: KDML, IR and FGWM, Aachen, Germany, September 8-10, 2014*. Ed. by T. Seidl, M. Hassani, and C. Beecks. Vol. 1226. CEUR Workshop Proceedings. CEUR-WS.org, pp. 193–204 (cit. on p. 37)

Ghosh, S. S., C. D. Burns, D. Clark, K. Gorgolewski, Y. O. Halchenko, C. Madison, R. F. Tungaraza, and K. J. Millman (2009). *Neuroimaging in Python (previously called BrainPy)*. online at http://nipy.org/nitime/index.html (cit. on p. 204)

Gibert, K. and M. S.-M. V. Codina (2010). “Choosing the Right Data Mining Technique: Classification of Methods and Intelligent Recommendation”. In: *International Congress on Environmental Modelling and Software* (cit. on p. 33)

Giusti, R. and G. E. A. P. A. Batista (2013). “An Empirical Comparison of Dissimilarity Measures for Time Series Classification”. In: *Intelligent Systems (BRACIS), 2013 Brazilian Conference on*, pp. 82–88 (cit. on p. 36)

Golden, W. and P. Powell (2000). “Towards a definition of flexibility: in search of the Holy Grail?” In: *Omega* 28.4, pp. 373–384 (cit. on p. 5)

Görnitz, N. (2015). *tilitools*. online at https://github.com/nicococo/tilitools (cit. on p. 90)

Graaff, T. de (2015). “Implementierung eines Supportvektor-Dekoder Modells in Python”. Bachelorarbeit. Carl von Ossietzky Universität Oldenburg (cit. on pp. 90, 157)

Gritzmann, P. (2013). *Grundlagen der Mathematischen Optimierung*. 1st ed. Aufbaukurs Mathematik. Springer Vieweg (cit. on p. 12)

Guo, X., Y. Yin, C. Dong, G. Yang, and G. Zhou (2008). “On the Class Imbalance Problem”. In: *2008 Fourth International Conference on Natural Computation*. Vol. 4, pp. 192–201 (cit. on p. 42)

Hamby, D. M. (1994). “A review of techniques for parameter sensitivity analysis of environmental models”. In: *Environmental Monitoring and Assessment* 32 (cit. on pp. 51, 52)

Hamilton, J. D. (1994). *Time Series Analysis*. Princeton University Press (cit. on p. 56)

Hastie, T., R. Tibshirani, and J. H. Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer series in statistics. 4rd printing with corrections, Dec 2010. Springer (cit. on p. 31)

He, H. (2013). “Introduction”. In: *Imbalanced Learning*. John Wiley & Sons, Inc., pp. 1–12 (cit. on p. 39)

He, H. and E. Garcia (2009). “Learning from Imbalanced Data”. In: *Knowledge and Data Engineering, IEEE Transactions on* 21.9, pp. 1263–1284 (cit. on pp. 39, 42, 43, 48, 49, 65, 66)

He, X., G. Mourot, D. Maquin, J. Ragot, P. Beauseroy, A. Smolarz, and E. Grall-Maës (2014). “Multi-task learning with one-class SVM”. In: *Neurocomputing* 133, pp. 416–426 (cit. on pp. 38, 42, 62)

Heiselberg, P., H. Brohus, A. Hesselholt, H. Rasmussen, E. Seinre, and S. Thomas (2009). "Application of sensitivity analysis in design of sustainable buildings". In: *Renewable Energy* 34.9. Special Issue: Building and Urban Sustainability, pp. 2030–2036 (cit. on pp. 52, 53)

Hills, J., J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall (2014). "Classification of Time Series by Shapelet Transformation". In: *Data Min. Knowl. Discov.* 28.4, pp. 851–881 (cit. on p. 36)

Hinrichs, C. (2014). "Selbstorganisierte Einsatzplanung dezentraler Akteure im Smart Grid". online available at http://oops.uni-oldenburg.de/1960/. PhD thesis. Carl von Ossietzky Universität Oldenburg (cit. on pp. 3, 4, 10–13, 15–18, 28)

Hinrichs, C., J. Bremer, S. Martens, and M. Sonnenschein (2016). "Partitioning the Data Domain of Combinatorial Problems for Sequential Optimization". In: *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems*. Ed. by M. Ganzha, L. A. Maciaszek, and M. Paprzycki. Vol. 8. Annals of Computer Science and Information Systems, pp. 551–559 (cit. on p. 25)

Hinrichs, C., J. Bremer, and M. Sonnenschein (2013). "Distributed Hybrid Constraint Handling in Large Scale Virtual Power Plants". In: *IEEE PES Conference on Innovative Smart Grid Technologies Europe (ISGT Europe 2013)*. IEEE Power & Energy Society (cit. on pp. 13, 18, 206)

Hinrichs, C., S. Lehnhoff, and M. Sonnenschein (2014). "COHDA: A Combinatorial Optimization Heuristic for Distributed Agents". In: *Agents and Artificial Intelligence*. Ed. by J. Filipe and A. Fred. Vol. 449. Communications in Computer and Information Science. Springer, pp. 23–39 (cit. on pp. 16, 205)

Hinrichs, C., M. Sonnenschein, and S. Lehnhoff (2013). "Evaluation of a Self-Organizing Heuristic for Interdependent Distributed Search Spaces". In: *International Conference on Agents and Artificial Intelligence (ICAART 2013)*. Ed. by J. Filipe and A. L. N. Fred. Vol. Volume 1 – Agents. SciTePress, pp. 25–34 (cit. on p. 16)

Al-Hmouz, R., W. Pedrycz, A. Balamash, and A. Morfeq (2015). "Description and classification of granular time series". In: *Soft Computing* 19.4, pp. 1003–1017 (cit. on p. 62)

Hoens, T. R. and N. V. Chawla (2013). "Imbalanced Datasets: From Sampling to Classifiers". In: *Imbalanced Learning*. John Wiley & Sons, Inc., pp. 43–59 (cit. on pp. 42, 49)

Hunter, J. D. (2007). “Matplotlib: A 2D graphics environment”. In: *Computing In Science & Engineering* 9.3, pp. 90–95 (cit. on p. 90)

Ilić, M. (2007). “From Hierarchical to Open Access Electric Power Systems”. In: *Proceedings of the IEEE* 95.5, pp. 1060–1084 (cit. on p. 16)

Jankowski, N. and M. Grochowski (2004). “Comparison of Instances Seletion Algorithms I. Algorithms Survey”. English. In: *Artificial Intelligence and Soft Computing - ICAISC 2004*. Ed. by L. Rutkowski, J. Siekmann, R. Tadeusiewicz, and L. Zadeh. Vol. 3070. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 598–603 (cit. on pp. 50, 64)

Japkowicz, N. (2013). “Assessment Metrics for Imbalanced Learning”. In: *Imbalanced Learning*. John Wiley & Sons, Inc., pp. 187–206 (cit. on pp. 42, 49)

Jones, E., T. Oliphant, P. Peterson, et al. (2001). *SciPy: Open source scientific tools for Python*. [Online; accessed 2014-12-10] (cit. on p. 90)

Jurek, A., Y. Bi, S. Wu, and C. Nugent (2014). “A survey of commonly used ensemble-based classification techniques”. In: *The Knowledge Engineering Review* 29 (05), pp. 551–581 (cit. on pp. 46, 51, 71)

Kang, J. H. and S. B. Kim (2013). “A clustering algorithm-based control chart for inhomogeneously distributed TFT-LCD processes”. In: *International Journal of Production Research* 51.18, pp. 5644–5657 (cit. on p. 63)

Keogh, E. J., S. Chu, D. Hart, and M. Pazzani (2004). “Segmenting Time Series: A Survey and Novel Approach”. In: *Data Mining In Time Series Databases*. Ed. by M. Last, A. Kandel, and H. Bunke. Vol. 57. Series in Machine Perception and Artificial Intelligence. World Scientific Publishing Company. Chap. 1, pp. 1–22 (cit. on p. 36)

Khan, S. S. and M. G. Madden (2014). “One-class classification: taxonomy of study and review of techniques”. In: *The Knowledge Engineering Review* 29.3, pp. 345–374 (cit. on pp. 42, 44, 45)

Kleijnen, J. P. C. (2015). *Design and Analysis of Simulation Experiments*. International Series in Operations Research & Management Science. Springer (cit. on p. 52)

Klement, W., S. Wilk, W. Michalowski, and S. Matwin (2011). “Classifying Severely Imbalanced Data”. In: *Advances in Artificial Intelligence: 24th Canadian Conference on Artificial Intelligence*. Ed. by C. Butz and P. Lingras. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 258–264 (cit. on p. 42)

Knuth, D. E. (1976). “Big Omicron and Big Omega and Big Theta”. In: *SIGACT News* 8.2, pp. 18–24 (cit. on pp. 54, 141)

Köppen, M. (2000). “The curse of dimensionality”. In: *5th Online World Conference on Soft Computing in Industrial Applications (WSC5)*, pp. 4–8 (cit. on p. 25)

Kotsakos, D. and D. Gunopulos (2014). “Data Classification”. In: ed. by C. C. . Aggarwal. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Chapman & Hall/CRC. Chap. 13 Time Series Data Classification, pp. 365–378 (cit. on pp. 35, 36)

Kotsiantis, S., D. Kanellopoulos, and P. Pintelas (2006). “Handling imbalanced datasets: A review”. In: *GESTS Int Trans Comput Sci Eng* 30.1, pp. 25–36 (cit. on pp. 39, 42, 46, 50, 63)

Kouzelis, K. (2015). “Load and Flexibility Models for Distribution Grid Management”. PhD thesis. Department of Energy Technology (cit. on pp. 4, 6)

Kramer, O. (2010). “A Review of Constraint-handling Techniques for Evolution Strategies”. In: *Appl. Comp. Intell. Soft Comput.* 2010, 3:1–3:19 (cit. on p. 13)

– (2014). *A Brief Introduction to Continuous Evolutionary Optimization*. SpringerBriefs in Applied Sciences and Technology. Springer (cit. on pp. 12, 51, 149)

Krawczak, M. and G. Szkatuła (2014). “An approach to dimensionality reduction in time series”. In: *Information Sciences* 260, pp. 15–36 (cit. on p. 36)

Krawczyk, B. (2016). “Hybrid One-Class Ensemble for High-Dimensional Data Classification”. In: *Intelligent Information and Database Systems: 8th Asian Conference, ACIIDS 2016, Da Nang, Vietnam, March 14-16, 2016, Proceedings, Part II*. Ed. by T. N. Nguyen, B. Trawiński, H. Fujita, and T.-P. Hong. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 136–144 (cit. on pp. 38, 45)

Kuncheva, L. I. (2014). *Combining Pattern Classifiers: Methods and Algorithms*. 2nd. John Wiley & Sons (cit. on p. 31)

Lavine, M. (2009). *Introduction to Statistical Thought*. University Press of Florida (cit. on p. 189)

Li, J., G. Poulton, and G. James (2010). “Coordination of Distributed Energy Resource Agents”. In: *Applied Artificial Intelligence* 24.5, pp. 351–380 (cit. on p. 15)

Li, J., G. Poulton, and G. James (2008). *Distributed Energy Management*. Tech. rep. Patent WO 2008/014562 A1 (cit. on p. 15)

Liang, G. and C. Zhang (2012). “A Comparative Study of Sampling Methods and Algorithms for Imbalanced Time Series Classification”. In: *AI 2012: Advances in*

Artificial Intelligence: 25th Australasian Joint Conference. Ed. by M. Thielscher and D. Zhang. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 637–648 (cit. on p. 57)

Lin, W.-J. and J. J. Chen (2013). "Class-imbalanced classifiers for high-dimensional data". In: *Briefings in Bioinformatics* 14.1, pp. 13–26 (cit. on pp. 37, 43, 48–50, 63)

Lines, J. A. and A. Bagnall (2015). "Time series classification with ensembles of elastic distance measures". In: *Data Mining and Knowledge Discovery* 29.3, pp. 565–592 (cit. on pp. 35, 36, 57)

Liu, H., H. Motoda, B. Gu, F. Hu, C. R. Reeves, and D. R. Bush (2001). *Instance Selection and Construction for Data Mining*. Ed. by H. Liu and M. Hiroshi. 1st ed. Vol. 608. The Springer International Series in Engineering and Computer Science. Springer US (cit. on pp. 50, 64)

Liu, M. and F. F. Wu (2007). "Portfolio optimization in electricity markets". In: *Electric Power Systems Research* 77.8, pp. 1000–1009 (cit. on p. 10)

Liu, X.-Y. and Z.-H. Zhou (2013). "Ensemble Methods for Class Imbalance Learning". In: *Imbalanced Learning*. John Wiley & Sons, Inc., pp. 61–82 (cit. on pp. 42, 51)

López, V., A. Fernández, S. García, V. Palade, and F. Herreraz (2013). "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics". In: *Information Sciences* 250, pp. 113–141 (cit. on pp. 42, 43, 46, 55)

Lund, P. D., J. Lindgren, J. Mikkola, and J. Salpakari (2015). "Review of energy system flexibility measures to enable high levels of variable renewable electricity". In: *Renewable and Sustainable Energy Reviews* 45, pp. 785–807 (cit. on p. 5)

Lünsdorf, O. (2012). "Selbstorganisation virtueller Geräte für das Lastmanagement von Kleinverbrauchern". online available at http://oops.uni-oldenburg.de/1380/. PhD thesis. Carl von Ossietzky Universität Oldenburg (cit. on p. 6)

Lusa, L. and R. Blagus (2012). "The Class-Imbalance Problem for High-Dimensional Class Prediction". In: *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*. Vol. 2, pp. 123–126 (cit. on pp. 33, 43)

MacDougall, P., B. Roossien, C. Warmer, and K. Kok (2013). "Quantifying flexibility for smart grid services". In: *Power and Energy Society General Meeting (PES), 2013 IEEE*, pp. 1–5 (cit. on p. 6)

Maldonado, S., R. Weber, and F. Famili (2014). “Feature selection for high-dimensional class-imbalanced data sets using Support Vector Machines”. In: *Information Sciences* 286, pp. 228–246 (cit. on p. 43)

Markowitz, H. (1952). “Portfolio Selection”. In: *The Journal of Finance* 7.1, pp. 77–91 (cit. on p. 10)

Mazhelis, O. (2006). “One-class classifiers: a review and analysis of suitability in the context of mobile-masquerader detection”. In: *South African Computer Journal* 36, pp. 29–48 (cit. on p. 42)

McArthur, S. D. J., E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziargyriou, F. Ponci, and T. Funabashi (2007). “Multi-Agent Systems for Power Engineering Applications—-Part I: Concepts, Approaches, and Technical Challenges”. In: *IEEE Transactions on Power Systems* 22.4, pp. 1743–1752 (cit. on p. 16)

Menardi, G. and N. Torelli (2014). “Training and assessing classification rules with imbalanced data”. In: *Data Mining and Knowledge Discovery* 28.1, pp. 92–122 (cit. on p. 42)

Mika, S., B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch (1999). “Kernel PCA and De-noising in Feature Spaces”. In: *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*. Cambridge, MA, USA: MIT Press, pp. 536–542 (cit. on p. 20)

Molina, J., J. Garcia, A. Garcia, R. Melo, and L. Correia (2009). “Segmentation and Classification of Time-Series: Real Case Studies”. In: *Intelligent Data Engineering and Automated Learning - IDEAL 2009*. Ed. by E. Corchado and H. Yin. Vol. 5788. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 743–750 (cit. on p. 37)

Müller, R. (1990). *Rauschen*. 2nd ed. Vol. 15. Halbleiter-Elektronik. Springer (cit. on pp. 203, 204)

Naeini, M. P., B. Moshiri, B. N. Araabi, and M. Sadeghi (2014). “Learning by abstraction: Hierarchical classification model using evidential theoretic approach and Bayesian ensemble model”. In: *Neurocomputing* 130. Track on Intelligent Computing and Applications Selected papers from the 2012 International Workshop on Information, Intelligence and Computing (IWIIC 2012) Complex Learning in Connectionist Networks Selected papers from the World Congress on Nature and Biologically Inspired Computing (NaBIC), pp. 73–82 (cit. on p. 38)

Al-Naymat, G. and J. Taheri (2008). “Effects of dimensionality reduction techniques on time series similarity measurements”. In: *2013 ACS International Conference on Computer Systems and Applications (AICCSA)*, pp. 188–195 (cit. on p. 36)

Neugebauer, J., J. Bremer, C. Hinrichs, O. Kramer, and M. Sonnenschein (2016). “Generalized Cascade Classification Model with Customized Transformation Based Ensembles”. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE (cit. on pp. 55, 79, 119)

Neugebauer, J., O. Kramer, and M. Sonnenschein (2015). “Classification Cascades of Overlapping Feature Ensembles for Energy Time Series Data”. In: *Data Analytics for Renewable Energy Integration*. Ed. by W. L. Woon, Z. Aung, and S. Madnick. Vol. 9518. Lecture Notes in Computer Science. Springer, pp. 76–93 (cit. on pp. 55, 58, 60, 79)

– (2016). “Improving Cascade Classifier Precision by Instance Selection and Outlier Generation”. In: *ICAART (8)*. Ed. by J. van den Herik and J. Filipe (cit. on pp. 55, 65, 79)

– (2017). “Instance Selection and Outlier Generation to Improve the Cascade Classifier Precision”. In: *Agents and Artificial Intelligence: 8th International Conference, ICAART 2016, Rome, Italy, February 24-26, 2016, Revised Selected Papers*. Ed. by J. van den Herik and J. Filipe. Vol. 10162. Lecture Notes in Computer Science. Springer International Publishing, pp. 151–170 (cit. on p. 79)

Nguyen, A.-T. and S. Reiter (2015). “A performance comparison of sensitivity analysis methods for building energy models”. In: *Building Simulation* 8.6, pp. 651–664 (cit. on p. 51)

Nieße, A., S. Lehnhoff, M. Tröschel, M. Uslar, C. Wissing, H. J. Appelrath, and M. Sonnenschein (2012). “Market-based self-organized provision of active power and ancillary services: An agent-based approach for Smart Distribution Grids”. In: *Complexity in Engineering (COMPENG), 2012*, pp. 1–5 (cit. on p. 3)

Nieße, A. (2015). “Verteilte kontinuierliche Einsatzplanung in Dynamischen Virtuellen Kraftwerken”. online availbale at http://oops.uni-oldenburg.de/2416/. PhD thesis. Carl von Ossietzky Universität Oldenburg (cit. on pp. 3, 10, 11, 16–18)

Nieße, A., S. Beer, J. Bremer, C. Hinrichs, O. Lünsdorf, and M. Sonnenschein (2014). “Conjoint Dynamic Aggregation and Scheduling Methods for Dynamic Virtual Power Plants.” In: *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*. Ed. by M. Ganzha, L. A. Maciaszek, and M. Paprzycki.

Vol. 2. Annals of Computer Science and Information Systems. IEEE, pp. 1505–1514 (cit. on p. 10)

Nikonowicz, Ł. B. and J. Milewski (2012). "Virtual Power Plants - general review: structure, application and optimization". In: *Journal of Power Technologies* 92.3, pp. 135–149 (cit. on pp. 2, 3)

Oliphant, T. E. (2006). *Guide to NumPy*. Provo, UT (cit. on p. 90)

Ottmann, T. and P. Widemayer (2012). *Algorithmen und Datenstrukturen*. Spektrum Akademischer Verlag (cit. on pp. 54, 141)

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830 (cit. on p. 90)

Peña Landaburu, Y. K. (2006). "Optimal Allocation and Scheduling of Demand in Deregulated Energy Markets". PhD thesis. Technische Universität Wien (cit. on p. 15)

Petersen, M., K. Edlund, L. Hansen, J. Bendtsen, and J. Stoustrup (2013). "A taxonomy for modeling flexibility and a computationally efficient algorithm for dispatch in Smart Grids". In: *American Control Conference (ACC), 2013*. IEEE, pp. 1150–1156 (cit. on pp. 5, 7)

Piao, Y., H. W. Park, C. H. Jin, and K. H. Ryu (2014). "Ensemble method for classification of high-dimensional data". In: *Big Data and Smart Computing (BIGCOMP), 2014 International Conference on*, pp. 245–249 (cit. on pp. 38, 62)

Pinedo, M. L. (2012). *Scheduling: Theory, Algorithms, and Systems*. 4th ed. Originally published by Prentice-Hall 1995. Springer (cit. on p. 9)

Plancke, G., K. De Vos, R. Belmans, and A. Delnooz (2015). "Virtual power plants: Definition, applications and barriers to the implementation in the distribution system". In: *European Energy Market (EEM), 2015 12th International Conference on the*, pp. 1–5 (cit. on p. 2)

Pournaras, E. (2013). "Multi-level Reconfigurable Self-organization in Overlay Services". PhD thesis. Technische Universiteit Delft (cit. on p. 15)

Press, O. U. (2016). *Oxford Dictionaries*. online at http://www.oxforddictionaries.com/de/definition/englisch/flexibility. last visited 24.02.2017 (cit. on p. 5)

Priore, P., A. Gómez, R. Pino, and R. Rosillo (2014). "Dynamic scheduling of manufacturing systems using machine learning: An updated review". In: *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 28, pp. 83–97 (cit. on p. 11)

Pudjianto, D., C. Ramsay, and G. Strbac (2007). “Virtual power plant and system integration of distributed energy resources”. In: *IET Renewable Power Generation* 1.1, pp. 10–16 (cit. on p. 2)

Radovanović, M., A. Nanopoulos, and M. Ivanović (2010). “Time-Series Classification in Many Intrinsic Dimensions”. In: *SIAM International Conference on Data Mining* (cit. on p. 73)

Ranawana, R. and V. Palade (2006). “Multi-Classifier Systems: Review and a Roadmap for Developers”. In: *Int. J. Hybrid Intell. Syst.* 3.1, pp. 35–61 (cit. on p. 51)

Raskutti, B. and A. Kowalczyk (2004). “Extreme Re-balancing for SVMs: A Case Study”. In: *SIGKDD Explor. Newsl.* 6.1, pp. 60–69 (cit. on p. 43)

Rokach, L. (2010). “Ensemble-based classifiers”. In: *Artificial Intelligence Review* 33.1-2, pp. 1–39 (cit. on pp. 46, 51, 71)

Roossien, B. (2012). *Mathematical quantification of near realtime flexibility for Smart Grids, Flexines D8.1*. project report D8.1. available online at http://www.flexines.org/publicaties/eindrapport/BIJLAGE14a.pdf. Groningen: Energy research Centre of the Netherlands (ECN) (cit. on pp. 6, 7)

Rossi, F., P. v. Beek, and T. Walsh (2006). *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. New York, NY, USA: Elsevier Science Inc. (cit. on p. 12)

Saboori, H., M. Mohammadi, and R. Taghe (2011). “Virtual Power Plant (VPP), Definition, Concept, Components and Types”. In: *Power and Energy Engineering Conference (APPEEC), 2011 Asia-Pacific*, pp. 1–4 (cit. on p. 2)

Sachs, A., C. Thiel, and F. Schwenker (2009). “One-class supportvector machines for the classification of bioacoustic time series”. In: *ICGST International Journal on Artificial Intelligence and Machine Learning (AIML)*, p. 2006 (cit. on p. 45)

Saeys, Y., I. Inza, and P. Larrañaga (2007). “A review of feature selection techniques in bioinformatics”. In: *Bioinformatics* 23.19, pp. 2507–2517 (cit. on p. 38)

Sammut, C. and G. I. Webb, eds. (2010). *Encyclopedia of Machine Learning*. Springer US (cit. on pp. 31, 35, 37, 38, 56)

Saravanan, B., S. Das, S. Sikri, and D. P. Kothari (2013). “A solution to the unit commitment problem–-a review”. In: *Frontiers in Energy* 7.2, pp. 223–236 (cit. on p. 14)

Schiendorfer, A., G. Anders, J.-P. Steghöfer, and W. Reif (2015). “Abstraction of Heterogeneous Supplier Models in Hierarchical Resource Allocation”. In: *Transactions* *on Computational Collective Intelligence XX*. Ed. by N. T. Nguyen, R. Kowalczyk, B. Duval, J. van den Herik, S. Loiseau, and J. Filipe. Vol. 9420. Lecture Notes in Computer Science. Springer International Publishing, pp. 23–53 (cit. on pp. 17, 21, 22, 27, 169)

Schiendorfer, A., C. Lassner, G. Anders, W. Reif, and R. Lienhart (2015a). “Active Learning for Abstract Models of Collectives”. In: *3rd Workshop on Self-optimisation in Organic and Autonomic Computing Systems (SAOS)* (cit. on p. 23)

– (2015b). “Active Learning for Efficient Sampling of Control Models of Collectives”. In: *International Conference on Self-adaptive and Self-organizing Systems (SASO)* (cit. on p. 23)

Schiendorfer, A., J.-P. Steghofer, and W. Reif (2014). “Synthesised constraint models for distributed energy management”. In: *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, pp. 1529–1538 (cit. on p. 22)

Schiendorfer, A., J.-P. Steghöfer, and W. Reif (2014). “Synthesis and Abstraction of Constraint Models for Hierarchical Resource Allocation Problems”. In: *Proc. 6th Int. Conf. Agents and Artificial Intelligence (ICAART’14)*. Vol. 2. SciTePress, pp. 15–27 (cit. on pp. 17, 22, 27)

Schmid, J. (2009). *FENIX - Flexibile Electricity Networks to Integrate the expected ’energy evolution’*. online at http://fenix.iwes.fraunhofer.de/home.htm. research project from 2005 - 2009 last visited 24.02.2017 (cit. on p. 2)

Schölkopf, B., J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson (2001). “Estimating the Support of a High-Dimensional Distribution”. In: *Neural Comput.* 13.7, pp. 1443–1471 (cit. on p. 46)

Seo, M. and S. Oh (2013). “A novel divide-and-merge classification for high dimensional datasets”. In: *Computational Biology and Chemistry* 42, pp. 23–34 (cit. on pp. 38, 62)

Serrà, J. and J. L. Arcos (2014). “An empirical evaluation of similarity measures for time series classification”. In: *Knowledge-Based Systems* 67, pp. 305–314 (cit. on p. 36)

Shanab, A. A., T. M. Khoshgoftaar, R. Wald, and J. Van Hulse (2011). “Comparison of approaches to alleviate problems with high-dimensional and class-imbalanced data”. In: *Information Reuse and Integration (IRI), 2011 IEEE International Conference on*, pp. 234–239 (cit. on p. 43)

Shang, Y.-W. and Y.-H. Qiu (2006). “A Note on the Extended Rosenbrock Function”. In: *Evol. Comput.* 14.1, pp. 119–126 (cit. on p. 182)

Sim, K., V. Gopalkrishnan, A. Zimek, and G. Cong (2013). “A survey on enhanced subspace clustering”. English. In: *Data Mining and Knowledge Discovery* 26.2, pp. 332–397 (cit. on p. 63)

Skou, A. (2015). *TotalFLex*. online at http://www.totalflex.dk/. research project from 2012 - 2015 last visited 24.02.2017 (cit. on p. 6)

Sonnenschein, M. and L. Hofmann (2015). *Smart Nord*. online at http://www.smartnord.de. research project from 2012 - 2015 last visited 24.02.2017 (cit. on pp. 1, 7, 175)

Spiegel, S. (2015). “Time Series Distance Measures Segmentation, Classification, and Clustering of Temporal Data”. PhD thesis. Technischen Universität Berlin (cit. on p. 36)

Steck, M. H. E. (2013). “Entwicklung und Bewertung von Algorithmen zur Einsatzplanerstellung virtueller Kraftwerke”. PhD thesis. Technische Universität München (cit. on p. 15)

Stefanowski, J. (2016). “Dealing with Data Difficulty Factors While Learning from Imbalanced Data”. In: *Challenges in Computational Statistics and Data Mining*. Ed. by S. Matwin and J. Mielniczuk. Cham: Springer International Publishing, pp. 333–363 (cit. on pp. 34, 41, 42, 46, 55)

Steghöfer, J.-P., G. Anders, F. Siefert, and W. Reif (2013). “A System of Systems Approach to the Evolutionary Transformation of Power Management Systems”. In: *Proceedings of INFORMATIK 2013 – Workshop on “Smart Grids”*. Vol. P-220. Lecture Notes in Informatics. Bonner Köllen Verlag (cit. on pp. 11, 17)

Sudjianto, A., S. Nair, M. Yuan, A. Zhang, D. Kern, and F. Cela-Díaz (2010). “Statistical Methods for Fighting Financial Crimes”. In: *Technometrics* 52.1, pp. 5–19 (cit. on pp. 33, 42)

Sun, Y., A. K. C. Wong, and M. S. Kamel (2009). “CLASSIFICATION OF IMBALANCED DATA: A REVIEW”. In: *International Journal of Pattern Recognition and Artificial Intelligence* 23.04, pp. 687–719 (cit. on pp. 42, 49)

Sutton, C., M. Sindelar, and A. McCallum (2005). *Feature Bagging: Preventing Weight Undertraining in Structured Discriminative Learning*. IR 402. Department of Computer Science, University of Massachusetts Amherst (cit. on p. 37)

Tax, D. M. J. and R. P. W. Duin (2002). “Uniform Object Generation for Optimizing One-class Classifiers”. In: *J. Mach. Learn. Res.* 2, pp. 155–173 (cit. on pp. 50, 65)

Tax, D. M. J. (2001). “One-class classification – Concept– learning in the anbsence of counter–examples”. PhD thesis. Delft University of Technology (cit. on pp. 31, 33, 44, 45, 47, 49, 125)

Tax, D. (2013a). *DDtools, the Data Description Toolbox for Matlab*. version 2.0.0 (cit. on pp. 90, 179)

– (2013b). *PRTools, Matlab Pattern Recognition Toolbox for representation and generalization*. version 5.0 (cit. on p. 90)

Thomas, B. (2007). *Mini-Blockheizkraftwerke: Grundlagen, Gerätetechnik, Betriebsdaten*. 1st ed. Würzburg: Vogel Verlag Und Druck (cit. on p. 171)

Tomašev, N., K. Buza, K. Marussy, and P. B. Kis (2015). “Feature Selection for Data and Pattern Recognition”. In: ed. by U. Stańczyk and C. L. Jain. Berlin, Heidelberg: Springer Berlin Heidelberg. Chap. Hubness-Aware Classification, Instance Selection and Feature Construction: Survey and Extensions to Time-Series, pp. 231–262 (cit. on pp. 36, 50, 64, 73)

Tomašev, N. and D. Mladenić (2013). “Hubness-aware shared neighbor distances for high-dimensional *k*-nearest neighbor classification”. In: *Knowledge and Information Systems* 39.1, pp. 89–122 (cit. on pp. 48, 73)

Tröschel, M. (2010). “Aktive Einsatzplanung in holonischen Virtuellen Kraftwerken”. online available at http://www.uni-oldenburg.de/fileadmin/user_upload/informatik/download/Promotionen/Dissertation_Martin_Troeschl.pdf. PhD thesis (cit. on pp. 3, 5, 11, 15)

Tröschel, M. and H.-J. Appelrath (2009). “Multiagent System Technologies: 7th German Conference, MATES 2009, Hamburg, Germany, September 9-11, 2009. Proceedings”. In: ed. by L. Braubach, W. Hoek, P. Petta, and A. Pokahr. Berlin, Heidelberg: Springer Berlin Heidelberg. Chap. Towards Reactive Scheduling for Large-Scale Virtual Power Plants, pp. 141–152 (cit. on pp. 11, 13)

Tsai, C.-F., W. Eberle, and C.-Y. Chu (2013). “Genetic algorithms in feature and instance selection”. In: *Knowledge-Based Systems* 39, pp. 240–247 (cit. on pp. 50, 64)

Tsay, R. S. (2005). *Analysis of Financial Time Series*. 2nd ed. John Wiley & Sons, Inc. (cit. on p. 204)

Tsimpiris, A. and D. Kugiumtzis (2012). “Feature selection for classification of oscillating time series”. In: *Expert Systems* 29.5, pp. 456–477 (cit. on p. 36)

Valentini, G. (2014). “Hierarchical Ensemble Methods for Protein Function Prediction”. In: *ISRN Bioinformatics* 2014. review articlaé (cit. on p. 38)

Valsomatzis, E., K. Hose, T. B. Pedersen, and L. Šikšnys (2015). “Measuring and Comparing Energy Flexibilities”. In: *Proceedings of the Joint EDBT/ICDT 2015 Workshops, EDBT’15*. Vol. 1330. CEUR Workshop Proceedings 1330. Brussels, Belgium: CEUR Workshop Proceedings 1330, pp. 78–85 (cit. on p. 6)

Van Hulse, J., T. M. Khoshgoftaar, and A. Napolitano (2007). “Experimental Perspectives on Learning from Imbalanced Data”. In: *Proceedings of the 24th International Conference on Machine Learning*. ICML ’07. Corvalis, Oregon: ACM, pp. 935–942 (cit. on p. 42)

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Information Science and Statistics. New York, NY, USA: Springer-Verlag New York, Inc. (cit. on p. 47)

Wang, Z., Z. Zhao, S. Weng, and C. Zhang (2015). “Solving one-class problem with outlier examples by {SVM}”. In: *Neurocomputing* 149, Part A. Advances in neural networksAdvances in Extreme Learning MachinesSelected papers from the Tenth International Symposium on Neural Networks (ISNN 2013)Selected articles from the International Symposium on Extreme Learning Machines (ELM 2013), pp. 100–105 (cit. on p. 38)

Weiss, G. M. (2004). “Mining with Rarity: A Unifying Framework”. In: *SIGKDD Explor. Newsl.* 6.1, pp. 7–19 (cit. on pp. 33, 39, 40)

– (2013). “Foundations of Imbalanced Learning”. In: *Imbalanced Learning*. John Wiley & Sons, Inc., pp. 13–41 (cit. on pp. 39, 42)

Wetterdienst, D. (2010). *Testreferenzjahre von Deutschland für mittlere und extreme Witterungsverhältnisse (TRY), TRY Region 3*. online at http://www.dwd.de/DE/leistungen/testreferenzjahre/testreferenzjahre.html;jsessionid=86822DBA17F342D8C522A43564EA801E.live21073?nn=507312. last visited 07.06.2017 (cit. on p. 175)

Whalen, S. and G. Pandey (2013). “A Comparative Analysis of Ensemble Classifiers: Case Studies in Genomics”. In: *2013 IEEE 13th International Conference on Data Mining*, pp. 807–816 (cit. on pp. 46, 71)

Wilson, D. and T. Martinez (2000). “Reduction Techniques for Instance-Based Learning Algorithms”. English. In: *Machine Learning* 38.3, pp. 257–286 (cit. on p. 64)

Wu, F. F., K. Moslehi, and A. Bose (2005). “Power System Control Centers: Past, Present, and Future”. In: *Proceedings of the IEEE* 93.11, pp. 1890–1908 (cit. on pp. 16, 28)

Wu, J., R. Dhingra, M. Gambhir, and J. V. Remais (2013). “Sensitivity analysis of infectious disease models: methods, advances and their application”. In: *Journal of The Royal Society Interface* 10.86 (cit. on p. 52)

Xiong, N. and P. Funk (2008). “Concise case indexing of time series in health care by means of key sequence discovery”. In: *Applied Intelligence* 28, pp. 247–260 (cit. on p. 36)

Yang, H., I. King, and M. Lyu (2010). “Multi-task Learning for one-class classification”. In: *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pp. 1–8 (cit. on pp. 42, 43, 45, 62)

Yin, H. and K. Gai (2015). “An Empirical Study on Preprocessing High-Dimensional Class-Imbalanced Data for Classification”. In: *High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on*, pp. 1314–1319 (cit. on pp. 38, 42, 43)

Zhang, S., S. Sadaoui, and M. Mouhoub (2015). “An Empirical Analysis of Imbalanced Data Classification”. In: *Computer and Information Science* 8.1, pp. 151–162 (cit. on p. 42)

Zhuang, L. and H. Dai (2006). “Parameter Optimization of Kernel-Based One-Class Classifier on Imbalance Text Learning”. English. In: *PRICAI 2006: Trends in Artificial Intelligence*. Ed. by Q. Yang and G. Webb. Vol. 4099. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 434–443 (cit. on pp. 42, 47, 51, 64)

Hiermit erkläre ich, die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben.

Oldenburg, June 22, 2017