CARL
VON
OSSIETZKY
universität OLDENBURG

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften

Department für Informatik

Wind Power Prediction with Machine Learning Ensembles

Dissertation zur Erlangung des Grades eines
Doktors der Naturwissenschaften

vorgelegt von

Justin Philipp Heinermann, M.Sc.

21. September 2016

Tag der Disputation
05. Dezember 2016

Prüfungskommission
Apl. Prof. Dr.-Ing. habil. Jürgen Sauer (Vorsitzender)
Jun.-Prof. Dr. habil. Oliver Kramer (Erstgutachter)
Prof. Dr. Jörg Lässig (Zweitgutachter)
Dr. Marco Grawunder (Mitglied der wiss. Mitarbeiter)

Zusammenfassung

Eine erfolgreiche und nachhaltige Integration von Windenergieanlagen in das Stromnetz erfordert präzise und verlässliche Prognosemethoden. Mit einem steigenden Anteil von Windenergie an den Strommärkten und einer wachsenden Anzahl installierter Anlagen ist auch ein gesteigertes Interesse an Kurzfristprognosen der erzeugten Leistung festzustellen. Für diese Anwendung bieten Methoden des maschinellen Lernens die Möglichkeit einer rein datenbasierten, raumzeitlichen Prognose, die oft bessere Ergebnisse erzielt als die traditionell verwendeten numerischen Wetterprognosen. Die beiden größten Herausforderungen für den Einsatz maschineller Lernverfahren sind die weitere Steigerung der Prognosegüte sowie die effiziente Berechnung in einer angemessenen Zeit.

Diese Arbeit stellt ein Vorhersageframework vor, das auf heterogenen Ensemblemodellen basiert. Ein Ensemble kombiniert eine Vielzahl möglichst unterschiedlicher Modelle, die mit maschinellen Lernverfahren trainiert werden. Zunächst werden Ensembles untersucht, die jeweils nur einen Basisalgorithmus und Samplingtechniken verwenden. Im Gegensatz zum Stand der Technik kann ein geringerer Vorhersagefehler erzielt werden. Anschließend wird gezeigt, dass heterogene Ensembles mit mehreren unterschiedlichen Vorhersagealgorithmen eine verbesserte Prognosegenauigkeit aufweisen. Mit diesem Verfahren kann außerdem die Trainingszeit stark verkürzt werden. Insbesondere lässt sich durch die Parameterwahl des Ensemblemodells ein guter Kompromiss zwischen Prognosegüte und benötigter Berechnungszeit erreichen. Für die Optimierung des Modells kommt in dieser Arbeit eine naturinspirierte Heuristik in Form von evolutionärer Mehrzieloptimierung zum Einsatz, die eine effiziente Parametersuche gewährleistet.

Abstract

For a sustainable integration of wind power into the electricity grid, precise and robust predictions are required. With increasing installed capacity and changing energy markets, there is a growing demand for short-term predictions. Machine learning methods can be used as a purely data-driven, spatio-temporal prediction model that yields better results than traditional physical models based on weather simulations. However, there are two big challenges when applying machine learning techniques to the domain of wind power predictions. First, when applying state-of-the-art algorithms to big training data sets, the required computation times may increase to an unacceptable level. Second, the prediction performance and reliability have to be improved to cope with the requirements of the energy markets.

This thesis proposes a robust and practical prediction framework based on heterogeneous machine learning ensembles. Ensemble models combine the predictions of numerous and preferably diverse models to reduce the prediction error. First, homogeneous ensemble regressors that employ a single base algorithm are analyzed. Further, the construction of heterogeneous ensembles is proposed. These models employ multiple base algorithms and benefit from a gain of diversity among the combined predictors. A comprehensive experimental evaluation shows that the combination of different techniques to an ensemble outperforms state-of-the-art prediction models while requiring a shorter runtime. Finally, a framework for model selection based on evolutionary multi-objective optimization is presented. The method offers an efficient and comfortable balancing of a preferably low prediction error and a moderate computational cost.

Contents

Part I

Introduction

1 Introduction and Overview

For a sustainable integration of wind power into the electricity grid, a precise and reliable short-term prediction method is required. A purely data-driven approach based on machine learning and a spatio-temporal model yields a good prediction accuracy. There are two big challenges for applying the machine learning technique. First, the prediction error and reliability have to be improved to cope with the energy markets' requirements. Second, the required computation times need to be reduced to an acceptable level.

In this thesis, a robust and practical prediction framework based on machine learning ensembles is proposed. Homogeneous ensembles employing only one base algorithm and heterogeneous ensembles employing different algorithms are investigated. The novel approach can reduce both the prediction error and the required computation time. For the model selection, a multi-objective evolutionary optimization approach is used. It offers an efficient and comfortable balancing of the objectives of prediction performance and computational cost.

This chapter is structured as follows: Section 1.1 gives a motivation for the research. The proposed solution is briefly introduced in Section 1.2. The chapters of this thesis are summarized in Section 1.3 to give an overview on the thesis. The remainder of this work will be written in a scientific style with the use of "we" rather than "I".

1.1 Motivation

In the recent years, a strong increase in wind energy can be observed. For example, there is an installed wind energy capacity of 142 GW in the European Union (EU) [106]. 12,800 MW of wind energy capacity was installed in the EU in 2015. Figure 1.1 shows the volumes of wind energy produced in Germany for 2005-2015. It can be seen that wind energy production is strongly increasing, which makes the research topic of good quality predictions relevant. With growing capacity, there is also a growing demand of reliable and precise forecasts for the energy provided by the highly volatile wind. The trading of wind energy at the power markets is only possible with good forecasts. Here, a trend towards shorter forecast horizons can be noticed. For instance, the lead time has been reduced to 30 minutes on all intraday markets of the EPEX Spot [27]. Other applications for short-term power predictions are planning of control energy, ensuring power grid stability, and charging of storage systems.

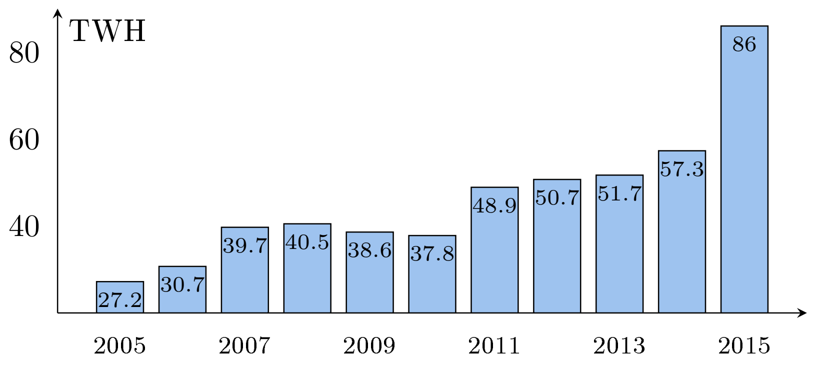


Fig. 1.1. Yearly Wind Energy Production in Germany, cf. [6].

Besides the traditional numerical weather predictions (NWPs), machine learning methods can be used as a purely data-driven, spatio-temporal prediction model and yield better results for short-term forecast horizons. The general approach is depicted in Figure 1.2. The model is trained only on historical data instead of computing physical model simulations. The data are split into training

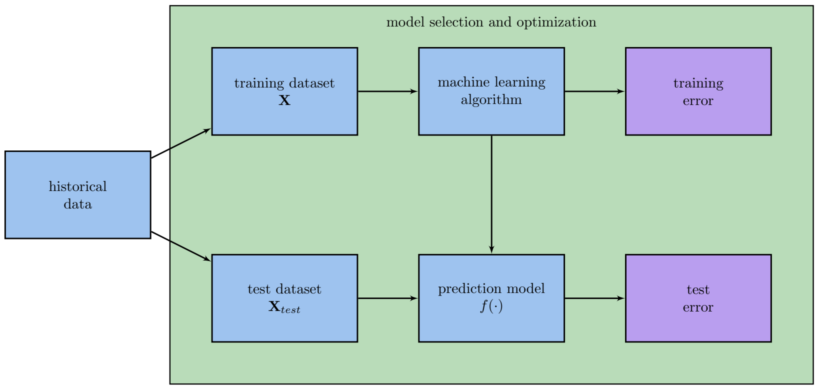

model selection and optimization
historical data
training dataset $\mathbf{X}$
machine learning algorithm
training error
test dataset $\mathbf{X}_{test}$
prediction model $f(\cdot)$
test error

Fig. 1.2. Training and selection of an appropriate machine learning model.

and test set. Using the training data, a machine learning model is trained. The model is based on a particular algorithm and chosen parameters. In order to optimize these choices, different parameters are used and afterwards an expected test error is computed for model assessment. A more detailed view on the machine learning method is given in Chapter 2.

We have to deal with two challenges when applying these statistical learning techniques to the field of wind power prediction: First, the prediction performance and reliability need to be improved further. Second, in order to achieve the best prediction accuracy possible with machine learning techniques, the computation time can grow tremendously. In particular, the training data sets are preferably as big as possible but slow down the learning process. Training or evaluating a support vector regression (SVR) model, k-nearest neighbors (k-NN) or artificial neural networks (ANNs) can easily take hours. Further, finding optimal parameters for the algorithms can be difficult and expensive. The required computation times must be reduced for practical application.

1.2 Contribution of this Thesis

In this thesis, we propose a prediction framework based on heterogeneous machine learning ensembles. The basic idea of ensemble models is to use many diverse prediction models and combine them to a more powerful prediction. Because the

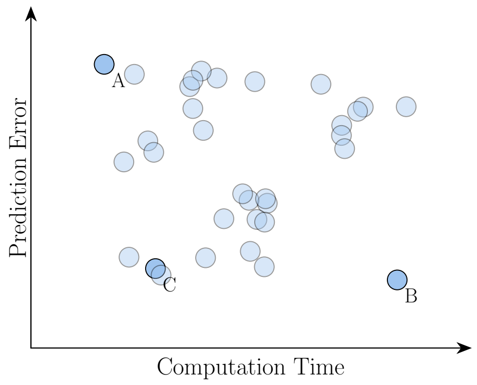


Fig. 1.3. Different models offer different prediction errors and computational requirements.

single models show different behavior, the overall prediction performance can be improved substantially. We first analyze homogeneous ensemble regressors that make use of a single base algorithm and compare decision trees (DTs) to k-NN and SVR with different parameter setups. A weighted bagging approach yields substantially improved prediction results. As next step, we construct heterogeneous ensembles that make use of multiple base algorithms and benefit from a gain of diversity among the combined predictors. In a comprehensive experimental evaluation, we show that the combination of different techniques to an ensemble outperforms state-of-the-art predictors.

In addition to the improved prediction error, we show that the ensemble models require a shorter runtime. With the proposed hybrid approach, the user of the framework is able to balance the computational cost and the prediction performance, which is desirable. The schema in Figure 1.3 shows the behavior of different solutions and the basic idea of our proposed model selection approach. Each point in the plot represents a different solution that can be achieved by choosing a particular prediction model setting. Especially for ensembles, the runtime and error depend on various parameters. For instance, an ensemble based on just a few DTs can be computed faster than an ensemble consisting of a large number of ANNs. In Figure 1.3, we highlighted three points that visualize the quality of possible solutions. While solution A is very fast, it cannot offer a really good prediction error. In contrast, solution B yields a prediction error that is the best found among all models in the plot – but takes a large computation time. Solution C is a solution most machine learning practitioners would prefer both over A and B because it offers a moderate computation time while yielding a very low prediction error.

In this work, we show that ensembles in general and heterogeneous ensembles in particular are well-suited for the task of finding a good model while searching for a compromise between small computational cost and a low prediction error. One reason is the use of small random samples of the training data and random feature subsets of the patterns, which reduces the computation time of the single models in the ensemble. Computation time and prediction error can be controlled by the number of models and sample sizes among other parameters. The second reason is that an increased diversity in the heterogeneous ensemble can be helpful for the balancing of the two optimization objectives. When using a heterogeneous choice of base models, the diversity among the predictors leads to a decreased number of models that are needed for achieving an acceptable prediction error.

For the optimization and selection of the parameter settings, we employ evolutionary multi-objective optimization algorithms (EMOAs) and show that these nature-inspired heuristics are well-suited for an efficient and comfortable model selection. The practitioner obtains a Pareto set of solutions which give the best trade-off between a good error and a short computation time.

1.3 Structure of this Thesis

In the following, we give an overview on the thesis and a brief summary of each chapter. Parts of this work are based on results we have published in proceedings of peer-reviewed conferences and a journal, which are listed for each chapter.

Part II: Foundations

Chapter 2: Machine Learning

In Chapter 2, we introduce the machine learning methodology. Machine learning algorithms are statistical methods to gain knowledge from observed data, i.e., creating a mathematical model for predictions on new observations. It is nowadays applied in a wide range of disciplines comprising biology, chemistry, computer graphics, medicine, automotive applications, security, and many others. Because the focus of this work is on regression techniques, the chapter introduces and formalizes the idea of supervised learning. The k-NN and DT algorithms are introduced as basic methods. A very important aspect of machine learning is the model selection and parameter tuning, which are described as well.

Chapter 3: Wind Power Prediction

In the recent years, a strong increase in wind energy can be observed and there are numerous different applications for wind power prediction. We describe the use cases of wind power prediction and give a classification of different forecast horizons. Prediction techniques can be divided in two groups, forecasting based on NWP on the one hand and predictions based on the historical time series on the other hand. The state of the art methods are described, followed by the spatio-temporal regression model used in this work.

Chapter 4: Ensembles

A good alternative to the well-known machine learning algorithms is the use of ensembles, i.e. combining several basic models to an *ensemble predictor*. The prediction error can be decreased and ensembles can often offer a short computation time. In contrast to state-of-the-art machine learning algorithms, ensemble methods require less tuning and expert domain knowledge. This chapter introduces the relevant ensemble methods.

Part III: Ensembles for Wind Power Prediction

Chapter 5: Support Vector Regression Ensembles

We propose a novel SVR ensemble method for wind power prediction in order to improve the forecast quality and spend less computation time. Instead of using one single support vector regressor, we train a number of weak SVR regressors with a weighted bagging approach. We investigate different parameter choice strategies as well as different weighting methods. Compared to state-of-the-art SVR prediction, our approach yields a better forecast performance in a reasonable computation time. The application of SVR ensembles for wind power prediction was published in:

- **Justin Heinermann** and Oliver Kramer, "Precise Wind Power Prediction with SVM Ensemble Regression", In Proceedings of the 24th International Conference on Artificial Neural Networks (ICANN). Lecture Notes in Computer Science, Springer, 2014.

Chapter 6: Combination of Speed and Power Time Series

In this chapter, we analyze various regressors trained with patterns composed of different features, i.e., power output measurements, wind speed measurements, and differences of these. The algorithms we compare are k-NN, SVR, and random forest (RF) that turned out to be very successful in various applications. Last, we combine the best combinations of regressors and the different features to ensembles and show experimentally that these outperform their single predictor competitors. The main results were published in:

- **Justin Heinermann** and Oliver Kramer, "Short-Term Wind Power Prediction with Combination of Speed and Power Time Series", in Proc. KI, 2015

Chapter 7: Heterogeneous Ensembles

In this chapter, we discuss the practical use of *heterogeneous regression ensembles* for the task of wind power forecasting, aiming at optimal regression accuracy as well as maintaining a reasonable computation time. In the first step we compare homogeneous ensemble predictors consisting of either DT, k-NN, or SVR as base algorithms. As diversity among the ensemble members is crucial for the accuracy of the ensemble, we propose the use of *heterogeneous ensemble predictors* consisting of different types of base predictors for wind power prediction. Our comprehensive experimental results show that a combination of DT and SVR yields better results than the analyzed homogeneous predictors while offering a decent runtime behavior. Going further, we show that heterogeneous ensemble predictors are very well-suited for using large numbers of neighboring turbines and past measurements and improve the prediction performance. Most parts of this chapter are based on:

- **Justin Heinermann** and Oliver Kramer, "On Heterogeneous Machine Learning Ensembles for Wind Power Prediction", in Proc. AAAI Workshops, 2015.
- **Justin Heinermann** and Oliver Kramer, "Machine learning ensembles for wind power prediction", in *Renewable Energy*, Volume 89, April 2016, Elsevier

Chapter 8: Evolutionary Multi-Objective Optimization

The success of machine learning models highly depends on the algorithm choice and the selection of optimal parameter settings, which are difficult tasks. These aspects also have a large impact on the runtime behavior. Unfortunately, we have to deal with an infinite number of possible combinations and choices for model selection. In Chapter 8, we show that ensemble models for wind power prediction can be optimized and selected using evolutionary multi-objective optimization algorithms in order to find a good trade-off between a short runtime and a low prediction error. The results of this chapter are accepted for publication in:

- **Justin Heinermann**, Jörg Lässig, and Oliver Kramer, "Evolutionary Multi-Objective Ensembles for Wind Power Prediction", in Proc. ECML Workshop DARE, 2016. In print.

Part IV: Summary & Outlook

Chapter 9: Summary and Outlook

In this chapter, we draw the conclusions and summarize the different research aspects and results of the thesis. Further, we present an outlook. Ideas for future research directions are presented, which are not in the scope of this thesis but could be reasonable extensions of our work. These comprise the handling of missing data, the computation of prediction intervals, the implementation of further ensemble techniques or deep learning, and the prediction of power ramp events using ensemble predictors.

Part II

Foundations

2

Machine Learning

In this chapter, we briefly introduce the machine learning methodology, which is an important foundation for our contribution. Machine learning (ML) is a class of statistical algorithms designed to gain knowledge from observed data, i.e., creating a mathematical model for predictions on new observations. The main goal is an automated, data-driven computation of models without the need of human decisions. These models are trained with observed facts of the real world and shall extract an idea of the underlying information from these data as best as possible. It is nowadays applied in a wide range of disciplines comprising biology, chemistry, computer graphics, medicine, automotive applications, security, and many others. A famous example application for machine learning is the automated recognition of handwritten digits, see Figure 2.1. Scanned grey-scale images of handwritten digits shall be recognized and a digit 0-9 is assigned. For the training of the model, a huge historical database of handwritten digits of different persons is used, whereas the examples to be recognized could be written by a completely different and before unknown person. Like a human with the ability to read the characters and numbers written by an unknown person, a main requirement for machine learning models is the ability of generalization.

For the recent success of machine learning in various applications, it was a necessity to have enough computational power at hand [2]. Only with the growth of computer processors' speed, main memory and storage size, dealing with larger

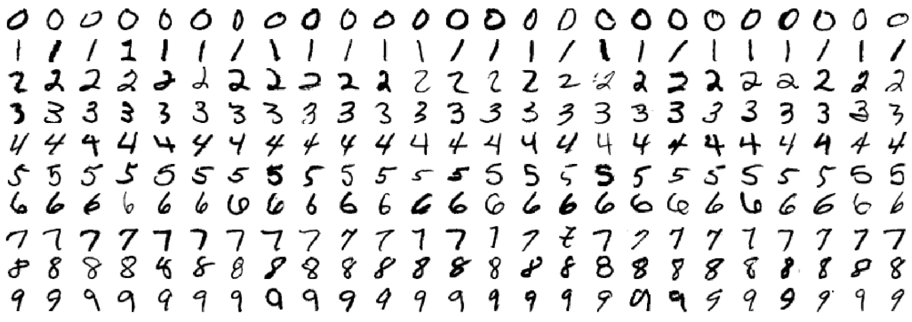


Fig. 2.1. Example for handwritten digits from the MNIST database [71]. The same digit written by different persons can substantially differ, which makes the classification difficult. A main requirement for a good classifier is the generalization ability.

datasets became possible. Further, machine learning models are able to handle problems which humans would not be capable of. A recent trend in computer science called *big data* deals with the challenge of tremendously growing quantity of data [127]. This is also true for the domain of energy and smart grids as the number of sensors is growing and should be processed automatically [20].

There exist plenty of different machine learning algorithms for different applications that differ in the generation of a model, accuracy, practical properties as well as computational cost [11, 39, 79]. We usually divide machine learning algorithms in three classes: *Supervised*, *unsupervised*, and *semi-supervised* learning. In the unsupervised setting, the objective is to discover the structure of a given dataset that consists of patterns without label information. In particular, *clustering* algorithms help finding groups of similar patterns. *Dimensionality reduction* methods give a mapping to a lower-dimensional space, i.e. patterns with less features than the given ones, while maintaining the best-possible reconstruction of the original patterns. Whereas semi-supervised methods are an interesting extension of supervised methods when only scarce data are available, we focus on the supervised setting in this work. Here, labels are available and can be used to compute models for prediction and inference.

This chapter is structured as follows. The idea of supervised learning and a formalization is given in Section 2.1. The k-nearest neighbors regression, presented

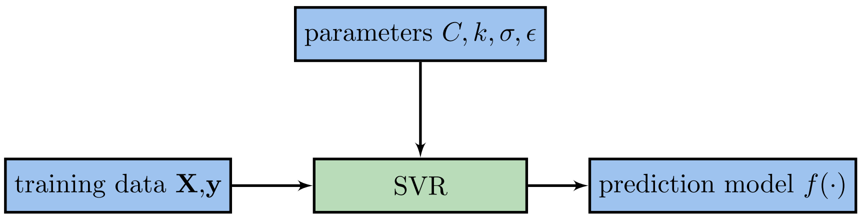

parameters C, k, σ, ϵ
training data $\mathbf{X}$,$\mathbf{y}$
SVR
prediction model $f(\cdot)$

Fig. 2.2. Example for the supervised machine learning setting. A model f is computed with an algorithm based on the labeled training dataset and a number of parameter settings. In this case, an SVR model is trained with parameters penalty C, Kernel k, kernel bandwidth σ, and loss sensitivity ϵ.

in Section 2.2 is a good example for a basic machine learning algorithm. Decision Trees are briefly described in Section 2.3. An important part of machine learning is the model selection and parameter tuning, which is depicted in Section 2.4.

2.1 Supervised Learning

In this work, we limit ourselves to the task of supervised learning. I.e., labels are assigned to patterns in the training set and the task is to predict a label for an unknown test pattern. With discrete labels, the task is called classification. With continuous values, the task is called regression. For the time series prediction task in this work, we deal with a regression problem an therefore put emphasis in this chapter on regression methods.

A *pattern* $\mathbf{x}_i \in \mathbb{R}^d$ is a vector of *features* $\mathbf{x}_i = (x_1, \dots, x_d)^T$ with $x_i \in \mathbb{R}$. A machine learning algorithm generates a statistical model based on a training set $\mathbf{X}$ of length N $\mathbf{X} = [\mathbf{x}_i]_{i=1}^N$. The columns of that matrix consist of patterns $\mathbf{x}_i$, i.e., $\mathbf{X} \in \mathbb{R}^{d \times N}$. In the context of supervised learning, each pattern $\mathbf{x}_i$ is mapped to a target value $y_i \in \mathbb{R}$ (regression) or respectively a label $y_i \in \mathbb{N}$ (classification). A model $f(\cdot)$ is trained on a set $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ such that we get a reasonable prediction $f(\mathbf{x}')$ for a new pattern $\mathbf{x}'$ without label. A common measure for the quality of the prediction model on a dataset is the mean squared error (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f(\mathbf{x}_i) - y_i)^2. \qquad (2.1)$$

The difference between the label y_i of a test instance and corresponding prediction $f(\mathbf{x}'_i)$ is squared in order to penalize larger differences more than smaller ones.

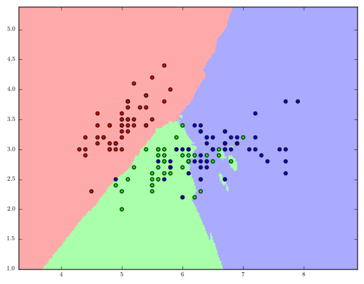
A recent overview and comparison of different available algorithms is given by Fernandez-Delgado *et al.* [28]. They evaluate 179 classifiers from 17 algorithm families. From the wide range of algorithms, some have its origins in statistics, some in symbolic artificial intelligence, and some are biologically inspired. It is often left to the expertise or taste of the practitioner, which algorithm is the best choice. Further it is highly dependant on the data which algorithm performs best. The 10 most influential algorithms in data mining[1] have been identified by the IEEE International Conference on Data Mining (ICDM) in December 2006 and comprise C4.5, k-Means, support vector machine (SVM), Apriori, EM, PageRank, AdaBoost, k-NN, Naive Bayes, and CART [123]. From a machine learning perspective, neural networks are a famous method as well, especially in the recent years through the idea of Deep Learning [7].

For this work, we mainly employ SVR, k-NN, and DTs. The k-NN and DT algorithms are described in this chapter as they help understanding machine learning methods. The regression variant of SVM is introduced as part of Chapter 5, dealing with ensembles of support vector regressors for wind power prediction. For the idea of machine learning ensembles, we refer to Chapter 4.

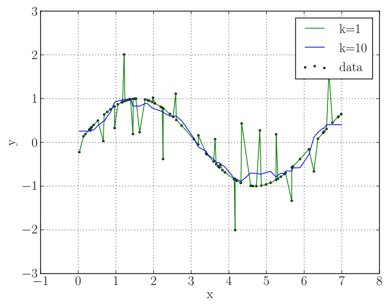
2.2 *k*-Nearest Neighbors

The popular *k-nearest neighbors* (k-NN) algorithm is a relatively simple but effective method for regression and classification problems, cf. [11, 39, 121]. It is well-known as one of the basic machine learning techniques and has been applied in many real-world scenarios like computer vision [98], astronomy [35, 44], computer graphics, biology, physics, and others.

[1] Data mining is usually based on machine learning algorithms [121]



(a) Classification with $k = 15$.



(b) Regression with $k = 1$ and $k = 10$.

Fig. 2.3. Example of classification and regression with k-nearest neighbors.

Nearest Neighbors Model

The k-NN model is based on the search of similar patterns in the training dataset w.r.t a distance metric. The most common choice is the Euclidean distance, which is defined as

$$\delta(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{i=1}^{d} (\mathbf{x}_i - \mathbf{x}'_i)^2} \tag{2.2}$$

for two patterns $\mathbf{x}$ and $\mathbf{x}'$. When using this metric, it is important to normalize the values of the features to a value between 0 and 1 – otherwise the features would have different importances depending on their scales of measurement, cf. [121]. On the other hand, one could use that behavior and define an adaptive distance metric, e.g. learning a general Mahalanobis distance, cf. [77, 120]. This research area is called distance metric learning.

When giving a prediction for a test pattern pattern $\mathbf{x}'$, the k nearest neighbors w.r.t. $\delta(\mathbf{x}, \mathbf{x}')$ are sought from all patterns of the training set $\mathbf{x} \in \mathbf{X}$. When dealing with regression tasks, the prediction model averages the label information of the k nearest neighbors via

$$f(\mathbf{x}) = \frac{1}{k} \sum_{i \in N_k(\mathbf{x})} y_i, \tag{2.3}$$

with the set of indices N_k of the nearest neighbors. For classification tasks, the predicted label is computed using majority voting. A distance-weighted average or voting is also possible and can yield good results, cf. [121].

For the success of the k-NN model for classification or regression, the choice of parameter k is very important. An example is shown in Fig. 2.3 (b). If k is chosen too small, the regressor is overfitted to the training set but cannot give good predictions for test patterns. The task of model selection is discussed in Section 2.4

Efficient Neighbor Search

For the k-NN model, no actual training phase is necessary, i.e. no learning is performed. The prediction for a pattern $\mathbf{x}'$ is solely based on the lookup in the training dataset. Therefore, the algorithm is a form of instance-based learning, cf. Witten [121]. While a naïve, brute-force implementation takes $\mathcal{O}(|\mathbf{X}| \cdot |\mathbf{X}_{test}|)$ time for a training set $\mathbf{X}$ and a test set $\mathbf{X}_{test}$, there are important improvements for a more efficient computation available. The creation of powerful spatial data structures is very common. The most important of them are k-d trees [8, 39] or *cover trees* [10]. A standard k-d tree is a balanced binary tree defined as follows: The root of the tree $\mathcal{T}$ corresponds to all points and its two children correspond to (almost) equal-sized subsets. Splitting the points into such subsets is performed in a level-wise manner, starting from the root (level $i = 0$). For each node v at level i, one resorts to the median in dimension $i \bmod d$ to partition the points of v into two subsets. The recursion stops as soon as a node v corresponds to a singleton or as soon as a user-defined recursion level is reached. Since it takes linear time to find a median, the construction of such a tree can be performed in $\mathcal{O}(n \log n)$ time for n patterns in $\mathbb{R}^d$ [8]. These trees offer logarithmic runtime for small dimensionalities $d \leq 15$.

Another acceleration strategy is *locality-sensitive hashing* [3, 98], which aims at computing $(1 + \varepsilon)$-approximations (with high probability). The latter type of schemes mostly addresses learning tasks in high-dimensional feature spaces. Furthermore, some implementations have been proposed for accelerating nearest neighbor queries by using a graphics processing unit (GPU). In most cases, such approaches aim at providing a decent speed-up for medium-sized datasets, but fail for large datasets [16, 34]. There exist also very efficient implementations employing data structures on the GPU but are not always applicable [35, 44].

2.3 Decision Trees

Decision Trees are basic machine learning tools for classification and regression. Besides moderate computational costs, the main advantage of decision trees is their model interpretability: A decision tree is usually a binary tree, where each node describes a decision criterion considering one particular feature of the test pattern. To each leaf node a label is assigned. Therefore, the machine learning practitioner can easily comprehend the decisions made when traversing the tree [11, 39]. There exist different algorithms for creating decision trees based on a training dataset. Quinlan developed the famous ID3 [90], C4.5 [91], and commercial successor C5.0[2]. In this work, we limit ourselves to the classification and regression trees (CART) algorithm from Breiman [14]. Both CART and C4.5 are amongst the top 10 algorithms in data mining identified by the IEEE International Conference on Data Mining (ICDM) in December 2006, cf. Wu *et al.* [123]

Decision trees are very simple and their capabilities are limited, but they are applied in a wide range of disciplines. However, decision trees are very important for this work and for recent machine learning in general because of their applications. The C5.0 algorithm makes use of boosting capabilities [30], whereas the CARTalgorithm found its extension in bagging [12] and the successful RF [13]. According to state-of-the-art research [28], these methods are amongst the most accurate and robust models.

[2] `http://rulequest.com/see5-info.html`

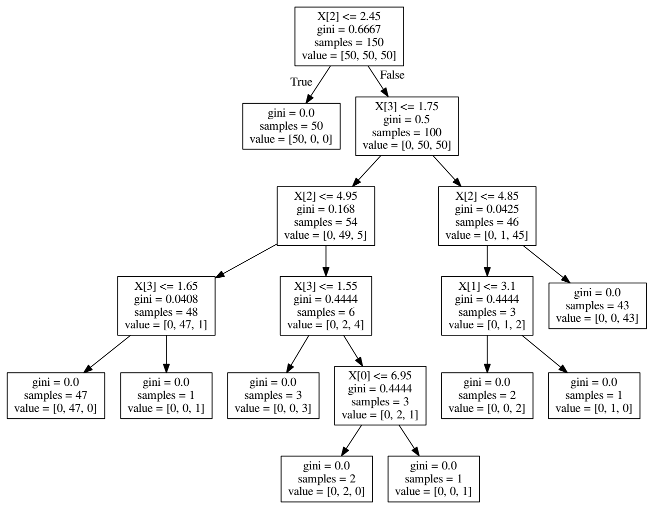


Fig. 2.4. Example decision tree for classification on the Iris dataset.

Decision Tree Construction

In the following, we use the notation from Louppe et al. [76]. The main concept of constructing a decision tree is to hierarchically partition the input space into reasonable regions. To classify a test pattern $\mathbf{x}'$ the tree is traversed to find the leaf containing $\mathbf{x}'$. Each tree node t contains a subset of the training dataset with N_t samples and implements a splitting criterion $s_t = (x_i < c)$ for this subset along one axis i. An example is visualized in Figure 2.4. E.g., from the root node, the left child is traversed if $x_2 < 2.45$ for the test pattern $\mathbf{x}'$, otherwise the right child node is visited. When constructing a tree from training set $\mathbf{X}$, the best split s_t for each tree node t, partitioning the samples from t into t_L and t_R, is found by maximizing the impurity decrease

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R) \tag{2.4}$$

with $p_L = N_{t_L}/N_t$ and $p_R = N_{t_R}/N_t$. Here, an impurity measure like the Shannon entropy (ID3, C4.5) or the Gini index (CART) is used to give a measure of the

Training
Validation
Test

Fig. 2.5. When enough training data are available, the dataset is commonly split into 50%, 25%, 25% sets for training, validation and testing.

information gain. The tree construction is done when each leaf nodes only contains training samples of the same class [14]. If decision trees become very complex, they tend to be overfitted to the training dataset, cf. Section 2.4. To avoid this, the tree built is pruned to a smaller depth in order to yield a better generalization performance. In the CART algorithm, pruning is carried out based on a cost-complexity model with cross-validation [123].

2.4 Model Selection and Parameter Tuning

When employing machine learning methods, a very important task is the appropriate model evaluation. I.e., one has to choose an algorithm and find the best possible parameters. To decide which model is the best to use, we need a measurable criterion for the quality of a model. For both algorithms and parameter settings, the list of possible choices is endless and therefore this task is not easy.

For computing the *training error* of a regression model f, the MSE of f on the training dataset is evaluated. While the training error would be a naïve choice to optimize, it is not a perfect measurement of model quality at all [11, 39, 50]. E.g., a 1-nearest neighbor model yields a training error of approx. 0 because each training pattern $\mathbf{x}$ has a distance to itself $\delta(\mathbf{x}, \mathbf{x}) = 0$. However, on unknown data the model can perform really bad due to overfitting, cf. Figure 2.3 (b). More important is the generalization performance: “prediction capability on independent test data.” [39] The objective of model selection is choosing the model with the lowest test error [50]. For this sake, we need sufficient test data.

Validation Set Method

In situations with enough data, the best approach [39] is dividing the training dataset randomly into three partitions: a training set, a validation set and a test set, see Figure 2.5. Each possible model is fitted to the training set and the prediction error for model selection is assessed on the validation set. With the test set, the generalization error of the chosen model is obtained by evaluating a prediction on the test set. Witten *et al.* [121] state that after model evaluation and testing, it is valid to train the chosen model again and now include also the validation and test data into the training dataset.

Bias-Variance-Tradeoff

When applying statistical learning methods, two important measures for the quality of a regression model f are the *bias* and the *variance* [11, 39, 58, 121]. These two properties are competitors and the machine learning practitioner has to find a trade-off between them. The expected test MSE for f over a large number of training sets for one test pattern $\mathbf{x}$ is defined as:

$$E[y - f(\mathbf{x})]^2 = Var(f(\mathbf{x})) + (Bias(f(\mathbf{x})))^2 + Var(\epsilon) \tag{2.5}$$

While the term $Var(\epsilon)$ is irreducible, a machine learning model that has both a low variance and a low bias is sought [50]. These depend on the complexity of the model. Consider a training and validation set. The behavior of the training and test error are visualized in Figure 2.6. A too simple model that cannot adapt well to the training data is going to have a large error because of bias. With increasing complexity of the model, the both training and validation error decrease first because of increasing variance, i.e., the ability of adapting the characteristics of the training data. The validation error, however only decreases to a certain point: With a too complex model, one is going to achieve a very good training error but the validation error may increase, essentially because of adaptation to noise in the training data. The use of a validation set or other cross-validation methods helps to find an optimal model while preventing overfitting [50, 58].

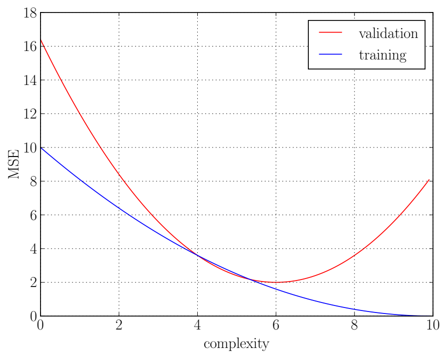


Fig. 2.6. Conceptional diagram of training and validation error depending on model complexity. At first, the validation error decreases along with the traniing error. Although the training error is decreasing further, the validation error then increases because of overfitting.

Cross-Validation

In real-world scenarios there is not always much data available and one has to learn and choose models with only insufficient data at hand. A simple and often used technique for giving a good estimation of a model's prediction error is the *cross-validation* (CV) method [39, 57, 103]. Figure 2.7 shows an example of a K-fold cross-validation with $K = 5$. The available data is randomly split into K equal-sized partitions $D_1 \dots D_K$. There are conducted runs $i \in \{1 \dots K\}$. In each run i, D_i is used as validation set on which a prediction error MSE_i is computed. The training set comprises all remaining partitions D_j with $i \neq j$. The cross-validation error is computed as the average of the prediction errors of all K runs:

$$CV\ error = \frac{1}{K} \sum_{i=1}^{K} MSE_i \tag{2.6}$$

The CV error is then used as an estimation for the expected test MSE. Leave-one-out cross-validation (LOOCV) is a special case of K-fold CV with $K = N$ folds [50]. Here, a single pattern is used as validation set in each run.

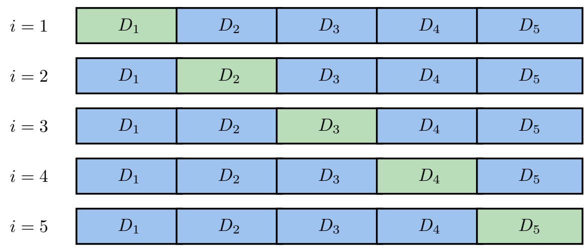


Fig. 2.7. K-fold cross-validation with $K = 5$ is a common choice for model selection with prevention of overfitting.

Therefore, LOOCV is computationally expensive and in practice a $K = 5$ or $K = 10$ is typically chosen.

Model Selection and in particular parameter choice are optimization problems. A typical approach for finding good parameters is grid search with manual choice of the ranges for each available parameter. Each combination of parameters is then evaluated with cross-validation. The grid search is sufficient for many use cases but can take very long time. More sophisticated methods are available for a faster search of an optimal model in the search space like random search [9], evolutionary computation [31], or Bayesian hyperparameter optimization [101]. The ensemble method, which is introduced in Chapter 4 offers another solution to the problem. Good generalization ability is given because of the ensemble architecture and therefore less tuning is required. By combining diverse estimators to an ensemble, a better behavior w.r.t the bias-variance-tradeoff can be achieved [15, 86]

2.5 Summary

In this chapter we introduced the machine learning methodology. Machine learning provides various algorithms for building successful data-driven models. As examples, the k-NN and DT are introduced. Besides SVR, they are the foundations of the developed methods in this thesis. Very important for the success of machine learning models is parameter tuning and model selection, which are difficult tasks. We discussed the bias-variance-tradeoff and described the cross-validation (CV) method as possible solution. Preferably, a model has good generalization abilities rather than overfitting to the training data. Objective is to give a good prediction for unknown data. In this work, we employ machine learning methods for creating a precise and robust prediction framework. We use machine learning ensembles for further improvements on accuracy and computation time.

3

Wind Power Prediction

In the recent years, a strong increase in wind energy can be observed. During 2015, 12,800 MW of wind power capacity was installed in the EU. There are now 142 GW of installed capacity, of which 131 GW are onshore and 11 GW offshore [106]. Germany has 45 GW installed capacity, which is the largest share in the EU. With growing capacities, there is also a growing demand of reliable and precise forecasts for various purposes. In this chapter, we describe the applications of wind power prediction and the state of the art methods, followed by introducing the spatio-temporal regression model used in this work.

3.1 Use Cases for Wind Power Prediction

Various use cases require good predictions for different forecast horizons. From a current time point t_0, the *forecast horizon* Δt denotes the time offset, for which the power or wind speed value shall be predicted in advance.

An important application for predictions is the trading of wind energy at the electricity markets, which is only possible with good forecasts because the energy is sold for future points in time. At the electricity markets, there is a trend towards shorter forecast horizons. For instance, the lead time has been reduced to 30 minutes on all intraday markets of the European Power Exchange

Table 3.1. Definition of different forecast horizons, cf. [29, 49, 102].

Horizon	Time Range	Applications
Very short-term	Few seconds – 30 minutes	Market clearing
		Trading
		Balancing
		Virtual Power Plants
Short-term	30 minutes – 6 hours	Load balancing
		Intraday trading
		Regulation
Medium-term	6 hours – 1 day	Day-Ahead trading
		Price optimization
Long-term	1 day – 1 week or more	Planning of reserve energy
		Scheduling of maintenance

(EPEX) Spot[1]. In the field of energy trading, there is demand for both interday and intraday forecasts, i.e., forecast horizons from minutes to days.

For grid balancing, precise forecasts are needed, too. The important task of keeping the voltage and frequency in the grid stable can only be managed when decisions concerning reserve power and distribution can be made early. The stakeholders are the grid operators as well as each energy supplier that feeds current to the grid. For the planning of balancing electricity, good forecasts are needed as well. In general, the balancing electricity or control energy is used to settle unexpected events in the grid. In Germany and the European Union, the control energy to be made available is traded via auction[2]. A trader therefore needs to know how much energy he can offer for a certain time point in the near future.

[1] `https://www.epexspot.com/en/press-media/press/details/press/EPEX_SPOT_and_ECC_successfully_reduce_lead_time_on_all_intraday_markets`

[2] For instance, the german control energy is traded via the online portal `http://www.regelleistung.net`.

In the future, the role of storage is going to be very important, too. When should batteries or other storages be filled and when is the stored energy used? The task of prediction is furthermore for the application of virtual power plants that comprise different types of power plants but act as one.

The different forecast horizons are classified in Table 3.1 and example applications are given. The methods that are currently available for these forecasts are presented in Section 3.2.

3.2 State of the Art

Prediction techniques can be divided in two groups, forecasting based on *numerical weather prediction* (NWP) on the one hand and predictions based on the historical time series on the other hand [22,29]. In both categories a wide variety of different techniques and hybrid approaches can be found. A review of methods for wind power forecasting is given by Foley *et al.* [29].

A review for forecasting of both wind speed and generated power was written by Lei *et al.* [73]. Soman *et al.* [102] give a comprehensive overview of techniques with emphasis on the use for different horizons. Wang *et al.* [49] provide a classification of various wind power forecasting methods. In the following, we give a brief introduction into the NWP methods in Section 3.2.1, followed by a more detailed view on statistical learning techniques in Section 3.2.2.

3.2.1 Numerical Weather Prediction

NWP models are based on physical computations describing the state of the atmosphere, including values of radiation, turbulence, and pressure. Besides the laws of physics, typically Navier-Stokes-Equations are employed, which are used to describe the motion of viscous liquids [29, 67]. Weather prediction can be addressed with global or regional models of different resolution. For dealing with larger resolution and better representations of the atmospheric processes, NWP

[3] `http://www.dwd.de/DE/forschung/wettervorhersage/num_modellierung/01_num_vorhersagemodelle/numerischevorhersagemodelle_node.html`

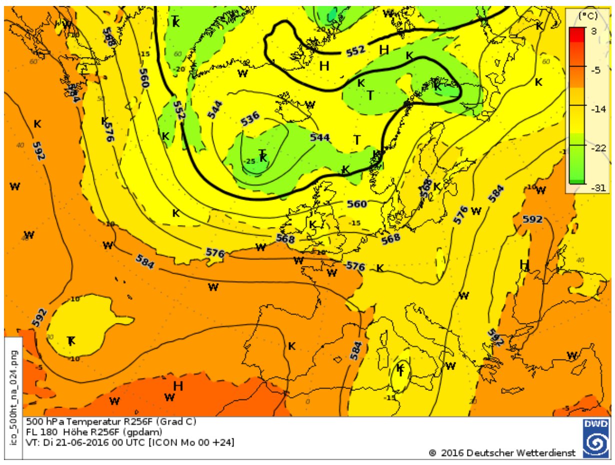

(°C)
3
-5
-14
-22
-31
ico_500ht_na_024.png
500 hPa Temperatur R256F (Grad C)
FL 180 Höhe R256F (gpdam)
VT: Di 21-06-2016 00 UTC [ICON Mo 00 +24]
© 2016 Deutscher Wetterdienst
DWD

Fig. 3.1. Example for NWP by the German Weather Service.[3]

models are usually computed using supercomputers at the weather services or in research institutes. The forecasts are not produced for one particular purpose, but rather for different use cases in industry and science [29]. The output of the model is not only wind speed but the state of the atmosphere for a given place and time, resulting in a coarse grid of forecasts. Based on the grid, with distance between the grid points of few kilometers, the wind speed at the location of the target turbine is computed [69]. The speed value is transformed into a power value by applying the power curve of the turbine, see Figure 3.2.

Although these models are widely applied, there are some drawbacks. The consumer of the forecasts is dependent on the weather services providing the forecasts. The time scales that can be used are always fixed and the forecasts are only available at a certain point of time. Because of the chaotic nature of the atmosphere, it is a very challenging task to give a good prediction with a physical model and therefore, for short forecast horizons other approaches like statistical learning yield superior results. Besides these approaches, computational fluid dynamics (CFD) gained attention in the recent past. Marti *et al.* [80] propose a model feasible for forecast horizons up to four hours. Castellani *et al.* [18]

[5] Found on the manufacturer's web page: `http://www.enercon.de/fileadmin/Redakteur/Medien-Portal/broschueren/pdf/en/ENERCON_Produkt_en_06_2015.pdf`

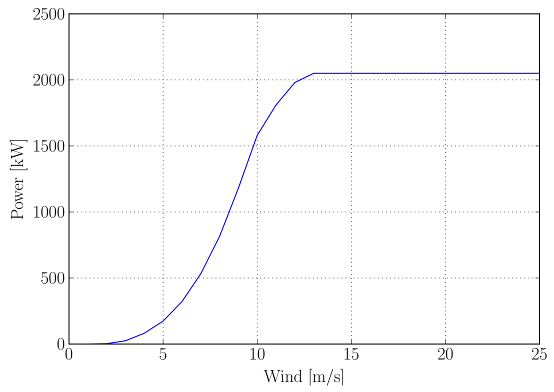


Fig. 3.2. The power curve of an Enercon E-82 wind turbine[5]. For each wind speed, the expected power is given.

investigate the hybridization of computational fluid dynamic (CFD) and artificial neural networks.

In the field of numerical weather forecasts, it is common to use ensemble models [84]. On the one hand, forecasts can be improved. On the other hand, not only a deterministic prediction is desired but rather an prediction interval with uncertainty information, which is obtained by probabilistic methods. Several NWP models are initialized with different values and combined to an ensemble forecast. In the field of NWP models, this approach is especially reasonable because of the difficulty to obtain the accurate values of the current atmospheric state. Further, small changes in the initial states yield large deviations in the forecast outcome. The forecasting using ensembles was one of the most important novelties of the recent years in the field of NWP. The weather services offer ensemble forecasts and their use is considered as state of the art [52, 107].

The statistical postprocessing of these ensemble forecasts is a major improvement. Gneiting *et al.* [36] found ensembles to reduce the prediction error by applying the ensemble model output statistics (EMOS) method to diverse weather forecasts. Similarly, Thorarinsdottir and Gneiting [107] are using a so-called heteroscedastic censored regression for maximum wind speed prediction over the American Pacific Northwest. Mahoney *et al.* [78] show that the combination of different NWP methods to an ensemble can be very effective. One of the models used is a 30-member NWP ensemble calibrated with an analog ensemble Kalman filter and Quantile Regression. Junk *et al.* [52] present an analysis of different weighting strategies for an analog ensemble. The uncertainty information is derived from a deterministic model and the prediction performance is improved by 20 %.

3.2.2 Statistical Learning

It has been shown that machine learning methods are well-suited to the domain of wind speed and wind power prediction [1,68,110]. Techniques like k-NN [65,96] or neural networks [92] have successfully been applied. Especially when spatio-temporal information is available machine learning models can yield feasible prediction performance.

In a recent comparison, Treiber *et al.* [110] show that support vector regression is superior to numerical weather predictions for shortest-term forecast horizons. Up to three hours, machine learning techniques are superior to post-processed meteorological models using forecasts of the German Weather Service (DWD). For more than six hours, meteorological methods should be preferred. Since short-term forecasts have important applications and the purely data-driven approach has many advantages, we aim at improving predictions for the short-term time scale further. In the future, a hybridization with meteorological methods could be beneficial for different forecast horizons. In the following sections, applications of the most important machine learning techniques are introduced.

Support Vector Techniques

The SVM technique is very successful for classification and regression tasks. In particular, good generalization abilities can be achieved. The method itself is described in Chapter 5 where it is used as foundation of our proposed method.

For short-term wind power prediction, Kramer and Gieseke [62] successfully applied the SVR algorithm. Here, a loss function parameter study is conducted and analyses of the prediction on both grid point and park levels suggest that SVR is very well-suited when ϵ-loss[6] is employed.

Mohandes *et al.* [83] employ support vector machines for wind speed prediction, too. The performance of the SVM prediction is compared to the multilayer perceptron (MLP) neural networks. The SVM model outperforms the MLP in most cases. However, the experiments are only conducted on mean daily wind speed data from Madina city, Saudi Arabia and give no insights for short-term prediction horizons.

Another SVR approach to short-term wind power forecasting is given by Zhang *et al.* [126]. They describe a framework based on grid search and a multi scale SVR. A comparison with an MLP demonstrates that the SVR approach is "robust, precise, and effective" [126]. Salcedo-Sanz *et al.* [96] employ SVR for the reconstruction of wind speed measurements from neighboring turbines. In the majority of cases, the SVR model works better than an MLP.

Artificial Neural Networks

The use of artificial neural networks in the field of wind power prediction has been studied, too. An example of wind speed prediction in the mountainous region of India using an artificial neural network model is given by Ramasamy *et al.* [92]. The wide range of applications of artificial neural networks in energy systems presented by Kalogirou [53] comprise various prediction tasks.

Upadhyay *et al.* [112] show that wind speed forecasting tasks can be handled by using back-propagation neural networks. Different variants of backpropagation and in particular Resilient Propagation are compared by Stubbemann *et al.* [104]. Their results suggest that the improved variants RPROP+ and iRPROP+ outperform the classical backpropagation methods.

Other technqiues

Besides the mentioned well-known techniques, there is a wide range of other methods that have been employed for wind prediction. Probabilistic short-term

[6] See Chapter 5.

wind power forecasting based on kernel density estimators has been proposed by Juban *et al.* [51]. Graff *et al.* present an approach to wind speed forecasting using genetic programming. Treiber and Kramer [109] use evolutionary computation for feature weighting in k-NN regression.

Besides machine learning techniques, autoregressive time series techniques like autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA) can be used, too [55, 99]. Although not directly connected to wind we consider the approach of Valdes [113] worth mentioning. Valdes shows that spatio-temporal information can be used well when employing ARIMA models, e.g. for brain manifolds. A comparison of ARIMA vs. neural networks for wind speed forecasting is given by Palomares *et al.* [87]

It has been shown that hybrid approaches perform particularly well for different forecasting tasks. Wang *et al.* [118] propose a self-adaptive hybrid approach for wind speed forecasting. Xiao *et al.* [124] show that combined forecasting models for wind energy forecasting outperform the single models in most cases. Hybridization also works well for machine learning predictions [19, 110, 125] and time series methods like ARIMA [75].

Closely related to hybrid models is the promising approach of machine learning ensembles, which are investigated in this thesis. The application of machine learning ensembles to the field of wind power prediction has also been shown to work well: Kusiak, Zhang and Song [68] successfully apply different methods to short-term wind power prediction, one of which is the bagging trees algorithm. Fugon *et al.* [32] compare various algorithms for wind power forecasting and show that random forests with and without random input selection yield a prediction performance similar to SVR, but recommend to prefer a linear model when the computation time grows too large. Another similar application of ensembles is given by Hassan, Khosravi, and Jaffar [38] for electricity demand forecasting. Here, neural network ensembles are applied. A key for the success is, again, the diversity amongst the predictors. For solar power output prediction, Chakraborty *et al.* [19] built up an ensemble of a weather forecast-driven Naïve Bayes Predictor as well as a kNN-based and a Motif-based machine learning

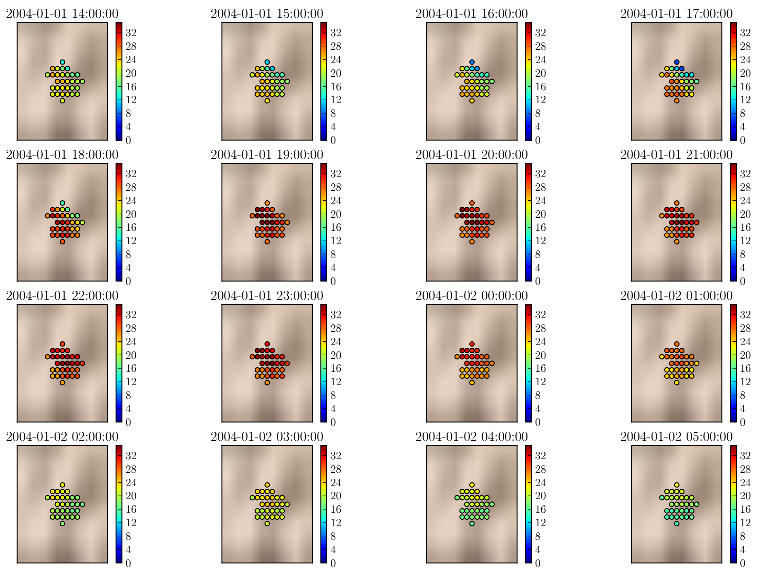


Fig. 3.3. Wind speed measurements for a wind park near Reno. The color denotes the wind speed, showing similar behavior for neighboring turbines.

predictor. The results of the three predictors are combined with a Bayesian Model Averaging. Chakraborty *et al.* show that the prediction error can be reduced by inducing ensemble methods to forecasting power output.

3.3 Spatio-Temporal Regression Model

In this work, we investigate the prospects of enhanced machine learning techniques for wind power prediction. Hence, we treat short-term wind power prediction as a regression problem. In contrast to numerical weather predictions, machine learning methods usually only make use of the time series data itself, i.e., power or speed measurements. A training dataset consists of historical measurements. When performing a forecast, the objective is to predict the measurement after a *forecast horizon* $\Delta t.$, e.g., in half an hour. The input patterns consist of μ past time steps, which is called the *feature window* throughout this thesis. In this work, we use a spatio-temporal model based on the one proposed by Kramer *et al.* [64], who show the benefit of involving neighboring turbines to the input vector. Figure 3.3 shows the wind speed measurements for a set of turbines near Reno. It can be seen that nearby turbines show similar speeds at the same time and it exists some correlation between the time series of them. If one wants to give a power output prediction for a certain turbine based on its past time series measurements used as patterns, including the measurements of turbines in the vicinity of a few kilometers into the patterns greatly helps to reduce the prediction error.

Algorithm 1 Defining a Wind Park

```
1: Inputs:
       Longitude and latitude of all available $N$ turbines,
       Target index $j$,
       Number of neighbor turbines $m$
2: Returns:
       Indices $\mathbf{M}$ of the neighbor turbines,
       Distances $\mathbf{D}$ of each turbine $i$ to the target turbine
3: for i = 1 to $N$ do
4:     $\mathbf{D}_i \leftarrow$ haversineDistance($i$, $j$)
5: end for
6: $\mathbf{M} \leftarrow$ nearestNeighbors($\mathbf{D}$, $j$, $m$)
```

In this work, a wind park is defined by a target turbine in the center and the m nearest neighboring turbines, as depicted in Algorithm 1. The distance is computed by the haversine distance using the longitude and latitude of the turbines.

Using the measurements of the parks turbines, the patterns are constructed as follows. Let $p_i(t)$ be the measurement of a turbine $\mathbf{M}_i$ at a time t, and $\mathbf{M}$ the m indices of the m neighboring turbines. Then, for a target turbine with index j we define a pattern-label-pair $(\mathbf{x}, y)$ for a given time t_0 as shown in Formula 3.1

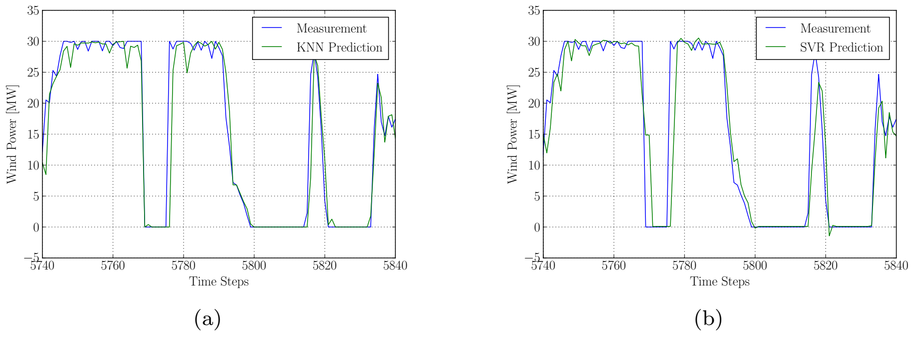


Fig. 3.4. Wind power prediction for a wind park near Casper: (a) k-NN regression, (b) SVR regression.

$$\begin{pmatrix} p_1(t_0-\mu) & p_1(t_0-\mu+1) & \dots & p_1(t_0-1) & p_1(t_0) \\ p_2(t_0-\mu) & p_2(t_0-\mu+1) & \dots & p_2(t_0-1) & p_2(t_0) \\ \dots & \dots & \dots & \dots & \dots \\ p_m(t_0-\mu) & p_m(t_0-\mu+1) & \dots & p_m(t_0-1) & p_m(t_0) \end{pmatrix} \rightarrow p_j(t_0+\Delta t) \qquad (3.1)$$

For a discussion of performance and efficiency of newly proposed prediction algorithms, we have to compare our results to the commonly used algorithms. In the following, we give an example of an application of the spatio-temporal model. For wind power prediction, k-NN and support vector regression (SVR) are considered as state of the art-methods. k-NN is a relatively fast method, but the prediction accuracy is usually outperformed by SVR, as can be seen in Table 3.2. We use the MSE as measure for the prediction accuracy. For both algorithms, we performed a 10-fold CV to find optimal parameters k for k-NN, and C,σ for SVR. Of Course, this process takes a huge amount of time, which makes this type of parameter search infeasible for practical use. Because the training and test time is depending on the parameters, after CV evaluation we use the resulting optimal parameters to train a model on the whole training dataset to measure training and test time as well as the test MSE. The NREL dataset has been used, see Appendix A. The data from 01/2004 until 06/2005 is used as training data set and the data from 7/2005 until 12/2006 serves as test data set, for both only using $\frac{1}{5}$ of the data. We are using a feature window $\mu = 6$ (1 hour) and providing a forecast for a horizon $\Delta t = 3$ (30 minutes).

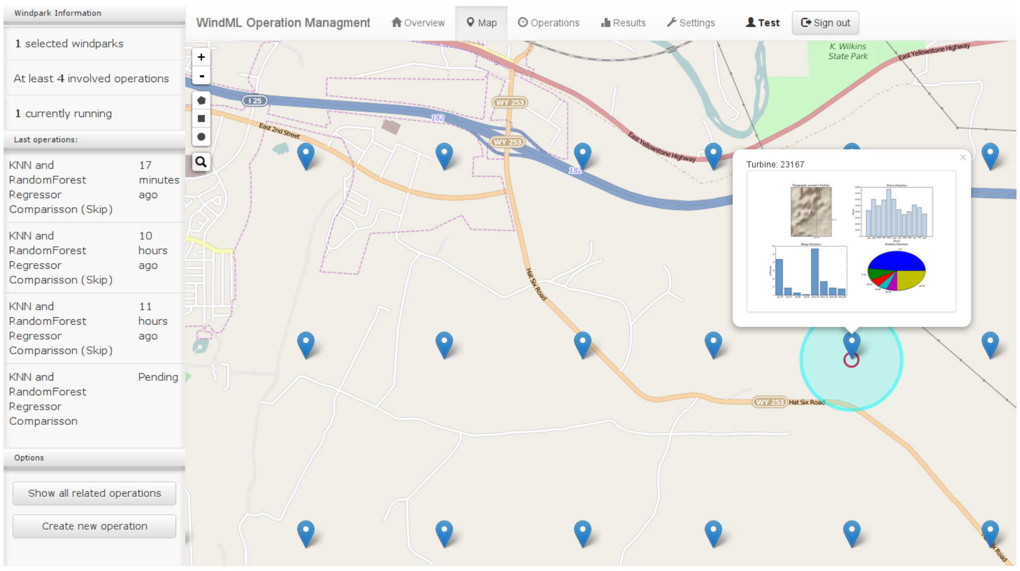
Table 3.2. MSE achieved by state of the art predictors with feature window $\mu = 3$ (30 minutes) and forecast horizon $\Delta t = 3$ (30 minutes).

Algorithm	k-NN			SVR		
Turbine	t_{train}	t_{test}	MSE	t_{train}	t_{test}	MSE
Casper	1	97	10.67	704	268	9.88
Las Vegas	1	118	10.46	1450	341	15.69
Hesperia	1	83	7.69	1218	261	7.39
Reno	2	98	14.81	1173	253	13.29
Vantage	2	105	6.86	601	251	6.54

3.4 WindML Framework

During the work on this thesis, the open source framework WindML [63] has been developed in the Computational Intelligence Group, University of Oldenburg. It is important for this work because it provides some basic functions required by the software implementations in this work. The WindML framework is released under the open source *BSD 3-Clause license.* The objective of the Python-based framework is easing the data-driven research in the wind power domain. Many important steps of the machine learning pipeline are provided.

First, a data server is provided. The loading mechanism automatically downloads requested data sets to a local cache, which are then stored in a local cache in the Numpy [114] binary file format. Data must only be downloaded once and recurrent loading is sped up. The interface for different data sets allows a generic integration of different data sources. In the current version 0.5, the NREL and



1 selected windparks

At least 4 involved operations

1 currently running

KNN and RandomForest Regressor Comparisson (Skip)	17 minutes ago
KNN and RandomForest Regressor Comparisson (Skip)	10 hours ago
KNN and RandomForest Regressor Comparisson (Skip)	11 hours ago
KNN and RandomForest Regressor Comparisson	Pending

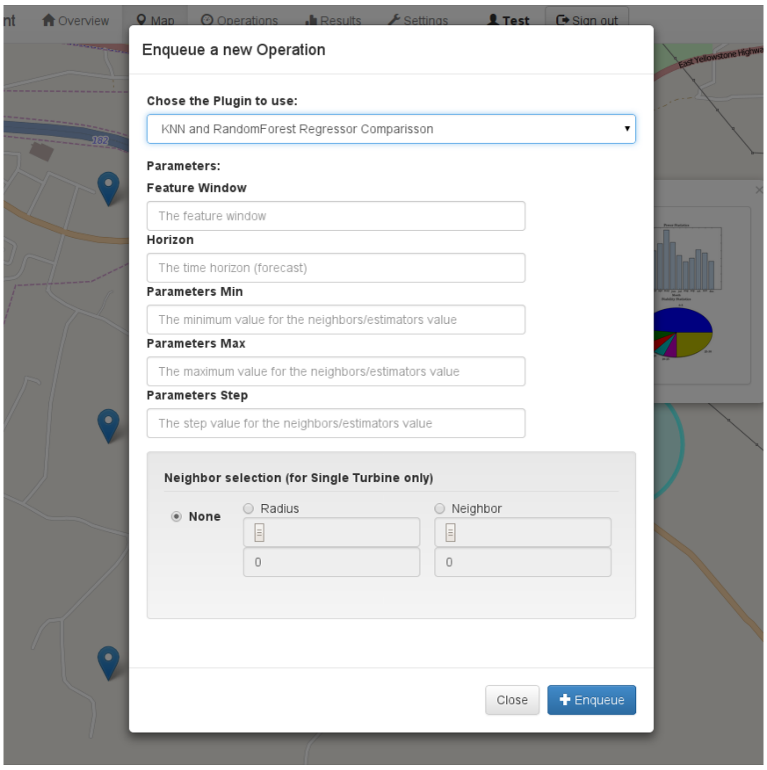
Fig. 3.5. Map View in the WindML Web Framework.

AEMO data sets are supported, see Appendix A. The documentation of the overall framework can be found on the WindML project website[7].

For the practical use of the time series data, the framework provides the generation of the spatio-temporal wind speed and power patterns and corresponding labels with an object-oriented architecture of wind parks, wind turbines, and the different measurements. Further, the problem of missing data can be handled with imputation methods. Besides several supervised and unsupervised machine learning models, essential helper functions are provided.

Based on a bachelor thesis, we developed a web frontend called WindML Web. Several practical requirements led to the development of the extensive experimental platform. For both research and industry applications, the machine practitioner needs a systematic and comfortable workbench for conducting experiments like parameter tuning, algorithmic comparisons, or simulations of operational management. In our case, the available datasets consist of hundreds of turbines and therefore a clearly arranged user interface is required. An important

[7] `http://www.windml.org`



Enqueue a new Operation

Chose the Plugin to use:

KNN and RandomForest Regressor Comparisson

Parameters:

Feature Window: The feature window

Horizon: The time horizon (forecast)

Parameters Min: The minimum value for the neighbors/estimators value

Parameters Max: The maximum value for the neighbors/estimators value

Parameters Step: The step value for the neighbors/estimators value

Neighbor selection (for Single Turbine only)

[✓ None] Radius 0 Neighbor 0

Fig. 3.6. Enqueuing an experiment in WindML Web.

part is a map view that shows all available turbines and allows for interaction with them. An example of the interactive map based on OpenStreetMap and the jQuery-Plugin Leaflet can be seen in Figure 3.5. As the machine learning experiments can easily take hours, it is desirable to run the processes on a server machine. In our case, the WindML Web application runs on an AMD Opteron 32-core system. One of the most important tools is a queue where the experiments can be appended – they are run when enough resources are free to use. The creation of an experiment is shown in Figure 3.6. The web application has a user management and a number of central processing unit (CPU) cores can be assigned to each user. Thus, a fair allocation of resources can be ensured.

For the particular use case of wind power prediction, a comparison of the available prediction techniques for each target turbine can be easily run and all results are persisted in a database (DB). Here, the document based database management system (DBMS) MongoDB is employed to provide for flexible storage of dynamic types of results. In particular, WindML Web can be easily extended with plugins and these can have different types of results. Although the web application is implemented in Java, all plugins are developed in Python and enhanced by a descriptive XML-File as interface.

3.5 Summary

Wind energy is a growing industrial sector and a clean source of renewable energies. For different use cases, there is a need for precise predictions at different forecast horizons. State-of-the-art are NWP and ML models, which are both active areas of research and subject to further improvements. In this work, our objective is to improve short-term predictions using ML methods. For this sake, a spatio-temporal regression model is employed which is based on the WindML framework.

4

Ensembles

A good alternative to the well-known machine learning algorithms is the use of ensemble models. By employing a number of diverse predictors and eventually combining their output values to a prediction, the accuracy of classification and regression can be improved while often reducing the required computation time [94]. In contrast to state-of-the-art machine learning algorithms, ensemble methods require less tuning and expert domain knowledge. Ensemble classifiers and regressors have been shown to work well in various applications like image recognition, medicine, network security, and others [86]

An example for ensemble regression applied to wind power prediction can be seen in Figure 4.1. Eight single regression models are trained using the DT algorithm. Each model on its own achieves only a poor prediction performance. Nevertheless, the average of the eight models predicts values very close to the real measurements. The intuition behind this seems very human: Before making an important decision, we ask several peers for their opinion.

A famous example for the intuition behind the ensemble paradigm can be found in the research of Sir Francis Galton, who describes the phenomenon called *vox populi* or wisdom of crowds [33]. Galton conducted a statistical experiment at a livestock fair. The objective for 800 people was to estimate the weight of a living ox. Most of the participants gave bad estimates, but the average of the estimates was very close to the real weight of the ox.

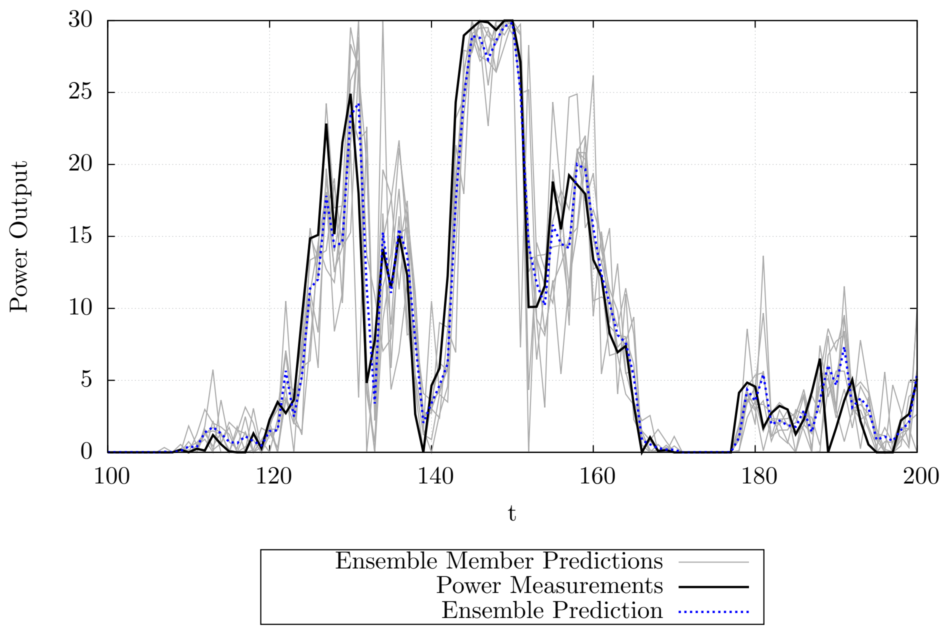


Fig. 4.1. Example for ensemble prediction of a wind power time series. While the eight single predictors do not provide a feasible prediction, their average gives a very good approximation to the real measurements.

This chapter is structured as follows. The ensemble methodology is introduced in 4.1. There are countless variants of different ensemble algorithms. One of the most important ones is the bagging algorithm, which is described in Section 4.2. A Section 4.3 describes the Random Subspaces method that helps to increase the diversity among the ensemble members. The boosting approach is described in Section 4.4, followed by stacking in Section 4.5 and the RF algorithm in Section 4.6. A summary is given in Section 4.7.

4.1 Ensemble methodology

As shown in Figure 4.2, an ensemble prediction is given by combining the predictions of T predictors f_i with $i \in \{1 \dots T\}$. A comprehensive overview and empirical analysis for ensemble classification is given by Bauer and Kohavi [5]. Another, more up-to-date review paper was written by Rokach [94]. The most important ensemble techniques are bagging and boosting. *bagging*, which stands for bootstrap aggregating, was introduced by Breiman [12] and is a relatively simple algorithm. The main idea is to build independent predictors using samples of the training set and average the output of these predictors. In contrast, *Boosting* approaches like AdaBoost [30], make use of predictors trained a consecutive manner with continuous adaptation of the ensemble weights. Rokach [94] classifies them as dependent models. Hastie [39] states the ensemble learning can be broken down to two tasks:

1. From the training data, a population of base learners is developed.
2. The base learners are combined to form the composite predictor.

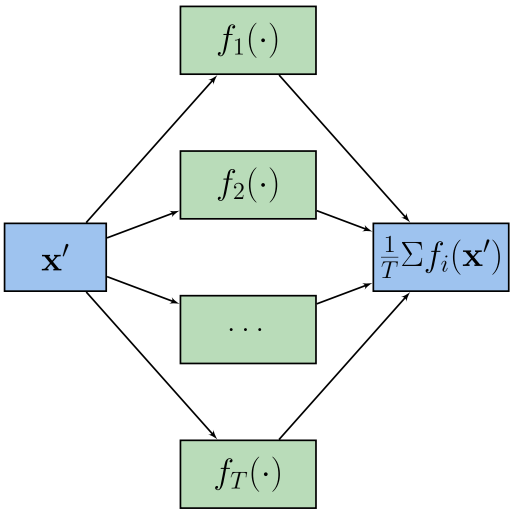


Fig. 4.2. Example of ensemble prediction with averaging of the member's output.

One key ingredient to successful building of ensembles is the concept of diversity. All the weak predictors should behave different or better uncorrelated [15,94]. A survey of diversity creation methods is given by Brown *et al.* [15]. There are many ways to generate such diversity, like manipulating the used training sample, the used features, and the weak predictors' parameters. Another possibility is the hybrid use of different learning algorithms, which we call heterogeneous ensembles. Oza and Tumer show that ensemble methods show a better behavior for the tradeoff between bias and variance [86]. Figure 4.3 shows an example architecture of training an ensemble.

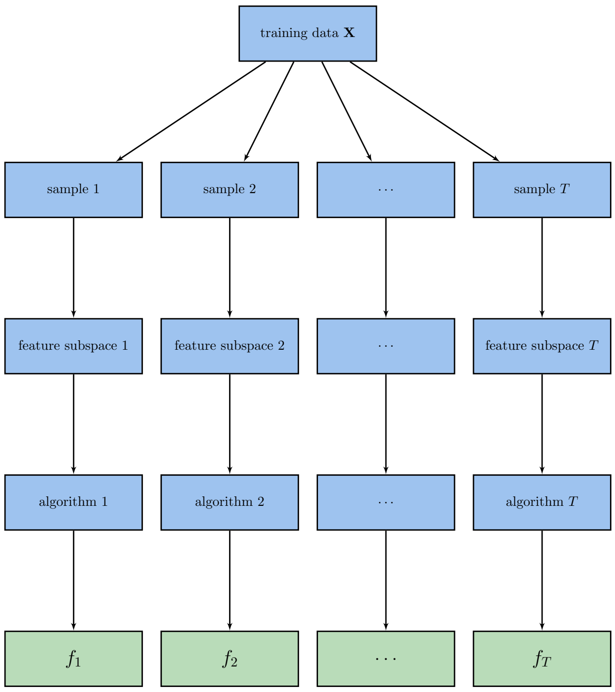


Fig. 4.3. Training of ensembles with sampling, feature subspaces, and different algorithms.

4.2 Bagging

An important and famous approach is bootstrap aggregating (bagging), which was introduced by Breiman [12]. The main idea is to build independent predictors using samples of the training set and average or voting of the outputs of these predictors. For regression, a prediction is given by

$$f(\mathbf{x}) = \frac{1}{T} \sum_{i=1}^{T} f_i(\mathbf{x}'). \tag{4.1}$$

Each model f_i is trained on bootstrap sample i. Breiman shows for both classification and regression, that bagging ensembles of DTs work much better compared to single trees. Furthermore, he gives arguments for the question why the bagging works. Because of the bootstrap method, each predictor has only a unique, limited point of view of the training data. The bootstrap is a resampling method from statistics. From one data set, random samples are chosen repeatedly with replacement. The samples give a good representation of the underlying distribution, although they are only drawn from the same dataset [39].

James *et al.* [50] give an illustrating example for the improvement of statistical learning models with the bootstrap. Consider the DT algorithm, and a training set split into two halves. On the data from each half, a DT model is trained. Because of the high variance of decision tree models, the two models could behave very different. In particular, the bootstrapping helps to reduce the models variance but also achieving a low bias. Bagging is very popular in combination with DTs, which are then grown deep and not pruned [50].

4.3 Random Feature Subspaces

Another method for providing diversity is the manipulation of the input features. As pointed out by Ho [46], bagging classifiers can benefit from choosing random feature subspaces, also called the random subspace method (RSM). Here, each model in the ensemble only takes a random subset of the available features into account. One has to decide how many features should be chosen. The choice can be done with or without replacement. Although replacement is sometimes considered as useful [24, 94], there is no explicit rule when to use it.

Scherbart *et al.* show that the results of bagging combined with RSM is comparable to RFs and ANNs [97]. An experimental study on the bias-variance-decomposition shows that RSM offers a good diversity among the estimators and good generalization abilities.

4.4 Boosting

Boosting approaches like AdaBoost [30] also combine T single predictors. The difference is that the predictors are not trained independently but in an iterative manner: Every training instance gets a weight assigned that is adapted in every iteration. In every iteration a new predictor is added to the ensemble and afterwards the prediction quality of the ensemble is tested on the training set instances. The wrong classified instances will now get higher weights than the correct classified ones. Therefore, the classification gets better with more iterations. In many cases, AdaBoost and other boosting approaches yield better prediction quality than bagging algorithms. On the other hand, boosting algorithms tend to overfitting problems [5,94]. Also, AdaBoost is not always better than bagging [24]. For us, a serious drawback is the fact that an adaptive boosting algorithm cannot train the weak predictors in parallel but step-by-step.

4.5 Stacking

Stacked generalization (Stacking) offers another concept of combining the ensemble member's outputs [39,86,122]. Instead of only averaging the output values or weighting them, another model is trained on the set of single predictions, see Figure 4.4. The top layer model computes the final output with a linear regression or any machine learning algorithm. It the objective to take also into account the behavior of the predictors. It is also trained on the training set and the errors of the single models can be evaluated and then be corrected in order to give a better prediction. For this work, we decided not to use stacking because it is more complicated than the other approaches and success is not guaranteed. The computational cost is also very high because both the ensemble members and the regression in the final layer have to be optimized. Since the approaches presented in this thesis work very well, we leave the research on stacking in the field of wind power prediction to future work, see Chapter 9.

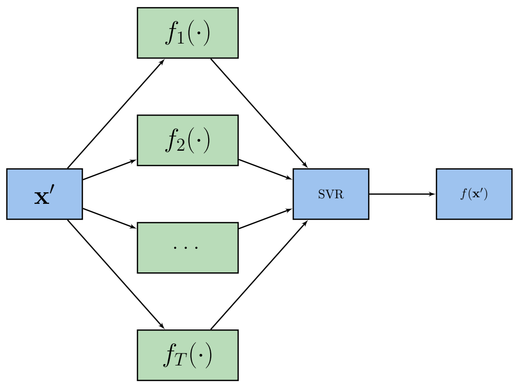


Fig. 4.4. Example of stacking with an SVR meta-learning model.

4.6 Random Forest Regression

A popular ensemble approach is the RF algorithm [13] that employs a large number of unpruned decision trees [94]. As with bagging, each decision tree is built with a subset sample from the training set, but only uses N of the available features of the patterns. One important factor for the success of ensembles is the concept of diversity. All the weak predictors should behave different if not uncorrelated to improve the prediction performance of the ensemble [15, 94]. In the random forest, the single models are decorrelated [50]. For selecting the splitting plane, only a random subset of m features is considered. Therefore, each model behaves different.

When using Random Forests, on can compute the importance of a feature x_i for predicting the correct label using the *Mean Decrease Gini*. For each tree in the ensemble, a weighted sum of the impurity decreases of all nodes t using x_i in split s_t is computed. For all N_T trees in the ensemble, an average of these sums is calculated:

$$Imp(x_i) = \frac{1}{N_T} \sum_{T} \sum_{t \in T : v(s_t) = x_i} p(t) \Delta i(s_t, t) \tag{4.2}$$

with node weight $p(t) = \frac{N_t}{N}$ and $v(s_t)$ the feature used in split t, cf. [13, 76].

4.7 Summary

When dealing with difficult problems in the field of machine learning, the use of ensemble models turns out to be an excellent alternative to the well-known machine learning algorithms. We combine a number of basic models in order to obtain an improved prediction model. The prediction accuracy for many applications of classification and regression can by improved. By employing ensemble models, one can also improve the computation time requirements. It is very practicable to employ ensemble models, because less tuning is required. In this thesis, we decided to stick to the independent methods bagging and RSM. The decision is based in the fact that these models are more straightforward and are easily parallelized on multicore architectures. In Chapter 5, we investigate a bagging approach with error-based weighting employing SVR models. Chapter 6 is using RF and SVR models with different types of features. Based on that, the RF and SVR models are combined to ensembles in various ways. The use of different inducers is further investigated in Chapter 7 and turns out to yield excellent predictions. Chapter 8 deals with the balancing of ensemble predictors with evolutionary computing.

Part III

Ensembles for Wind Power Prediction

5 Support Vector Regression Ensembles

It has been shown that good forecast results can be achieved using SVR [108]. The main problem of the SVR algorithm is the huge computational cost [25, 39]. In particular, when doing parameter studies and evaluating the prediction performance on large data sets, the optimization process becomes impractically slow. In order to achieve an acceptable forecast quality, training a SVR with CV methods can take hours or days. For practical applications, often a suboptimal prediction error has to be accepted. For both the prediction error and the efficiency, improvements are desirable.

In this chapter, we propose a novel SVR ensemble method for wind power predictions in order to further improve the forecast quality and spend less computation time. Instead of using a single support vector regressor, we train a number of regressors, hence called *weak predictors*, which together form an ensemble. Each of the weak predictors is trained on a small subset of the training set. The prediction is computed by a weighted average of the regression results of the weak predictors. An important motivation is the fact that SVR predictors perform very well in many use cases and in particular wind power prediction. This leads to the question if SVR are well-suited for ensemble building. In the field of ensembles, mainly decision trees are employed and the SVR algorithm has not been investigated extensively.

This chapter is structured as follows. Our ensemble method is described in Section 5.3. The experimental results, presented in Section 5.5, show that a random parameter choice and a MSE based weighting renders the best computation time as well as prediction performance. Compared to state-of-the-art algorithms, our approach yields a better forecast performance in a reasonable computation time. Our conclusions and future work can be found in Section 5.7.

5.1 Related Work

Kim *et al.* [56] build up classifiers using SVM ensembles using both bagging and boosting. The single SVMs are then aggregated either by using majority voting, an LSE-based weighting, or combined in a hierarchical manner. Tested data sets are the IRIS data set, the UCI hand-written digit data set, and a fraud detection database. The ensemble classifiers clearly outperform the single SVM predictor. An ensemble constructed with the boosting algorithm performs slightly better than the one constructed using bagging. According to Kim *et al.*, the most important thing in SVM ensemble constructing is that the single SVMS become as different as possible [56].

The latter aspect was also investigated further by Tsang, Kocsor and Kwok in [111]. They proposed orthogonality constraints for the weak SVM predictors in order to diversify them. The result is a better classification accuracy as well as a reduced training time.

Waegeman and Boullart [116] are using weighted support vector machines for ordinal regression. They show that SVM ensemble classification is well-suited for the problem of ranking.

A different approach that reminds of ensemble prediction is the Cascade SVM algorithm of Graf *et al.* [37]. The training set is split up into n small sets and the optimal support vectors for each subset are computed. Subsequently, a multi-stage approach merges and optimizes the n SVMs. The approach can be parallelized pretty easily. The authors show that the Cascade SVM algorithm is almost as precise as traditional SVMs while greatly reducing the computation

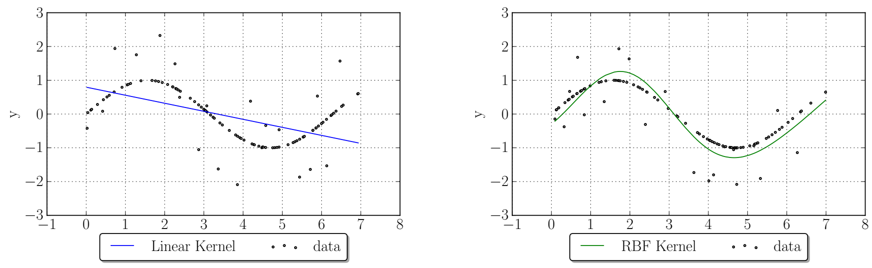


Fig. 5.1. Support Vector Regression with linear and RBF kernel.

time. In contrast to our approach, their main goal is to train the SVM in a shorter time wile achieving similar results. Our approach ha the objective of an improved prediction.

5.2 Support Vector Regression

The SVR algorithm often provides very good prediction results and is regarded as state-of-the-art regression technique. In general, the support vector machine (SVM) algorithm maximizes a geometric margin between the instances of the classes to be separated. Similar to SVM classification, the SVR algorithm aims at finding a prediction function $\hat{f} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$ that computes an accurate prediction value for an unseen pattern $\mathbf{x} \in \mathbb{R}^d$.

The SVR algorithm is based on SVMs that were proposed by Cortes and Vapnik [21] in 1995. For the training of the regressor, we aim at finding weights w by minimizing the following problem, formulated by Vapnik with an ϵ-sensitive loss function:

$$minimize\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{5.1}$$

$$subject\ to \begin{cases} y_i - \langle w, x_i\rangle - b & \leq \epsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i & \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq 0 \end{cases}$$

In this equation, $C > 0$ is a constant chosen by the user that is used as a parameter that penalizes only those errors which are greater than ϵ. Ideally, one would like to generate models that represent the training data well and at the same time are not too complex to avoid overfitting. The parameter C is called regularization parameter and determines the trade-off between these two objectives. The so-called slack variables ξ_i^* are introduced to cope with optimization constraints which would be otherwise infeasible [100].

Kernel functions are used to give good results on non-linear separable data. A kernel function can be seen as a similarity measure between patterns and is especially useful for non-linear regression tasks. A typical choice is the RBF-kernel, which is also used in this chapter. It makes use of a radial basis function:

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right) \tag{5.2}$$

An example for a SVR with a linear and a RBF-kernel is shown in Figure 5.1. While the linear kernel only gives a very rough approximation, the RBF-kernel is usually a better choice for complex data.

5.3 SVR Ensemble With Weighted Bagging

Our research objective is to answer the question, if SVR ensembles can be used to improve the prediction accuracy in the field of wind power forecasts. The second objective is to reduce the required computation times. The basic idea of the resulting training algorithm is depicted in Algorithm 2. Let $P = \{p_i | i = 1, \ldots, T\}$ be the set of predictors and $W = \{w_i \in \mathbb{R} | i = 1, \ldots, T\}$ the set of the corresponding weights. Each weight belongs to the predictor with the same index. The final prediction value is the weighted average of the estimators $p_i \in P$ using the weights $w_i \in W$.

$$f(\mathbf{x}) = \Sigma_{i=1}^{T} w_i \cdot p_i(\mathbf{x}) \tag{5.3}$$

As the design goal is to find a balance between a good regression performance and a feasible computational cost, we decided to implement a relatively simple bagging approach, which can easily be parallelized. Each iteration of the `for`-loop in line 4 is independent from its preceding run, so the loop can be replaced by a `map`-Operation and then executed on distributed computing systems or on multicore processors. For computing the prediction results, a similar parallelization can be done. In contrast to our bagging approach, iterative algorithms like AdaBoost are too expensive because of interdependent steps of the training algorithm.

Algorithm 2 Training of the SVR Ensemble Predictor

```
1:  Inputs:
        $\mathbf{X} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathbb{R}$ (training set),
        $S$ (sample size),
        $T$ (number of estimators)
2:  Returns:
        $P = \{p_i | i = 1, \ldots, T\}$ (estimators),
        $W = \{w_i \in \mathbb{R} | i = 1, \ldots, T\}$ (ensemble weights)
3:  Initialize:
        $w_i \leftarrow 1$, $i = 1, \ldots, T$
4:  for i = 1 to T do
5:      $\mathbf{X}_i \leftarrow sample(\mathbf{X}, S)$
6:      $\mathbf{X}_{val,i} \leftarrow \mathbf{X} - \mathbf{X}_i$
7:      $C, \sigma = ChooseParameters(SVR, \mathbf{X}_i)$
8:      $p_i = FitSVR(\mathbf{X}_i, C, \sigma)$
9:      $w_i = 1/RegressionPerformance(p_i, \mathbf{X}_{val,i})$
10: end for
```

5.3.1 Training the weak predictors

When training the estimators, one must decide which samples are used for the design of the training set and which settings are the best for the particular machine learning algorithm. Like Kim *et al.* [56], we build training sets $\mathbf{X}_i$ of the estimators by randomly sampling s instances from the global training set $\mathbf{X}$. Hence, the single sets $\mathbf{X}_i$ are non-disjoint and one single training instance $(\mathbf{x}_i, y_i)$ can occur multiple times or not at all. In future work, one could also test disjoint sets or even introduce orthogonality constraints like Tsang, Kocsor, and

Kwok [111], but randomly chosen sampling turned out to be a good choice and yields a reasonable improvement.

An important research aspect is the parameter tuning, which is implemented in the `ChooseParameters` method. In our case, we considered the regularization parameter C and the radial basis function (RBF) kernel bandwidth σ the most important ones and only varied these two parameters. We tested three different variants of parameter choices:

Global Optimization Each estimator's regression performance is given by the mean square error on the whole $\mathbf{X}_{val,i}$ validation set, which is the global training data set $\mathbf{X}$ without the training data sample $\mathbf{X}_i$. Thus, all estimators are optimized for the training data set via grid search.

Local Optimization The estimator's regression performance is optimized by a grid search using a cross-validation on the belonging training data sample $\mathbf{X}_i$.

Random Choice The parameters are randomly chosen. No optimization will be performed. An increase in diversity can be expected.

Apparently, both the global and local optimization are inherently expensive, because of the need to execute the SVR training algorithm multiple times. Also when using other optimization techniques like evolutionary algorithms, the SVR training has to be called multiple times, which results in a long computation time. Thus, one would prefer the random method if the prediction quality is not worse than the optimized methods. The experiments in the following sections show that a random choice of parameters can result in a better prediction performance.

5.3.2 Weighted Ensemble Prediction

We tested different methods of combining the estimators to an output are possible. The naïve choice is a uniform weighting with $w_i = 1$. Further, we tested weighting the predictors using a prediction error E on a validation data set $\mathbf{X}_{val,i}$. The error can be interpreted as importance of each estimator p_i in the ensemble:

$$w_i = \frac{1}{E(p_i, \mathbf{X}_{val,i})} \tag{5.4}$$

For E we tested the MSE, the square of the MSE, the inverse of the least square error (LSE), or the biggest square error (BSE).

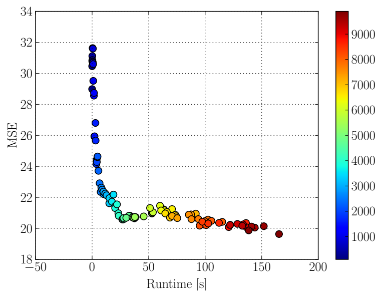
Besides the SVR and kernel parameters, and the ensemble weights, the two most important factors of the algorithm's success are the sample size S and the number of estimators T. As shown in Section 5.5, both variables should be tuned in order to achieve the best result.

5.4 Runtime

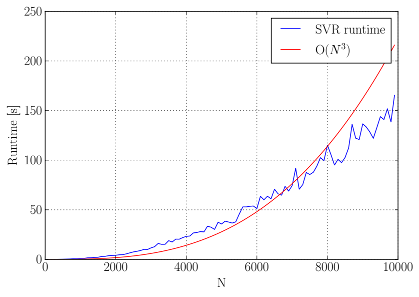
The SVR algorithm offers a runtime complexity of $\mathcal{O}(N^3)$ depending mainly on the number N of training data. While a small dataset can be processed very fast, with a growing N there is a rapid growth in the computing time needed. The reason for this lies in the optimization process required for finding the correct support vectors. In Figure 5.2, we show a visualization of the runtime behavior of SVR for two wind parks from the NREL dataset, see Appendix A. For the examples of Reno and Tehachapi and the years 2004-2006, we employed SVR with $C = 1000.0$ and $\sigma = 0.001$. The algorithm was only trained on a training subsets of size $N = \{100, 200, \ldots, 10.000\}$ taken from the first one and a half years. An Intel Core i3 with 1.90GHz was used to execute the training. The resulting predictors have been tested on the last one and a half years and the MSE has been computed. For both parks, we visualized in a scatter plot the behavior of runtime and MSE depending on the training set size N. Further, we show the runtime behavior depending on N. It can be seen that the runtime behavior is in the complexity class of $\mathcal{O}(N^3)$.

It can be seen that it is much cheaper to train a SVR predictors on a small training data set. Therefore, it is faster to train a large number of SVRs on small training set than one single SVR predictor on a large training data set. For the case of splitting of the training data set into partitions of size $n < N$, the runtime bound of our approach is given by:

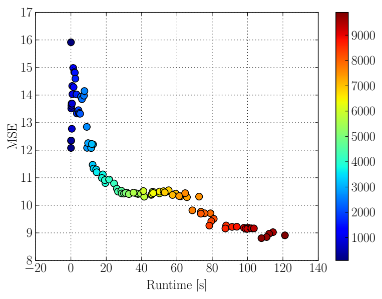
$$\frac{N}{n} \cdot n^3 = N \cdot n^2 < N^3 \text{ for } n < N \tag{5.5}$$



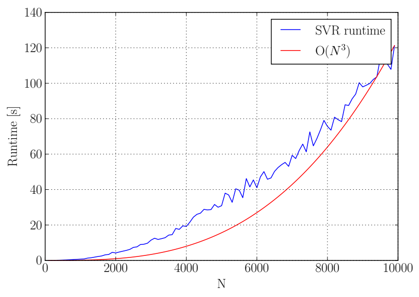
(a) Runtime and error behavior of SVR for a Wind Park near Reno with different training set sizes



(b) Runtime behavior of SVR and theoretical complexity for a Wind Park near Reno depending on training set size.



(c) Runtime and error behavior of SVR for a Wind Park near Tehachapi with different training set sizes



(d) Runtime behavior of SVR and theoretical complexity for a Wind Park near Tehachapi depending on training set size.

Fig. 5.2. The plots show the runtime behavior of support vector regressors with different training set sizes. On the left side (a, c), one can see that the MSE decreases rapidly first with a larger N and therefore larger runtime. After some point, a larger N only decreases the MSE sligthly. The runtime complexity of the SVR algorithm is in $\mathcal{O}(N^3)$. The cubic behavior depending on N is shown in the right plots (b,d).

In our case, we do not necessarily divide the whole training data set in partitions but rather sample T subsets of the size S. Therefore, our runtime and space complexity does not longer depend on the original training set size and the runtime boundary is given by

$$\mathcal{O}(T \cdot S^3) = T \cdot \mathcal{O}(S^3). \tag{5.6}$$

5.5 Experimental Results

In our experiments, we analyze the prediction performance of our ensemble regression approach. First we find the settings that achieve the best prediction performance and the lowest computation time needed. Given the best results that can be reached with our approach, we can compare the algorithm to the commonly used SVR. We used the power output data of five NREL wind parks that consist of the wind turbine the power output shall be predicted for, and the turbines in a radius of 3 kilometers. The feature window and horizon are $\mu = \lambda = 3$. As training dataset, the whole time series for the year 2004 is used and the data of the year 2005 serves as test data set. The experiments were run on an Intel Core i5 ($4 \times 3.10 GHz$) with $8 GiB$ of RAM.

5.5.1 Optimization and Weighting of the Weak Predictors

In our first experiment, we analyze the use of the different parameter optimization variants for the estimators. For C, the possible values used are $\{10^e | e \in \{0, \ldots, 4\}\}$ and σ is taken from $\{1 \cdot 10^{-e} | e \in \{0, \ldots, 4\}\}$. Furthermore, we compare the different weighting methods for each of the three algorithm variants. The results are presented in Table 5.1. For five wind parks, the MSEis compared for the three parameter choice methods. For the weights, least or biggest squared error based weights are not listed because of poor prediction performance. The results show that a random choice of the estimators parameters is better in the most cases while providing a much shorter runtime compared to the optimized variants. This behavior may be surprising at first, but complies with the intuition behind diversification [111] and random forests [12]. Another consideration is the possible overfitting when using the optimized variants. Therefore, we are using the computational cheap random variant with $\frac{1}{MSE^2}$ weighting in the following.

5.5.2 Number of Weak Predictors and Samples

The expected prediction error depends on the number of estimators. Furthermore, it is non-deterministic and we have to analyze the properties of our algorithm for

Table 5.1. Comparison of parameter choice methods and ensemble weighting methods ($T = 32, S = 1,000$). For the locally optimized, globally optimized, and random chosen parameters, the mean squared prediction error is evaluated for five turbines. For the weighting of the ensemble members, 1, $\frac{1}{MSE}$ and $\frac{1}{MSE^2}$ are tested. Furthermore, the runtime for the training process in seconds is given. The least error reached for each turbine is printed in **bold**, the least runtime is printed in *italic*.

(a) Local: Optimization of each predictor

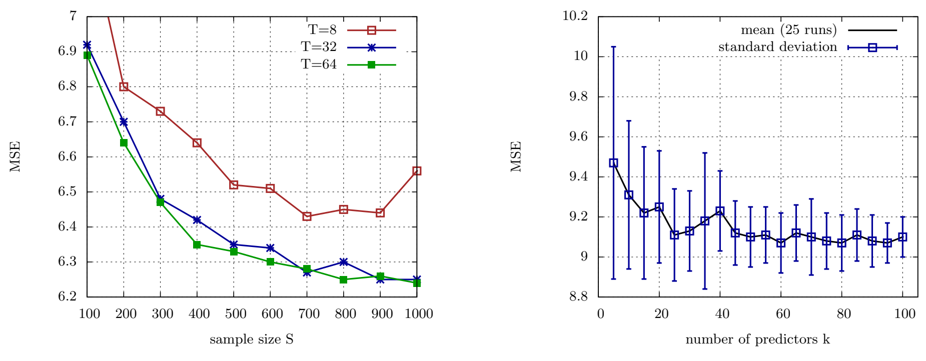
Value	Error			Time
Weights	1	M^{-1}	M^{-2}	
Cheyenne	7.84	7.84	7.84	175.93s
Lancaster	8.89	8.89	8.89	161.37s
Palm Springs	6.12	6.12	6.11	221.60s
Vantage	**5.63**	5.63	5.63	151.42s
Yucca Valley	10.29	10.29	10.29	226.89s

(b) Global: Optimization of combined predictor

Value	Error			Time
Weights	1	M^{-1}	M^{-2}	
Cheyenne	7.87	7.87	7.86	607.56s
Lancaster	9.04	9.04	9.03	513.66s
Palm Springs	6.13	6.12	6.12	476.66s
Vantage	**5.63**	5.67	5.67	525.88s
Yucca Valley	10.44	10.43	10.43	614.51s

(c) Random parameter choice

Value	Error			Time
Weights	1	M^{-1}	M^{-2}	
Cheyenne	12.38	8.14	**7.69**	*45.77s*
Lancaster	13.85	9.52	**8.81**	*36.80s*
Palm Springs	7.75	6.04	**5.96**	*41.68s*
Vantage	8.41	6.19	5.75	*37.68s*
Yucca Valley	10.59	10.10	**10.05**	*51.07s*



(a) (b)

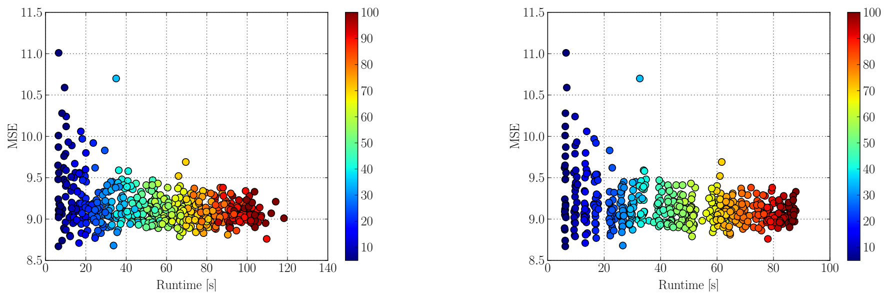
Fig. 5.3. (a) Prediction error for a wind park near Palm Springs depending on S, (b) Prediction error and standard deviation for a wind park near Lancaster depending on T.

varying parameters. Figure 5.3 (a) shows the behavior of our algorithm depending on the sample size S for a wind park near Palm Springs using $T \in \{8, 32, 64\}$ estimators: When increasing the number S of samples used for each estimator, the prediction error decreases. The standard deviation also decreases: E.g., for the case $T = 32$, the standard deviation when using predictors with sample size $S = 100$ is 0.30, and is reduced to 0.07 when using $S = 1,000$. Every prediction error is given by a mean of 25 repeated measures.

For the other parks, our approach yields the same behavior. Figure 5.3 (b) shows the dependency of the prediction error on the number T of estimators used. The measurements were repeated 25 times. The results show that a larger number T greatly decreases the prediction error and the standard deviation is reduced, too.

E.g., for $T = 5$, the standard deviation is 0.58, it decreases to 0.10 for $T = 100$. Thus, one can only expect reliable results with a sufficient T and sufficient S.

Figure 5.4 shows another visualization of the experiment. Here, each point depicts an instance of the ensemble model with $S = 2000$ samples and a varying number T of predictors which is denoted by the color. On the x-axis, the training time (a) or the testing time (b) is shown. The position on the y-axis shows the test MSE. One can see that a larger T in general leads to a smaller test error.



(a) Training time (b) Test time

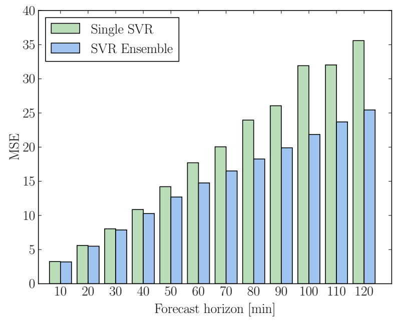
Fig. 5.4. The scatter plot shows the behavior of different ensemble predictors for a wind park near Lancaster. The color denotes the number T of estimators used.

Also, the closeness of points of one color shows that the deviation is smaller and the ensemble model can be expected to be more reliable.

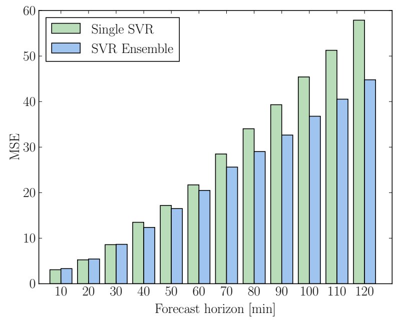
5.5.3 Comparison to Support Vector Regression

Table 5.2 shows the comparison of our approach to SVR, which is considered to be the state-of-the-art. As a cross-validation is too expensive, we only searched for a good parameter guess on 10% of the training data and used the parameters $C = 10,000$ and $\sigma = 1e - 5$ in the comparison. Our SVR ensemble approach is using $k = 64$ and $n = 2,000$. In four of five cases, our proposed prediction algorithm outperforms the SVR algorithm. A conclusion cannot be given without considering the runtime needed. The training and testing of our algorithm only needs a few minutes, giving better results than the SVR algorithm. The state-of-the-art SVR approach can easily take half an hour or longer for training. We point out that the runtimes are implementation- and hardware-specific and therefore the advantage of the ensembles over the SVR can differ on other systems. However, due to the theoretically runtime behavior and the difficulties of a parallel SVR implementation, we consider our proposed algorithm to be the better choice in most situations.

Another experiment shows the behavior of an ensemble using $T = 64$ estimators each using $S = 1,000$ samples when different forecast horizons Δt



(a) Cheyenne



(b) Lancaster

Fig. 5.5. Comparison of SVR ensembles and SVR for different forecast horizons.

Table 5.2. Comparison between SVR using CV-sought parameters and SVR ensemble regressor (SVRENS) using $S = 2,000$ and $T = 64$. For every turbine, the best result is printed in bold.

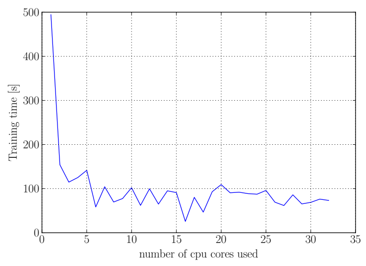
Algorithm	SVR			SVRENS		
	Train	Test	Error	Train	Test	Error
Cheyenne	1671.55	154.94	7.70	303.12	146.07	**7.54**
Lancaster	2067.55	126.43	9.98	266.35	115.30	**8.87**
Palm Springs	1907.32	87.56	7.59	252.69	83.88	**6.12**
Vantage	2194.81	164.78	8.33	224.04	122.77	**5.55**
Yucca Valley	536.80	115.43	10.40	276.51	121.70	**10.20**

are targeted. The ensemble is compared to a RBF-SVR with $C = 1,000$ and $\sigma = 0.0001$ on two test turbines from the NREL dataset using neighbor turbines within 3 kilometers, year 2004 for training and 2005 for testing. The feature window is chosen as $\mu = \Delta t$.

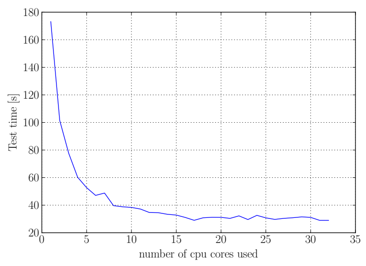
The results are shown in Figure 5.5. While for short horizons both ensembles and SVR perform equally well, the ensembles show a lower MSE for larger horizons, which suggests an improved robustness of the technique.

5.6 Parallel Implementation

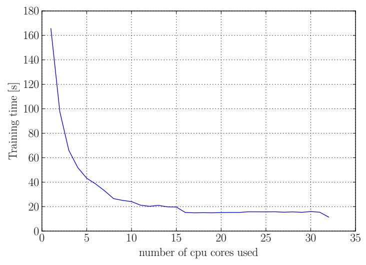
The algorithm we propose in this chapter can be easily parallelized. Our Python implementation is using the *Map & Reduce* paradigm [127]. Although most of our test runs were conducted on single machines with multiple cores, the implementation could be run on a distributed system with many compute nodes. We present an example for the reduction of the computation time in Figure 5.6.



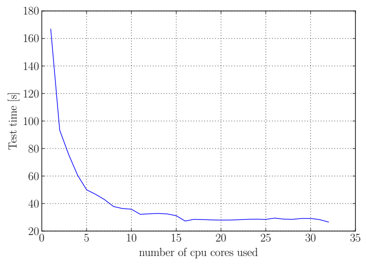
(a) Training with random parameter choice.



(b) Testing with random parameter choice.



(c) Training with fixed parameters.



(d) Testing with fixed parameters.

Fig. 5.6. Training and test time depending on number of cores.

Here, we use a 16-core AMD Opteron 6272 Processor and use $1 \dots 32$ processes for the `map` operation. In the ensemble $T = 32$ estimators are trained. Up to 16 parallel executions, the computation times of the independent estimators decrease. With more than 16 processes there is no further improvement. The results of the parallel implementation show that our algorithm can be employed successfully in applications where a fast training or testing is required and can possibly be provided through multicore processors.

An implementation on highly parallel graphics processors with their manycore architectures is not possible in a straight-forward manner because of the design of the SVR algorithm, different memory access for each estimator and poor performance. For future work an implementation with beneficial use of graphics processing units (GPUs) could be possible, since the field on GPU computing for machine learning is an active area of current research.

5.7 Conclusions

The integration of wind power into the smart grid is only possible with a precise forecast computed in reasonable time. For an improvement of the prediction performance and reduction of computational cost, we presented a SVR ensemble method using bagging and weighted averaging. We showed that we obtain the best results when using a random parameter choice for the estimators. The number of estimators and the samples have to be sufficient in order to provide a reliable forecast accuracy. Compared to state-of-the-art SVR, the prediction error can be decreased while offering a reasonable runtime. In addition to a theoretically lower runtime, a highly parallel implementation using multicore processors has been implemented and shown to perform very efficiently.

6

Combination of Speed and Power Time Series

Different machine learning algorithms can be used for short-term wind power prediction. The question comes up for the choice of appropriate features for the wind power prediction problem. Most past work concentrates on univariate prediction models that map a single time series to target values. Our approach takes into account the features from neighboring turbines, but was in the past restricted to the use of power features. If available, it might be beneficial for the prediction to include the wind speed features as well. Many approaches compute a forecast of the speed and then transform it to a power value using a power curve (PC) model, see [102].

In this chapter, we analyze various regressors trained with patterns composed of different features, i.e., power output measurements, wind speed measurements, and differences of these. We compare models based on k-NN, SVR, and RF that turned out to be very successful in various applications, see [28]. Last, we combine the best combinations of regressors and their features to ensembles and show experimentally that these outperform their single predictor competitors.

This chapter is structured as follows. The experimental study in Section 6.1 compares the use of the different feature spaces with the different regression algorithms. The combination of these predictors to an ensemble is analyzed in Section 6.2. Conclusions are drawn in Section 6.3.

6.1 Comparison of Input Patterns

6.1.1 Comparing Speed and Power Features

When applying machine learning algorithms to real-world data, the choice of appropriate features is important for achieving good prediction results. For our wind prediction task, there are two time series available, i.e., wind speed and wind power measurements of every turbine. While previous work often took only into account time series itself for prediction of future values, it could be beneficial to include all available data. In particular for wind, there is an important relation between the speed and the power values since the power output is a function of the actual wind speed.

For both available time series we also consider to preprocess the time series by including the differences of each measurement to the measurement before, thus including the slope as a feature. The use of these differences could contain a recent trend and helpful information to recognize recurring situations appearing. We expect that different regression algorithms show different behaviors when running on other feature representations. It may be that one algorithm performs better on a particular feature set while another yields a lower prediction error for another feature set.

Our objective is to compare the use of the two available time series and the preprocessed time series using RF, SVR, and k-NN regression methods. Because the choice of the right parameters is crucial for a good prediction result and the prevention of overfitting, we perform a grid search with 5-fold cross-validation (CV) on the training set with data from 01/2004 to 06/2005 to determine the optimal parameters for each turbine, the corresponding feature representation and the algorithm used. The trained predictor is then employed to make a prediction for a test set with data from 07/2005 to 12/2006. For RF, we vary the number of estimators chosen from $\{32, 64, 128, 256\}$. For the SVR, regularization parameter λ is chosen from $\{1, 10, 100, 1000\}$ using an RBF-kernel chosing its bandwidth σ from $\{0.0001, 0.001, 0.01, 0.1, 1.0\}$. For k-NN, k is chosen from $\{1, 5, 15, 20\}$.

Table 6.1. Comparison of MSE of RF, SVR, and *k*-NN using power output (P), speed (S), and differences ($\varDelta$) of the particular time series. For each target turbine, the best prediction error is printed in bold figures. The best prediction per algorithm and turbine is underlined.

(a) CV error

	RF				SVR				*k*-NN			
Turbine	P	P+$\varDelta$	S	S+$\varDelta$	P	P+$\varDelta$	S	S+$\varDelta$	P	P+$\varDelta$	S	S+$\varDelta$
Casper	10.54	$\underline{\mathbf{10.16}}$	11.41	10.69	10.31	$\underline{10.20}$	11.17	10.89	11.91	$\underline{11.78}$	13.63	13.05
Cheyenne	7.72	7.61	7.62	$\underline{\mathbf{7.27}}$	7.79	7.75	7.54	$\underline{7.34}$	$\underline{8.48}$	8.56	8.66	8.33
Hesperia	8.04	7.78	7.68	$\underline{\mathbf{7.16}}$	8.01	7.99	7.46	$\underline{7.30}$	9.30	9.18	9.50	$\underline{9.13}$
Lancaster	9.44	8.99	9.30	$\underline{\mathbf{8.28}}$	8.90	8.85	8.77	$\underline{8.66}$	10.45	$\underline{10.36}$	11.28	10.79
L.V.	9.92	9.49	10.12	$\underline{9.36}$	9.36	$\underline{\mathbf{9.30}}$	9.96	9.66	11.03	$\underline{10.79}$	11.88	11.42
P.S.	6.04	5.95	5.31	$\underline{\mathbf{4.90}}$	5.85	5.82	$\underline{5.05}$	4.97	7.51	7.53	7.67	$\underline{7.41}$
Reno	12.37	11.69	11.68	$\underline{\mathbf{10.67}}$	11.82	11.67	11.67	$\underline{11.29}$	15.15	14.98	15.48	$\underline{14.84}$
Tehachapi	7.65	$\underline{7.41}$	8.35	7.89	7.44	$\underline{\mathbf{7.35}}$	8.04	7.98	9.02	$\underline{9.00}$	9.86	9.49
Vantage	6.06	5.95	5.82	$\underline{\mathbf{5.42}}$	5.97	5.97	$\underline{5.59}$	$\underline{5.59}$	6.88	7.02	7.02	$\underline{6.81}$
Y.V.	11.34	11.23	10.98	$\underline{\mathbf{10.64}}$	11.40	11.27	10.93	$\underline{10.76}$	12.21	12.24	12.10	$\underline{11.77}$

(b) Test error

	RF				SVR				*k*-NN			
Turbine	P	P+$\varDelta$	S	S+$\varDelta$	P	P+$\varDelta$	S	S+$\varDelta$	P	P+$\varDelta$	S	S+$\varDelta$
Casper	10.62	$\underline{\mathbf{10.01}}$	12.25	11.41	10.76	$\underline{10.43}$	11.59	11.27	12.24	$\underline{12.05}$	14.30	13.74
Cheyenne	7.29	7.21	7.12	$\underline{6.71}$	7.17	7.10	6.75	$\underline{\mathbf{6.70}}$	7.93	7.99	7.87	$\underline{7.61}$
Hesperia	7.69	7.50	7.39	$\underline{\mathbf{6.91}}$	7.62	7.70	7.21	$\underline{6.95}$	8.76	8.59	8.92	$\underline{8.49}$
Lancaster	8.19	8.05	7.66	$\underline{\mathbf{7.06}}$	8.01	7.88	7.43	$\underline{7.23}$	9.15	9.01	9.34	$\underline{8.94}$
L.V.	10.52	9.98	10.23	$\underline{\mathbf{9.50}}$	10.43	$\underline{10.32}$	10.88	10.61	$\underline{11.85}$	11.69	12.43	11.92
P.S.	5.84	5.80	5.23	$\underline{\mathbf{4.78}}$	5.94	5.86	4.95	$\underline{4.83}$	7.11	7.06	7.25	$\underline{7.00}$
Reno	13.94	13.39	14.04	$\underline{\mathbf{12.78}}$	13.66	$\underline{13.39}$	14.23	13.68	16.65	$\underline{16.52}$	17.58	16.88
Tehachapi	7.30	$\underline{\mathbf{7.07}}$	7.69	7.18	$\underline{7.23}$	7.27	8.15	8.04	$\underline{8.45}$	$\underline{8.45}$	9.75	9.41
Vantage	6.81	6.69	6.51	$\underline{\mathbf{6.11}}$	6.64	6.60	6.34	$\underline{6.30}$	$\underline{7.83}$	7.84	8.17	7.92
Y.V.	9.14	9.18	8.72	$\underline{\mathbf{8.51}}$	9.05	9.00	8.66	$\underline{\mathbf{8.51}}$	9.73	9.84	9.57	$\underline{9.29}$

The results of the comparison are presented in Table 6.1. For all three regression algorithms, it is a good choice to include differences of the used features for most of the turbines. For RF and SVR, using the speed time series for predicting the power output seems more promising than the power features . For the k-NN experiments, there is no clear answer, if the power or the speed features should better be preferred. We can observe that RF regression outperforms the other two regression algorithms for eight out of ten turbines w.r.t the CV error. The analysis of test errors shows that the CV-selected type of features, parameters, and algorithm choices perform similarly well on the test sets, too.

6.1.2 Combining Speed and Power Features

In the following, we combine the features of both time series to one big pattern. Table 6.2 shows the experimental comparison for these composed features based on the power and speed (Tables 6.2 a and b) and also including the differences between the measurements (Tables 6.2 c and d). Because the patterns have different dimensionalities and different types of features, a parameter search is necessary for each experiment. Again, grid search with 5-fold CV is performed to find the best possible settings for each algorithm and target turbine. The main idea of the comparison is that the algorithms could behave differently when different features are used.

Tables 6.2 (a,b) show that without employing the feature differences, only for a few turbines the prediction can be improved for a particular algorithm. The optimal CV error and test error per turbine cannot be outperformed. However, when including the power and speed difference features to the pattern, a great improvement of CV error and test error can be observed for both RF and SVR, see Tables 6.2 (c,d). In nine out of ten cases, the optimal prediction error from Table 6.1 is outperformed. Therefore, we strongly recommend to consider the use of patterns consisting of the power measurements, the speed measurements, and their differences if available.

Table 6.2. Prediction error for combined patterns using power and speed measurements (a,b) without and (c,d) with differences included. The prediction errors lower than the ones using the same regression algorithm on the single time series presented in Table 6.1 are underlined. A lower error than the turbine optimum from Table 6.1 is printed in italics. The best value per turbine in each table is printed in bold figures.

(a) CV error

Turbine	RF	SVR	k-NN
Casper	*10.08*	***9.98***	11.91
Cheyenne	**7.51**	7.64	8.42
Hesperia	**7.26**	7.89	9.35
Lancaster	8.68	**8.61**	10.61
L.V.	9.57	**9.37**	11.21
P.S.	**5.36**	5.41	7.46
Reno	**11.08**	11.52	15.10
Tehachapi	***6.85***	7.42	9.00
Vantage	**5.80**	5.89	6.88
Y.V.	**10.93**	11.17	12.18

(b) Test error

Turbine	RF	SVR	k-NN
Casper	***9.69***	10.24	12.23
Cheyenne	7.09	***6.99***	7.89
Hesperia	**7.22**	7.38	8.70
Lancaster	7.59	**7.52**	9.20
L.V.	**9.78**	10.39	11.94
P.S.	**5.25**	5.29	7.08
Reno	**12.84**	13.34	16.55
Tehachapi	***6.23***	7.31	8.57
Vantage	**6.33**	6.48	7.92
Y.V.	**8.75**	8.89	9.60

(c) CV error with Δ

Turbine	RF	SVR	k-NN
Casper	***9.28***	*9.85*	11.74
Cheyenne	***7.15***	7.50	8.41
Hesperia	***6.70***	7.78	9.12
Lancaster	***7.65***	8.53	10.51
L.V.	***8.79***	9.18	10.97
P.S.	***4.88***	5.38	7.34
Reno	***10.14***	11.21	14.84
Tehachapi	***6.29***	*7.25*	8.91
Vantage	**5.47**	5.83	6.87
Y.V.	***10.55***	11.03	12.08

(d) Test error width Δ

Turbine	RF	SVR	k-NN
Casper	***9.03***	10.04	11.94
Cheyenne	***6.66***	6.80	7.87
Hesperia	***6.59***	7.21	8.52
Lancaster	***6.87***	7.48	8.99
L.V.	***9.09***	10.14	11.70
P.S.	**4.84**	5.20	6.88
Reno	***11.58***	12.80	16.32
Tehachapi	***5.78***	7.18	8.51
Vantage	***5.88***	6.36	7.85
Y.V.	***8.50***	8.78	9.66

6.2 Ensemble Combination

6.2.1 Combination of Predictors Based on Different Time Series

One of the main reasons for the success of ensemble predictors is the diversity amongst the combined predictors, i.e., a different behavior or uncorrelated prediction errors. In Section 6.1, we showed that the use of the different available time series features leads to an improvement of the prediction error. The question arises if we can further decrease the prediction error by combining regressors based on the different time series features. First, we combine one predictor based

Table 6.3. Comparison of MSE with combinations of predictors. Each combination consists of one prediction based on power time series and one prediction based on speed time series – using RF or SVR. For every park, the best value is printed in bold. Every value that outperforms the predictors from Table 6.1 for a given park is underlined. Every predictor that outperform the ones shown in Table 3 is printed in italics.

Turbine	RF+RF	RF+SVR	SVR+RF	SVR+SVR
Casper	9.69	**9.40**	9.67	9.64
Cheyenne	6.71	***6.60***	***6.60***	***6.60***
Hesperia	**6.71**	6.82	6.77	7.03
Lancaster	7.01	6.94	**6.88**	7.05
L.V.	**9.20**	9.59	9.28	9.93
P.S.	4.88	4.86	*4.83*	4.97
Reno	**11.87**	12.26	11.80	12.68
Tehachapi	**6.36**	6.58	6.44	6.87
Vantage	5.99	5.99	**5.94**	6.13
Y.V.	8.50	*8.47*	***8.36***	*8.49*

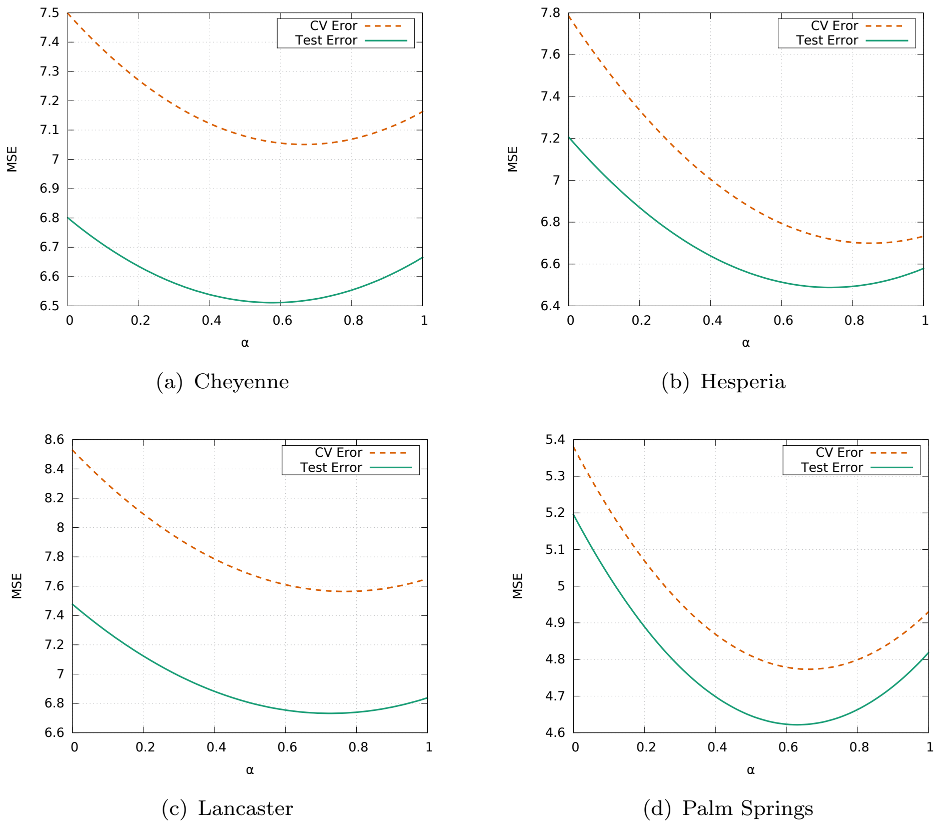
on the power time series with one predictors based on the speed time series, each also using the differences between the particular time steps. The main idea is that the algorithms behave differently in different feature space and therefore give very diverse predictions for the same target time step. For both the power and speed time series patterns, each an RF and an SVR predictor are trained separately. To give a prediction, one predictor based on the power features and one predictor based on the speed features is selected and then combined by computing the mean of the two predictors' output values. In Table 6.3, we can observe that a lower prediction error is possible compared to the regressors from Table 6.1, but in seven out of ten cases, the prediction is worse than the one based on the combined pattern shown in Tables 6.2 (c) and (d). With a weighted average, we could decrease the prediction error further, but still yielding no competitive results to the predictors using the combined patterns.

6.2.2 Combination of Predictors Based on All Available Features

As shown in Section 6.1, both RF and SVR yield very good prediction results when using combined patterns including power and speed measurements as well as the computed differences. In this section, we combine the predictors to an ensemble. We only consider RF and SVR because of their superiority compared to k-NN. The two prediction values are combined by computing a weighted average with $\alpha \in (0, 1)$:

$$f(\mathbf{x}) = \alpha \cdot f_{RF}(\mathbf{x}) + (1 - \alpha) \cdot f_{SVR}(\mathbf{x}) \tag{6.1}$$

Figure 6.1 shows the experimental results for four selected turbines. From the plots, we can observe that the combination of the two predictors helps to improve the prediction error. While the special case $\alpha = 0.5$ is a good first guess, the best value varies from turbine to turbine. Finding a feasible α is a parameter tuning problem which we address with CV. The question arises, if an α value that performs best in CV on the training set, can also give a near-optimal solution for the prediction on the test set. The experimental results shown in Table 6.4 demonstrate the practical relevance of the proposed approach. In the CV, we optimize the setting for α w.r.t the lowest CV error. When using the optimal α found in the CV for the test set prediction, we observe a competitive test error. The achieved error and the found α are very close to the best possible α and test error for the test set, which can never be known in advance. predictions for eight



(a) Cheyenne (b) Hesperia

(c) Lancaster (d) Palm Springs

Fig. 6.1. CV and test error of linear combination of RF and SVR predictions with all available features for varying coefficient α.

out of ten turbines, compared to the use of single predictors from Section 6.1. The same behavior can be observed in Figure 6.1, where the plots of CV error and test error have very similar shapes and optima. Summing up, we strongly recommend the combination of diverse predictors based on the combined patterns for wind power prediction.

Table 6.4. CV optimization of coefficient α . The CV error and test error with the found α are compared to the best possible α on the test set. The CV errors outperforming the single predictors shown in Table 6.2 (c) and the test errors outperforming the single predictors from Table 6.2 (d) are printed in italics.

Turbine	Best α (CV)	CV error	Test error	Best α (test)	Test error
Casper	0.67	*9.13*	*8.91*	0.73	8.90
Cheyenne	0.67	*7.05*	*6.52*	0.58	6.51
Hesperia	0.85	6.70	*6.50*	0.84	6.50
Lancaster	0.77	*7.56*	*6.74*	0.73	6.73
L.V.	0.62	*8.55*	9.12	0.84	9.04
P.S.	0.66	*4.77*	*4.62*	0.63	4.62
Reno	0.73	*9.92*	*11.35*	0.71	11.34
Tehachapi	0.78	*6.20*	5.84	1.0	5.76
Vantage	0.70	*5.40*	*5.80*	0.77	5.80
Y.V.	0.69	*10.43*	*8.34*	0.64	8.34

6.3 Conclusions

The integration of wind power generation into the smart grid can only succeed with precise and reliable forecast methods. With different measurements available, machine learning algorithms are able to achieve very good predictions for short-term horizons. In this chapter, we compared the use of wind power and wind speed time series with RF, SVR, and k-NN regression. While both power and speed features contain an essential amount of information for the prediction task, we showed experimentally that both should be combined in order to improve the prediction error. It can be further increased by including the differences of the power and speed measurements, which allows important insights into the trend. Further, we proposed an ensemble method for wind power prediction employing one RF and one SVR regressor and combined both with a weighted average. A near-optimal weighting coefficient α can be determined by cross-validation and beats the single predictors based on the different feature spaces.

7

Heterogeneous Ensembles

There are two big challenges when applying machine learning techniques to the field of wind power prediction: First, in order to achieve the best prediction accuracy possible with these algorithms, the computation time grows very large. For instance, training an SVR can easily take hours – strongly increasing with the number of considered neighboring turbines and past measurements. Second, the prediction performance needs to be improved further to cope with the actual energy markets needs. These two aspects make the choice of machine learning algorithm and parameters for wind power forecasting difficult.

In this chapter, we discuss the practical use of regression ensembles for the task of wind power forecasting, aiming at optimal regression accuracy as well as maintaining a reasonable computation time. In the first step we compare homogeneous ensemble predictors consisting of either *decision trees* (DT), k-NN, or support vector regressors as base algorithms. As diversity among the ensemble members is crucial for the accuracy of the ensemble, we propose the use of *heterogeneous ensemble predictors* consisting of different types of base predictors for wind power prediction. Our comprehensive experimental results on five test turbines show that a combination of DT and SVR yields better results than the analyzed homogeneous predictors while offering a decent runtime behavior. Going further, we show that heterogeneous ensemble predictors are very well-suited for using large numbers of neighboring turbines and past measurements and improve the prediction performance.

This chapter is structured as follows: The application of regression ensembles to the field of wind power prediction and a comprehensive experimental analysis are presented in Section 7.1, followed by Section 7.4 dealing with heterogeneous ensemble predictors. The analysis of the question, how to use the largest possible amount of valuable information from the available data is done in Section 7.5. In Section 7.6, the heterogeneous ensemble prediction model is applied to the power output prediction of wind parks. Conclusions are drawn in Section 7.9.

7.1 Algorithmic Framework

Our objective is to find out if heterogeneous machine learning ensembles are a superior alternative to state-of-the-art machine learning predictors. We decided to implement a relatively simple bagging approach with weighting, which has some advantages. While the implementation is straight-forward and offers a moderate computational cost, we consider the approach sufficient for a proof of concept, which is also shown in the experimental evaluation. Another good example for this ensemble algorithm is the famous Random Forest method that yields very good results, too, and is relatively fast compared to Boosting algorithms. The latter ones could outperform for some applications, but also tend to overfitting, and can hardly be parallelized.

Our algorithm is outlined in Algorithm 3. As usual in supervised learning, a training set $\mathbf{X}$ with known labels is given. The most important meta-parameters of the algorithm are the number T of weak predictors, the number S of samples, and the types of base algorithms used for each predictor. Both T and S have to be chosen large enough in order to provide satisfying results. However, the best choice for T and S depends on the base algorithm used, which also influences the runtime significantly. The main work of the algorithm takes place in the `for`-loop beginning in line 3. Each pass trains one weak predictor p_i and its assigned weight w_i. For each weak predictor, a subset of $\mathbf{X}$ with size s is sampled and used as

Algorithm 3 Training of Ensemble Predictor

```
1: Inputs:
      $\mathbf{X} = \{(\mathbf{x}_1, y_1) \dots, (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^d \times \mathbb{R}$
      Number of predictors: $T$
      Number of samples: $S$
      Algorithms to use: $\mathbf{A} = \{a_i | i \in 1 \dots T\}$
2: Returns:
      Predictors: $\mathbf{P} = \{p_i | i = 1, \dots, T\}$
      Weights: $\mathbf{W} = \{w_i \in \mathbb{R} | i = 1, \dots, T\}$
3: for i = 1 to $T$ do
4:    $\mathbf{X}_{sample} \leftarrow sample(\mathbf{X}, s)$
5:    $\mathbf{X}_{val} \leftarrow \mathbf{X} - \mathbf{X}_{sample}$
6:    $p_i \leftarrow$ Train model using algorithm $a_i$ on $\mathbf{X}_{sample}$
7:    $w_i \leftarrow \frac{1}{MSE(p_i, \mathbf{X}_{val})}$
8: end for
```

training set $\mathbf{X}_i$ for the particular predictor p_i. The sampling can be done with or without replacement, which we will discuss in the experimental part. In order to calculate weight w_i, the remaining training patterns are used as a validation set $\mathbf{X}_{val}$. The value w_i is then obtained by testing p_i on $\mathbf{X}_{val}$ and taking the inverse of the MSE.

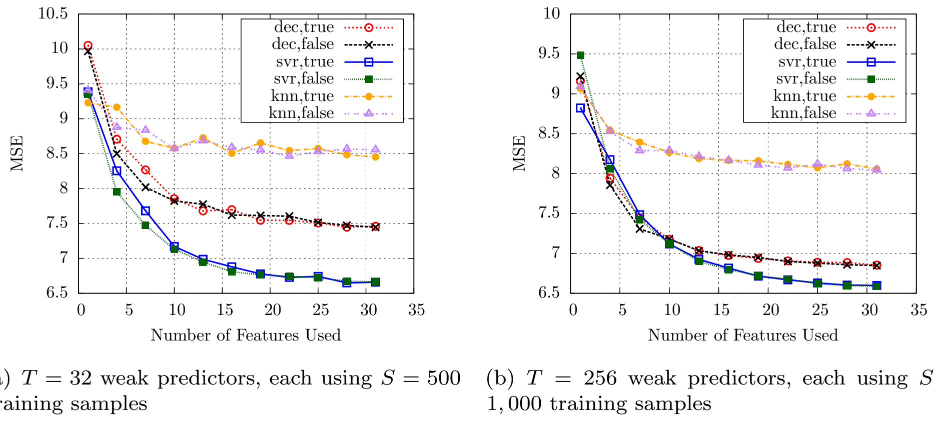
When the training algorithm computed the predictors and weights, for an unknown instance $\mathbf{x}'$ the predicted label is computed by:

$$f(\mathbf{x}') = \frac{\sum_{i=1}^{T} w_i \cdot p_i(\mathbf{x}')}{\sum_{i=1}^{T} w_i} \tag{7.1}$$

Each predictors output $p_i(\mathbf{x})$ is computed and then weighted by w_i in the resulting weighted average. In a realistic scenario, one would perform all calls of p_i in parallel using multi- or manycore processors. Because there are no computational dependencies between the ensemble members, the problem is embarrassingly parallel. To give an easy and fair comparison, in our experiments we only employ only one CPUcore for the runtime measurements. As depicted in Table 7.1, the runtime for training depends on the base algorithm used, the number of estimators employed, and the number of samples used. E.g., an ensemble of 32×500 DT ensemble regressor can be trained in only one second, whereas a $256 \times 1,000$ SVR ensemble needs more than ten minutes. Because the different ensemble predictors yield different prediction accuracies, the computation time should be considered when choosing a model for practical use. In Section 7.4, we investigate the problem of giving the best possible prediction in a short computation time.

7.2 Sampling of Features and Patterns

As pointed out by [46], random forests and bagging classifiers in general benefit from sampling from the feature space. That is, taking only a random subset of the available features into account. Besides the number of features used, the choice can be done with ot without replacement. Using replacement is sometimes considered as useful [24, 94], but there is no explicit rule for every situation. Therefore, we evaluated the behavior depending on the number of features as well



(a) $T = 32$ weak predictors, each using $S = 500$ training samples

(b) $T = 256$ weak predictors, each using $S = 1,000$ training samples

Fig. 7.1. Impact of random sampling of features with and without replacement on the regression error of ensemble regressors for a wind park near Vantage. From the 33 features available, only a random subset was used. The values given in the plots are calculated by the mean of 10 runs. The two graphs for each regression algorithm used as weak predictor strongly overlap and can hardly be distinguished.

as the boolean replacement parameter. Because we only want to get the basic idea of the behavior, we visualize the results for one wind park in Figure 7.1 for two setups of sample size and number of predictors. The results are exemplary for the other tested parks, too. First, we found that increasing the number of features used resulted in a lower regression error. But one must not forget that a lower number of features also results in a shorter computation time, so a trade-off is necessary. Second, we do not find any evidence that sampling with replacement is superior or inferior to sampling without replacement. Therefore, we employ sampling without replacement. For the basic comparison of the weak predictors, all features are used. For the comparison of heterogeneous ensembles, the number of features sampled will be considered again because of the possible trade-off between runtime and accuracy.

Concerning the training samples, we found no evidence for the supremacy of sampling with or without replacement, but due to the recommendations in literature [12, 94], we decided to employ sampling with replacement in the following experiments.

For both regression performance and runtime, the sample size is an important factor as well. Because it cannot be examined separately without looking at the number of weak predictors, we analyze both aspects together in the following experiments.

7.3 Choice of the Base Algorithms

Since the number of possible settings is huge, one has to make some assumptions to limit the number of combinations. The decision tree algorithm is a powerful yet simple tool for supervised learning problems [39]. While there are different algorithms for building up decision trees, we limit ourselves to the famous CARTalgorithm [14]. A famous yet relatively simple approach for classification and regression is the *k-nearest neighbors* (k-NN) model, see [121]. For parameter k, we make a random choice in the interval $[1; 25]$. The SVR algorithm often provides very good prediction results and is regarded as state-of-the-art regression technique. We utilized a RBF kernel and choose $C = 10,000$ and $\sigma = 1e - 5$

for this experiment. In the following sections, cross-validation is utilized for parameter-tuning.

We experimentally compare different regression algorithms composed to ensembles: Table 7.1 shows a comparison of decision trees, SVR, and k-NN used as weak predictors. A general observation is that increasing T and S decreases the prediction error. With the given T and S, no clear decision between decision trees and SVR can be made, but we stick to these two basic algorithms in the further experiments rather than k-NN.

Table 7.1. Comparison (MSE) of ensemble predictors consisting of different base algorithms used as weak predictor. For every turbine, the best result is printed in bold. Each experiment has been repeated 10 times.

Base Algorithm	DT				SVR				k-NN			
T	32	32	256	256	32	32	256	256	32	32	256	256
S	500	$1,000$	500	$1,000$	500	$1,000$	500	$1,000$	500	$1,000$	500	$1,000$
Casper	11.17	10.93	10.89	**10.62**	10.99	10.84	10.95	10.87	13.10	12.44	13.02	12.44
Las Vegas	10.84	10.61	10.51	10.27	**10.26**	**10.26**	10.27	10.27	12.84	12.36	12.81	12.30
Hesperia	7.98	7.82	7.76	**7.59**	7.62	7.61	7.60	**7.59**	9.41	8.96	9.36	8.98
Reno	14.76	14.53	14.47	14.19	14.11	14.10	14.00	**13.98**	18.92	18.03	19.14	18.14
Vantage	7.31	6.97	7.00	6.83	6.61	6.58	**6.57**	**6.57**	8.44	7.93	8.43	8.07
Approx. t_{train} (s)	1	2	10	15	60	120	600	1,200	36	60	292	481

7.4 Heterogeneous Ensembles

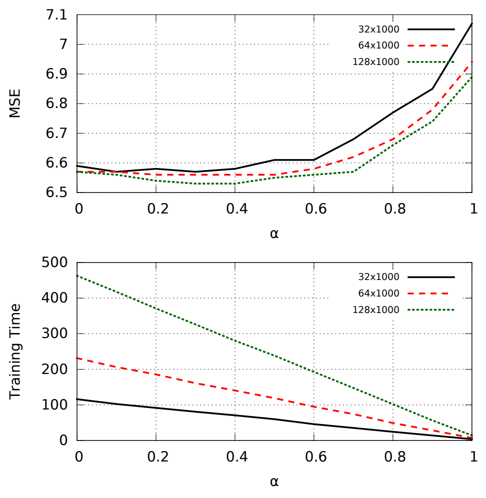
Of course, when dealing with forecasting tasks, the first goal is to reach the lowest prediction error possible. While it is possible to decrease the prediction error by using ensembles, a feasible runtime is equally important for practical relevance. If it takes hours to train a regressor for one turbine, a model would be unusable for a large number of turbines requiring a forecast – the time needed for parameter-tuning and cross-validation not mentioned. Therefore, our goal is to reach a good prediction performance as well as a short runtime for both training and testing. As seen in Table 7.1, there is no clear answer which algorithm should be preferred. Since it is known that ensemble predictors benefit from diversity amongst the ensemble members, it a heterogeneous choice of base algorithms could yield a better regression accuracy as well.

7.4.1 Mixed Ensemble with Coefficient α

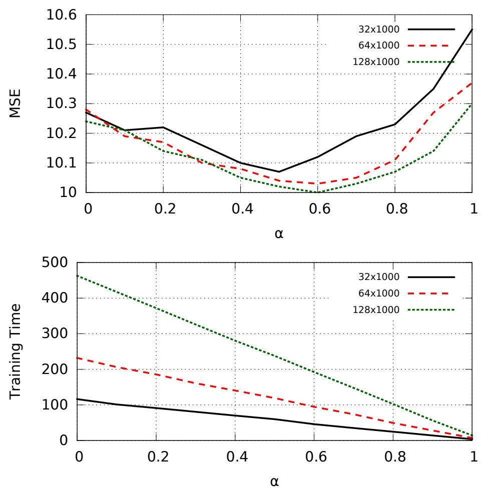
First, we analyze heterogeneous ensembles that employ both an amount of SVR predictors and an amount of decision trees. We define a coefficient α that specifies the amount of weak predictors trained with the decision tree algorithm. Hence, $1-\alpha$ is the amount of SVR weak predictors in the ensemble. Figure 7.2 shows the experimental results for two wind turbines. For both (a) and (b), three ensemble settings were analyzed showing MSE and training time: S is set to $1,000$ and T is set to 32, 64, or 128. The higher the number of predictors is, the lower the prediction error becomes. But also the runtime increases notable. With $\alpha = 0$, we observe an ensemble with only SVR predictors, with $\alpha = 1$, only decision trees are chosen. The interesting point is that, given a sufficient number T, the prediction quality gets best with an α in the middle range. This points to an advantageous behavior of heterogeneous ensembles: With an α in the near of 0.5, the training and testing time of the ensemble predictor decreases dramatically compared to $\alpha = 0.0$. Therefore, the parameter α can be seen as explicit tuning parameter for the trade-off between prediction performance and computation time.

7.4.2 Ensemble Combining SVR and DT Ensembles

While the results of the former experiment could be pleasant for the user, we have to point out that the parameter α was varied for fixed T and S. One has to consider that different weak predictors show different behavior and could benefit from different combinations of T and S. I.e., we will see that a large amount of



(a) Mixture α for a wind turbine near Vantage

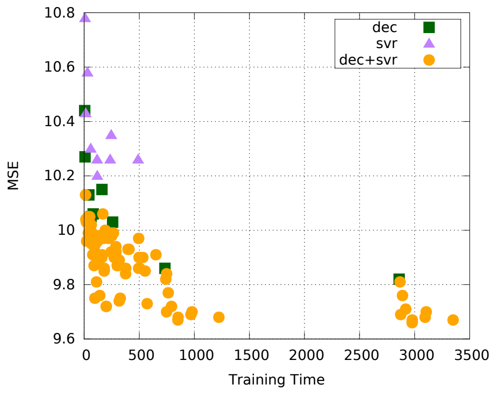


(b) Mixture α for a wind turbine near Las Vegas

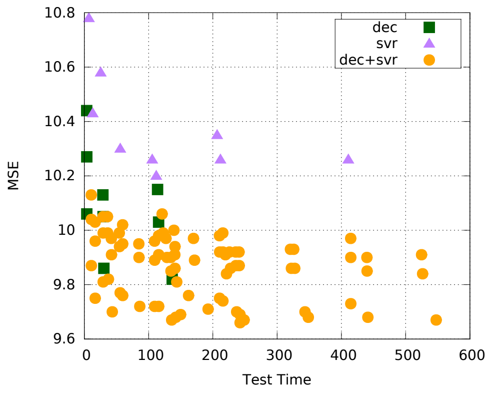
Fig. 7.2. Mixing of SVR and DT. With a balanced mixture, a better MSE is reached within a shorter training time.

predictors is a good possibility to ameliorate decision tree ensembles while the runtime does not suffer as much as in SVR ensembles. Instead of combining some SVR predictors and some decision trees, one could possibly better combine a huge number of decision trees and maybe also increase sample number S. In this work, we propose to use heterogeneous ensembles built upon SVR and decision tree regressors. We chose the most simple approach by training one SVR ensemble and one DT ensemble. The predicted value is then obtained by computing the mean of the two ensembles' predictions. As shown in the following experiments, the result is a robust prediction algorithm that offers a reasonable runtime behavior. The experiments in the following section address the question, if heterogeneous ensembles offer a better performance than homogeneous ensembles. The second question is: Can heterogeneous ensembles help to decrease the computation time needed?

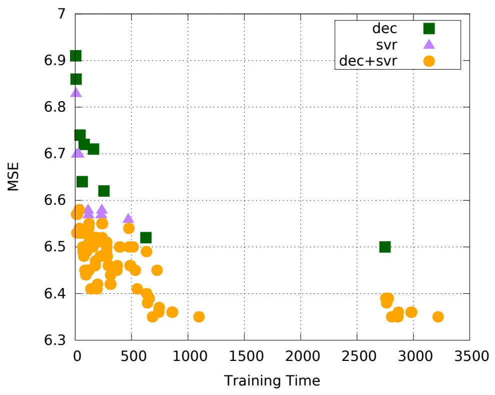
In our experiments, we analyze heterogeneous ensembles built upon SVR and decision tree regressors. In our experiments, we use the power output data



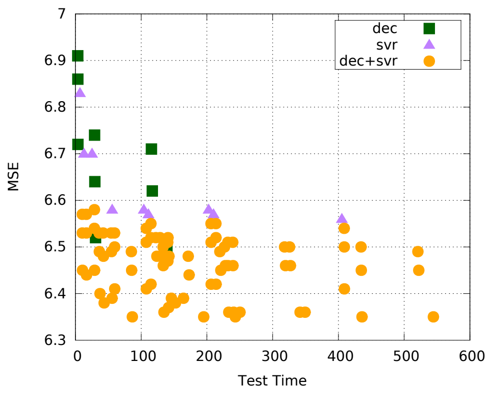
(a) Las Vegas: MSE depending on training time



(b) Las Vegas: MSE depending on test time



(c) Vantage: MSE depending on training time



(d) Vantage: MSE Depending on test time

Fig. 7.3. Behavior of runtime and prediction performance for homogeneous and heterogeneous ensembles for two wind turbines near Las Vegas and Vantage. The heterogeneous combinations can outperform the homogeneous ensembles. In particular, the solutions in each bottom left corner show ensembles with a very short computation time as well as a very low error.

of the five wind parks and include the measurements of 10 neighbored turbines into the patterns. As training data set, the data from 01/2004 until 06/2005 is used and the data from 7/2005 until 12/2006 serves as test data set. The experiments were run on an Intel Core i5 at 3.10GHz with 8GB of RAM. The algorithms were implemented in Python utilizing the kNN, decision tree, and SVR implementations of *Scikit-learn* [88].

Table 7.2. Behavior of different setups of SVR, DT, and combined ensembles for a wind turbine near Las Vegas.

(a) SVR ensembles

setup	T	S	t_{train}	t_{test}	MSE
1	4	500	7.58	7.11	10.78
2	4	1,000	14.69	13.09	10.43
3	4	2,000	29.94	25.46	10.58
4	32	500	59.73	55.91	10.30
5	32	1,000	117.86	105.61	10.26
6	32	2,000	245.85	206.68	10.35
7	64	500	119.40	111.91	10.20
8	64	1,000	236.26	211.83	10.26
9	64	2,000	489.92	410.81	10.26

(b) Decision Tree Ensembles

setup	T	S	t_{train}	t_{test}	MSE
1	32	2000	4.91	3.65	10.44
2	32	4,000	7.74	3.85	10.27
3	32	40,000	81.96	3.82	10.06
4	256	2,000	42.67	28.86	10.13
5	256	4,000	63.17	29.17	10.05
6	256	40,000	733.15	30.35	9.86
7	1,024	2,000	161.60	113.97	10.15
8	1,024	4,000	258.03	115.55	10.03
9	1,024	40,000	2,859.18	136.71	9.82

(c) Combinations of one DT ensemble and on SVR ensemble to one heterogeneous ensemble. For every combination, the MSE is shown. The row denotes which SVR ensemble is employed, whereas the column shows which DT ensemble is used.

	D1	D2	D3	D4	D5	D6	D7	D8	D9
S1	10.13	10.04	9.87	10.05	9.99	9.82	10.06	9.99	9.81
S2	10.03	9.96	9.75	9.97	9.91	9.70	9.97	9.90	9.69
S3	10.05	9.99	9.81	9.99	9.94	9.77	10.00	9.94	9.76
S4	10.02	9.95	9.76	9.95	9.90	9.72	9.97	9.89	9.71
S5	9.96	9.89	9.72	9.90	9.85	9.67	9.91	9.84	9.66
S6	9.98	9.92	9.75	9.92	9.87	9.70	9.93	9.86	9.70
S7	9.98	9.91	9.72	9.91	9.86	9.68	9.92	9.86	9.67
S8	9.99	9.92	9.74	9.92	9.87	9.69	9.93	9.86	9.68
S9	9.97	9.90	9.73	9.90	9.85	9.68	9.91	9.84	9.67

One has to consider that different weak predictors show different behavior and could benefit from different combinations of T and S. I.e., we will see that a large amount of predictors is a good possibility to ameliorate decision tree ensembles while the runtime does not suffer as much as in SVR ensembles. Instead of combining some SVR predictors and some decision trees, one could possibly better combine a huge number of decision trees and maybe also increase sample number s.

Therefore, we have to give a fair comparison, which considers both prediction performance and runtime. First, we analyze the behavior of the homogeneous ensembles based on SVR or decision trees. We try to find good combinations, which are computable in a feasible time. The result can be seen in Table 7.2: Like assumed, one can train more decision trees with a larger sample in the same time as SVR predictors. The central point of the experiment is the equally-weighted combination of one SVR ensemble and one DT ensemble at a time to one heterogeneous ensemble.

The results of these combinations are depicted in Table 7.2(c), which has the form of a matrix. In every cell of the matrix, the used SVR ensemble is given by the row and the used decision tree ensemble is given by the column. In the table, only the MSE is given for clear arrangement. The training and test times for one predictor is approximately the sum of the respective times of the two combined ensembles. Besides the very promising results, which outperform the homogeneous ensembles, we also can see that the combination of two weaker ensembles takes less time to deliver the same prediction error. We visualize this behavior for two wind turbines in Figure 7.3.

In Table 7.3, we give a comparison of our heterogeneous ensemble method to SVR and k-NN which are considered as state-of-the-art regressors. The parameters k and accordingly C and σ were optimized with a 10-fold cross-validation. The training times are measured using the optimal parameters, so the huge amount for parameter tuning is not included. The testing times showed similar behavior. As comparison, we show two different heterogeneous ensembles. Both consist of one SVR ensemble with $T = 32$ an $S = 1,000$ and one decision tree ensemble with $T = 256$ and $S = 10,000$. The first one uses all 33 features available, whereas the second only makes use of 15 randomly chosen features without replacement. The result of this comparison is that in most cases the ensemble predictor outperforms classical SVR. Further, the training time is much shorter. If one must make a trade-off and decrease training or testing time, he might want to use a feature-reduced variant.

Table 7.3. Comparison of MSE and training time for five wind turbines. Our ensemble using all features yields the best MSE in four cases, but only takes a small amount of the time taken by standard SVR. If training time is considered more important than MSE, on can reduce the number of features used without loosing much of prediction performance.

(a) Test Error (MSE)

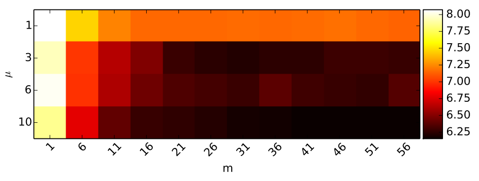
Turbine	k-NN	SVR	ENS33	ENS15
Casper	10.67	9.88	10.19	10.40
Hesperia	7.69	7.39	7.25	7.44
Las Vegas	10.46	15.69	9.78	10.11
Reno	14.81	13.29	13.16	13.83
Vantage	6.86	6.54	6.41	6.65

(b) Training Time (s)

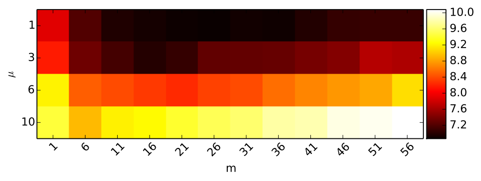
Turbine	k-NN	SVR	ENS33	ENS15
Casper	1	704	413	177
Las Vegas	1	1,450	378	165
Hesperia	1	1,218	387	173
Reno	2	1,173	399	173
Vantage	2	601	374	165

(c) Test Time (s)

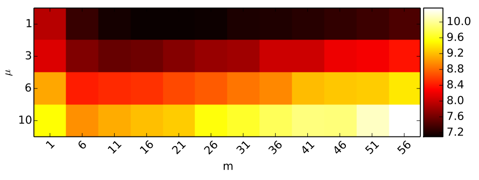
Turbine	k-NN	SVR	ENS33	ENS15
Casper	97	268	214	114
Las Vegas	118	341	206	108
Hesperia	83	261	227	113
Reno	98	253	205	113
Vantage	105	251	229	108



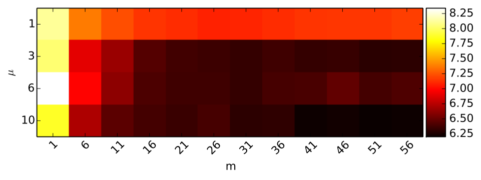
(a) CV-optimized Single SVR



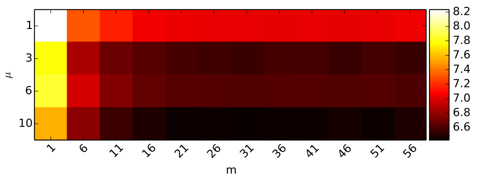
(b) CV-Optimized Single k-NN



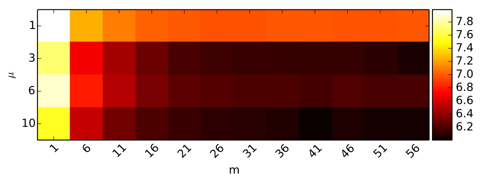
(c) KNN-ENS $64 \times 2,000$



(d) SVR-ENS $64 \times 2,000$



(e) DEC-ENS $64 \times 2,000$



(f) Heterogeneous Ensemble combining (d) and (e)

Fig. 7.4. Behavior of different regression algorithms for varying number of employed neighbor turbines m (x-axis) and feature window μ (y-axis) for a turbine near Vantage. The MSE is visualized by the color: A darker color denotes a lower error while a lighter color appears for higher prediction errors.

7.5 Increasing the Number of Used Features

Besides the parameter-tuning of the algorithms, it must be evaluated how large the number m of considered neighbor turbines and number μ of past time steps should be to get the best prediction error. I.e., one wants to use as much valuable information from the data as possible. Like shown before, there is valuable information existent in the neighboring turbines and in the past time steps. However, with increasing number of features, the data become more and more challenging for the employed regression algorithms: First, the computational time is often dependent on the dimensionality of the data. Second, the prediction accuracy can get worse. E.g., when considering k-NN with Euclidean distance measure, the expressiveness of the distance between two instances is decreased with a higher dimensionality. The computation time also grows large, and for dimensionalities $d > 15$ it is hardly possible to benefit from spatial data structures like k-D-Trees [8].

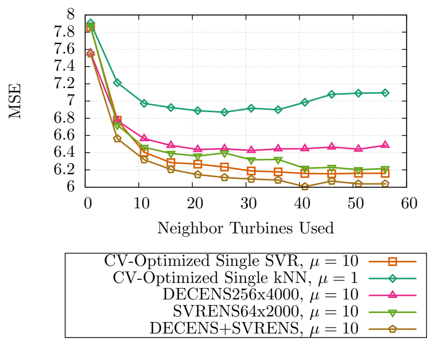


Fig. 7.5. Comparison of MSE depending on number m of used neighbor turbines for six regression algorithms for a turbine near Vantage.

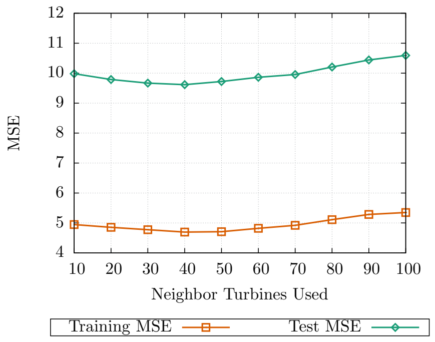


Fig. 7.6. Training and test error for a wind park near Las Vegas using a heterogeneous ensemble using 256 DT with 10,000 samples and 64 SVR predictors using 1,000 samples.

For the practical use of our proposed heterogeneous ensemble predictors, we have to analyze the abilities of the algorithmic framework to make use of as much information in the data as possible and improve the prediction further with tuning of the number of considered neighbor turbines and the past measurements of these and the target turbine.

In a first experiment, we show the behavior of different regression algorithms when varying the number m of neighbor turbines and the feature window μ. While for k-NN and SVR regressors are sought by a grid search and a 3-fold cross validation, for the ensemble predictors we chose a fixed setting. Figure 7.4 shows the MSE depending on m and μ for a turbine near Vantage. For k-NN and k-NN ensembles, the best prediction error is reached with small m and μ, i.e., a small number of features. We assume that k-NN ensembles perhaps could benefit from random feature subsets in order to deal with higher dimensionalities. The SVR predictor as well as the ensemble predictors show a different behavior: With a larger number of neighbor turbines included into the features, the prediction error gets smaller. But the feature window has to be increased as well to achieve better results. With a physical understanding of wind and the idea behind our spatio-temporal model in mind, this team play of both parameters is a reasonable behavior. Figure 7.5 shows a comparison of the six algorithms with the best setting for μ for each algorithm.

While these results show that a better prediction is possible with a larger number of considered features, we address an important issue with the selection of machine learning models in another experiment: The practitioner needs to know *a priori* which parameter values are the best for the prediction. Therefore, the parameter settings are selected on a validation set or using cross-validation technique. The question for the given task is: Can the optimal number m of neighbor turbines be selected while training the model? If this is the case, the parameter setting yielding the best training error also yields a very good if not the best prediction error on a test set. Like in Section 7.4, we employ a heterogeneous ensemble with 256 DT predictors using 10,000 training samples and 64 SVR predictors using 1,000 training samples with a fixed feature window of $\mu = 10$.

Figure 7.6 shows the MSE depending on the number $m \in \{10, \ldots, 100\}$ of neighboring turbines used for a wind turbine near Las Vegas. It can be seen that both error measurements show a similar behavior depending on m. In particular, the setting $m = 40$ yields the best error for both training and test error. The optimal found training and test errors found on five turbines are shown in Table 7.4. For four of the five, the value of m yielding the best training error corresponds to the same setting for the best test error. For the turbine near Casper, the best test error is achieved by a smaller number of neighbor turbines, but when using the best setting chosen by training error, the test error is still very low with 9.98. Therefore, we suggest including the number of neighbor turbines into the algorithms' parameter search.

Table 7.4. Best settings for number m of used neighbor turbines selected by best training and test error.

Turbine	m	E_{train}	m	E_{test}
Casper	60	4.93	40	9.74
Hesperia	60	3.81	60	6.91
Las Vegas	40	4.69	40	9.57
Reno	20	6.16	20	13.33
Vantage	50	2.83	50	6.09

7.6 Power Prediction for Wind Parks

The question comes up if the proposed prediction model can give good predictions for wind parks, too. Treiber, Heinermann, and Kramer [108] investigated the use of machine learning models for wind park power prediction while employing different aggregation settings. They found that predicting the overall power output of a whole wind park often yields a better forecast error than predicting the outputs of the single turbines and summing them up. In the following, we compare our heterogeneous ensemble approach to state-of-the-art SVR used for a prediction for wind parks. Each of the five tested wind parks consists of a center turbine and 10 neighboring turbines.

Let $p_i(t)$ be the measurement of a turbine i at a time t, and $2 \leq i \leq (m+1)$ the indices of the m neighboring turbines. For a center turbine with index 1 we define a pattern-label-pair $(\mathbf{x}, y)$ for a given time t_0 as

$$\begin{pmatrix} p_1(t_0 - \mu) & \dots & p_1(t_0) \\ \dots & \dots & \dots \\ p_{(m+1)}(t_0 - \mu) & \dots & p_{(m+1)}(t_0) \end{pmatrix} \rightarrow \sum_{i=0}^{m+1} p_i(t_0 + \Delta t). \tag{7.2}$$

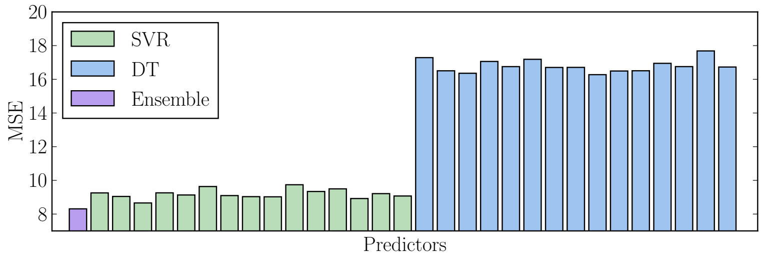
Again, the parameters for SVR are sought via three-fold cross-validation. A RBF-Kernel is employed with bandwidth $\sigma \in \{1e-5, 1e-4, \dots, 1\}$ and $C \in \{1e; 10; \dots; 10,000\}$. The ensemble regressor is built up from 256 decision trees using $10,000$ training samples and 64 SVR estimators using $2,000$ training samples. The patterns are constructed with $\mu = 6$ and $\Delta t = 3$. The results of the comparison are shown in Table 7.5. One can see that the heterogeneous ensemble regressor outperforms the optimized SVR model for each of the five parks tested. The lower test errors suggest that the use of a heterogeneous ensemble model can be a good choice for the overall power prediction of wind parks.

Table 7.5. Power predictions for the sum of five test wind parks. The test error (MSE) is given.

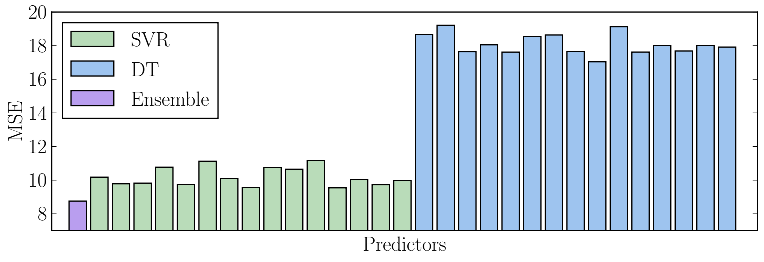
Center Turbine	SVR with CV	Heterogeneous Ensemble
Casper	888.09	**870.86**
Hesperia	785.09	**762.76**
Las Vegas	641.10	**633.12**
Reno	775.97	**765.61**
Vantage	592.34	**588.60**

7.7 Error of the Ensemble Members

For two NREL test turbines, we trained again an ensemble consisting of SVR and DT predictors, using first half of 2004 as training dataset and second half of 2004 as test dataset. For the sake of a clear arrangement, we chose $T = 30$ with 15 SVRs using $1,000$ samples and 15 DTs using $10,000$ samples. In Figure 7.7, we show the overall prediction error of the ensemble and the errors of the single ensemble members. It can be seen that for both SVR and DT, the errors among the members are different. Each single member gives a different prediction for one test pattern and therefore the MSE is different. The SVR errors are lower than the ones of the DTs. The overall error of the ensemble, plotted at the very left is a very interesting result. Both for Cheyenne and Lancaster, the ensemble error is lower than each members error.



(a) Cheyenne



(b) Lancaster

Fig. 7.7. Example of the ensemble prediction error and the errors of he single members.

7.8 Experiment on AEMO Dataset

In addition to the preceding experiments, we conducted further experiments on the Australian AEMO dataset. It offers 5-minute power output data, see Appendix A. We show that using the heterogeneous ensemble can increase the prediction performance for this use case as well. In contrast to the NREL experiments, only the time series of the target turbine itself is used.

In Table 7.6, we give a comparison of predictions for five wind farms in South Australia. Again, the k-NN and SVR are trained using a 5-fold-CV because optimal k or C and σ must be used for each new dataset. In contrast, the ensemble dos not need a cross-validation. The main factor for the prediction performance is the number of estimators used. Further, for the composition of the ensemble and parameters of the single estimators, a good parameter choice can be transferred from the preceding experiments – this is not the case for

SVR and k-NN. The dataset used consists of $105,120$ measurements for each

Table 7.6. Comparison (MSE) of k-NN, SVR, and heterogeneous ensemble on five wind farms from the AEMO dataset. The best MSE for each turbine is highlighted.

Turbine	Capacity	k-NN	SVR	ENS
Capital	140MW	231.18	225.55	**225.13**
Cullerin	30MW	275.68	269.88	**267.87**
Gunning	47MW	59.47	58.40	**57.99**
Wooodlawn	48MW	38.85	37.92	**37.88**
Canunda	46MW	33.05	32.85	**32.45**

wind farm in year 2011, of which $\frac{1}{5}$ is used. For all five test wind farms, the heterogeneous ensemble, consisting of 64 SVR and 256 DT regressors with $1,000$ and $10,000$ samples, respectively, outperforms the CV-optimized state-of-the-art methods. This shows the superiority of the ensemble approach because it yields a decreased error out-of-the box with only few minutes of training time while the SVR cross-validation takes between two and five hours for each wind farm on an Intel Core i3 wth 4×3GHz processor.

7.9 Conclusions

Wind power can only be integrated into the power grid with a forecast model that yields a reliable prediction error and is efficient enough to compute the predictions in a reasonable time. After analyzing different types of ensemble predictors, we propose a heterogeneous ensemble approach utilizing both DT and SVR. In our comprehensive experimental evaluation, we show that our approach yields better results within a shorter computation time than state-of-the-art machine learning algorithms. Compared to SVR, our heterogeneous ensemble approach yields improvements of up to 37%. The runtime can even be decreased: Our approach decreases the computation time for training by factors from $1.60\times$ to $8.78\times$. The trade-off between prediction performance and computation time can easily be managed by adapting the parameters like number of predictors, number of samples, and number of features used. Moreover, the number of neighbor turbines and past time steps can be increased to decrease the prediction error further. In the following chapter, we continue the idea of the tradeoff between accuracy and computation time by employing a multi-objective optimization technique.

8 Evolutionary Multi-Objective Optimization

In the preceding chapters, we presented a novel machine learning approach to short-term wind power prediction. The success of machine learning models highly depends on the algorithm choice and the parameter settings. Further, both aspects can have a negative impact on training and prediction runtime. Machine learning ensembles help to decrease the prediction error and possibly to achieve an acceptable computation time. A problem with the possible prediction models is the infinite number of possible combinations and choices. In this chapter, we show that ensemble models for wind power prediction can be balanced using evolutionary multi-objective optimization algorithms in order to find a good trade-off between runtime and prediction error.

Evolutionary computing is a class of biologically inspired algorithms for heuristic optimization. In the space of possible *solutions*, a randomized search is conducted with inspiration from evolutionary processes, i.e., selection, mutation, and recombination [26]. For multi-objective optimization problems (MOPs) with conflicting objectives, there exist special techniques to evolve a set of solutions that give the best balance between these objectives. These algorithm class is called EMOAs, which aim at approximating a *Pareto front* representing equally good solutions which we call *uncomparable.*

Why should one be interested in a Pareto-optimal solution? Consider three solutions for the prediction task. Solution 1 takes 100 seconds and yields a test error of 10.0. Solution 2 takes 1 second and yields a prediction error of 15.0. Solution 3 takes 200 seconds and yields a prediction error of 15.0 as well. So when the practitioner needs to decide which of these three solutions to choose, the choice between 1 and 2 would depend on his requirements. I.e., if the reduction of the prediction error is more important or the least possible runtime. The solution 3, however, would be the worst choice in each case. So we aim at optimizing all objectives with the question in mind which solutions offer the best tradeoff between the objectives. In our case it is thinkable to operate the wind power prediction for thousands of turbines on a high-performance data mining server and therefore each saving in computation time means a decreased requirement for expensive hardware.

Multi-objective optimization of machine learning models is an active research field. Using an evolutionary algorithm like the non-dominated sorting genetic algorithm NSGA-II by Deb *et al.* [23], yields a practical approach for the machine learning practitioner. The algorithm evolves a set of machine learning models from which one can choose his favorite model a posteriori. Furthermore, the approach requires much less evaluations than using a weighted single-objective approach. The success of EMOA based tuning of machine learning models has been shown by various researchers. A trade-off between accuracy and regularization of SVM models using NSGA-II has been presented by Mierswa [82]. Hu et al. [48] propose an evolutionary method for DT ensembles. In [85] we balanced accuracy and runtime of classification ensembles on artificial benchmark datasets using NSGA-II.

In this chapter, we combine the ensemble approach with the EMOA technique in order to give an optimization method for model selection in the field of wind power prediction. We show that by using the NSGA-II algorithm the balance between a low prediction error and a short computation time can be found. In our experimental analysis, we present three examples of ensemble models which are optimized with EMOA. First, we construct SVR ensembles [40] using estimators with different model complexity and therefore different runtimes and error behavior. Second, we employ the famous Random Forest [13] approach.

Finally, we construct heterogeneous ensembles [43] comprising a various number of SVR, Decision Tree and k-NN models.

This chapter is structured as follows. The framework of evolutionary computation is briefly introduced in Section 8.1, followed by the multi-objective optimization using NSGA-II in Section 8.2. Section 8.3 describes our experimental setup. The optimization of SVR ensembles is presented in Section 8.4. Section 8.5 comprises the experimental results for random forests and Section 8.6 for heterogeneous ensembles. A summary and conclusions are given in Section 8.8.

8.1 Evolutionary Computing

Evolutionary computing is a paradigm in computer science, which is inspired by the natural evolution. With evolution as a very powerful principle that can be found everywhere in nature, we obtain an abundant and efficient framework for problem solving [26]. There are various different variants of evolutionary algorithms (EAs). The basic idea is to invoke populations of individuals in an iterative manner, cf. Eiben and Smith [26]. Each individual consists of a chromosome which results in a fitness value. The chromosome is a point in the space of possible solutions and can be seen as genotype of the individual. In the taxonomy of evolution strategies, it consists of real-valued genes, i.e., $\mathbf{x} = (x_1, x_2, \ldots, x_N) \in \mathbb{R}^N$ [60]. For other algorithms, it can consist of binary genes. The fitness value is a measure of the quality of an individuals' performance, represented by an objective function with the chromosome as input. For a minimization problem, a fitness value as low as possible shall be achieved. The EA conducts a competition between the individuals. The basic scheme is visualized as a flow chart in Figure 8.1. At first, an initial population $\mathcal{P}$ is created and the fitness value for each individual is evaluated by computing the objectives functions. Then, in each iteration a new population of individuals is evolved. Here, different mechanisms of evolution can be employed. From a population, feasible parents are selected for reproduction, based on their fitness values.

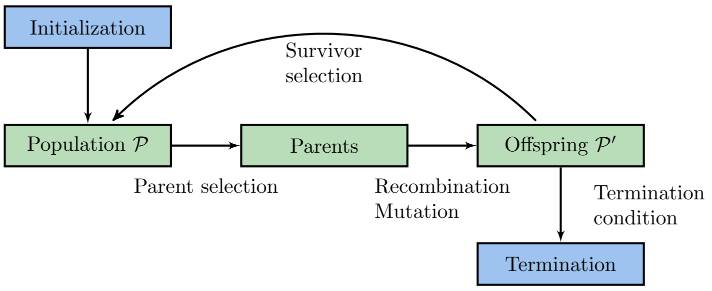


Fig. 8.1. Scheme of an evolutionary algorithm [26].

The reproduction step is done by mutation and recombination. For mutation, the individuals' genes are modified by random numbers. For example, the Gaussian mutation operator $x' = x + \mathcal{N}(0, \sigma)$ is a popular choice. Since the so-called mutation rate σ is an important parameter for the evolutionary optimization and has a great influence on the success, the correct choice of σ is part of current research. It can be set to a fixed value but also vary over time. In particular, there exist successful approaches of adaptive control of σ [60, 93]. However, due to the scope of this work, we do not go into detail here.

The recombination step is inspired by mating. A recombination operator uses a number of individuals and combines the single genes of their chromosomes. The idea is to combine the genetic material of two or more successful individuals for generating an possibly even more successful offspring. In the selection step, the offspring population $\mathcal{P}'$ of size λ is evolved using a selection operator. The $(\mu + \lambda)$ selects the best μ as parental population $\mathcal{P}$. The (μ, λ)-selection does not consider individuals from $\mathcal{P}$ itself to be in the offspring generation.

The paradigm of evolutionary computing has been applied successfully to a wide range of complex real-world applications like signal and image processing, computer vision, pattern recognition, industrial control, telecommunication, scheduling and timetabling, and aerospace engineering, see Cagnoni *et al.* [17]. There are also very fruitful applications in machine learning. An example for the evolutionary wind power prediction is given by Treiber *et al.* [109]. They conduct the selection of considered neighbor turbines' features for a spatio-temporal time series prediction and show that both (1+1)-EA and state-of-the-art CMA-ES yield good results.

8.2 NSGA-II

EAs are a successful method for solving MOPs. In contrast to a single-objective problem, the quality of a solution is defined by several objectives $f_1, \ldots, f_m$. It is common that these objectives are conflictive and therefore a compromise has to be made. For instance, if one wants to buy a new car, a possible objective is a low price. However, two other objectives could be that it is very fast or has a preferably big trunk. These objectives can be very conflictive because usually, faster cars are more expensive and do not necessarily have a big trunk. An alternative to a multi-objective formulation of a MOP is the representation using a single fitness value, which is computed using weights, and use a single-objective optimizer. However, this method is cumbersome because the user's preferences need to be known a priori.

In the field of EMOAs, the concept of *dominance* is very important. With two solutions $\mathbf{x}$ and $\mathbf{x}'$ given for a minimization problem, $\mathbf{x}$ dominates $\mathbf{x}'$, written as

$$\mathbf{x} \prec \mathbf{x}' \tag{8.1}$$

if it is better in all objectives $f_1, \ldots, f_m$. With conflicting objectives, there usually exists no solution that dominates all others. A solution is *non-dominated* if it is not dominated by any other. This means its quality w.r.t one objective cannot be increased without decreasing the others. We seek for a set of non-dominated solutions which is called the *Pareto set*. The corresponding objective values of the Pareto set define the Pareto front

$$\mathcal{PF} = \{\mathbf{f}(\mathbf{x}^*) \in \mathbb{R}^m | \nexists \mathbf{x} \in \mathbb{R}^N : \mathbf{x} \prec \mathbf{x}^*\}. \tag{8.2}$$

There exist various different algorithms for solving MOPs. One of the most famous ones is the Non-Dominated Sorting Genetic Algorithm NSGA-II by Deb *et al.* [23], which we use in this work. It is based on non-dominated sorting, which computes a domination rank for each solution. The NSGA-II employs the crowding distance metric, which is for one solution computed as the difference of the single fitness values of the neighbor solutions. A large value means, there are only few other solutions in the neighborhood of the solution. In the evolutionary computation, each new population is obtained by using a $(\mu + \lambda)$ survivor strategy with $\mu = \lambda$. The parent and offspring populations are merged and the new Pareto front is evolved based on dominance rank and crowding distance using a tournament operator [26]. Through elitism and diversity maintenance, the NSGA-II strategy has been shown to perform very successful.

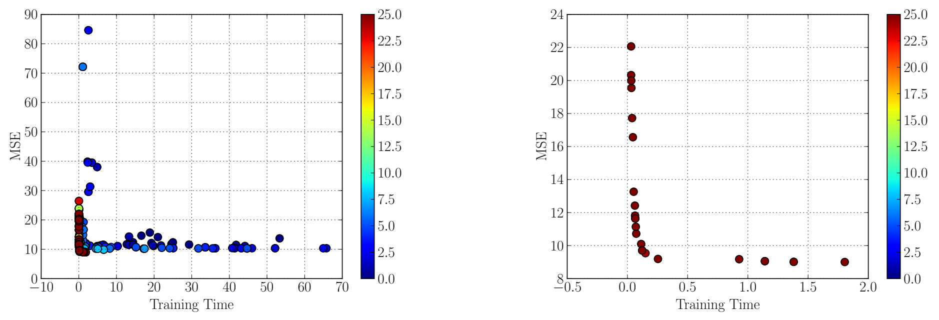
8.3 Experimental Setup

Objective of this chapter is a proof of concept for the multi-objective optimization of ensembles for wind power prediction. In the chapters 5, 6, and 7, we presented different methods that yield an improved prediction error and showed that different solutions offer different runtime behavior for both training and testing. In this chapter we start with the optimization of SVR ensembles, followed by RF and finally heterogeneous ensembles which are especially well-suited for the presented approach. For all three investigated types of model, we describe a method of representing a reasonable subset of ensemble models while offering maximum simplicity. Another result is that this method of parameter tuning can offer valuable insights in the landscape of possible solutions.

For three test turbines we use the data from the year 2004 of the NREL dataset. To ensure a reliable experiment, the data is split with the 5-fold cross-validation method. The optimization process is conducted with the NSGA-II algorithm using SBX-recombination and polynomial mutation. In each experiment, we use population size $\mu = 20$. The termination condition is the maximum of 500 problem evaluations, resulting in 25 generations. The objectives are the training time and the CV error (MSE). The results of the optimization are visualized as scatter plot with training time on the x-axis, CV error on the y-axis and the generation number as color.

8.4 SVR Ensembles

Support vector techniques belong to the most successful methods in machine learning and have been shown to work well for wind power prediction. In Chapter 5, we have shown the superiority of ensembles using SVR as base predictors over single SVR models. First, the prediction error can be decreased. Second, the runtime can be decreased utilizing the runtime behavior of the SVR algorithm and an divide & conquer approach. The best performing models were created using a random parameter choice of C and σ for each SVR model. The reason for this is the diversity amongst the predictors.



(a) Lancaster (b) Lancaster (Generation 25)

Fig. 8.2. SVR ensemble solutions evolved by NSGA-II in 25 generations.

In this chapter we give a proof-of-concept that multi-objective optimization is a feasible approach for the model selection. This allows for balancing the runtime and the prediction error. We choose a simple approach for constructing the ensemble, resulting in a chromosome with six genes. The bagging ensemble consists of $T_1 + T_2 + T_3$ SVR models with different parameter settings and uniform weighting, using the following parameter settings:

- T_1 models use samples of size S_1 and parameters $C = 1.0$ and $\sigma = 1.0$
- T_2 models using samples of size S_2 and parameters $C = 100.0$ and $\sigma = 10^{-3}$
- T_3 models using samples of size S_3 and parameters $C = 10,000.0$ and $\sigma = 10^{-5}$

The basic idea is to use models of different complexity, which can be provided by C and σ choices as well as sample sizes S_i. The number of possible parameter settings for SVR ensembles are countless and therefore we only demonstrate the feasibility of the approach with this simple subset. Each T_i of the initial population is initialized with a random number from $\{1, \ldots, 50\}$, S_i from $\{1, \ldots, 2000\}$.

The results for a Wind Turbine near Lancaster are shown in Figure 8.2. While Figure 8.2 (a) shows all evaluated solutions, Figure 8.2 (b) shows the evolved Pareto front of the 25th generation. It can be seen that a lot of different solutions with different error behavior and runtimes are generated. In each iteration, the solutions are more and more located in the left bottom corner, moving towards a desirable solution with a good tradeoff. The solutions of the final generation, seen in Figure 8.2 (b) show a useful Pareto front for the machine learning practitioner: Only solutions with feasible objective values are left and each solution shows a good tradeoff. Now the practitioner can select a solution based on his preferences.

In Table 8.1, some selected solutions are shown. For both training time (t) and CV error (E), the minimum, median, maximum from the 25th generation are sought and the belonging solutions depicted. While an optimal training time or optimal error give only a good result for that very objective, the other objective is falling behind. The same is true for the minimum an maximum objective valued solutions from all generations, which are shown in the last four rows: One of the objectives can score very poor, although the other one is very good. These results suggest that the multi-objective optimization is a very useful way to evolve desirable solutions with the best tradeoff.

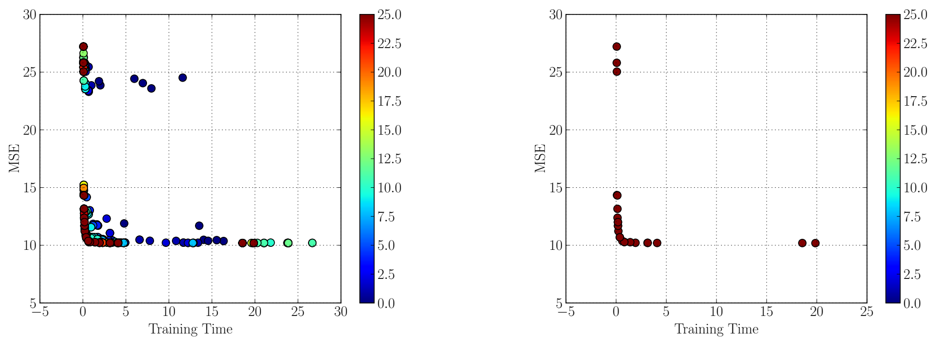
Table 8.1. Selected SVR ensemble solutions for Lancaster.

Pareto-Set	T_1	S_1	T_2	S_2	T_3	S_3	t	E
min(t)	0	0	0	191	17	41	0.03	22.06
median(t)	0	0	12	213	18	55	0.14	9.87
max(t)	1	0	18	733	26	444	1.83	9.10
min(E):	0	0	18	789	26	399	1.48	9.03
median(E):	0	0	12	213	18	55	0.14	9.87
max(E):	0	0	0	191	17	41	0.03	22.06

All Solutions	T_1	S_1	T_2	S_2	T_3	S_3	t	E
min(t):	0	0	0	191	17	41	0.03	22.06
max(t):	44	1967	43	1653	48	1741	65.80	10.35
min(E):	0	0	18	767	29	438	1.80	9.01
max(E):	31	1	0	792	18	368	2.55	84.63

8.5 Random Forest

As shown before, the Random Forest method is quite successful and efficient for giving a good wind power forecast. Besides the number of estimators used, there are some DT-specific parameters like the tree depth or the maximum number of features considered for the random splits. These have impact on both the prediction accuracy and the computation time needed. In the following experiment, we present the usefulness of a multi-objective optimization. The results are very similar to the SVR optimization. We tune the number of estimators which is initialized with a random number from $\{1, \ldots, 500\}$, the maximum features used for the random split (initially from $\{1 \ldots 10\}$) and the maximum depth of the trees (initially from $\{1 \ldots 5\}$). The results for the test wind turbine near Lancaster are shown in Figure 8.3. By tuning the parameters and in particular adding more estimators to the ensemble, the prediction performance can be improved. The solutions taking more time but yielding similar or worse results are rejected in every iteration of the evolutionary optimization. In Table 8.2 we show some



(a) Lancaster (b) Lancaster (Generation 25)

Fig. 8.3. RF solutions evolved by NSGA-II in 25 generations.

important solutions from the Pareto set and from all evaluated individuals. It is possible to select a solution that performs really fast but gives a large error. In contrast to the min(t) solution, for a relatively good prediction error it is not necessary to spend the largest training time. The median(t) and median(E) solutions show that it is sufficient to invest only little more time to achieve a decent prediction error.

8.6 Heterogeneous Ensembles

As ensembles benefit from the diversity of predictors, we showed that ensembles employing different prediction algorithms perform well and offer a moderate requirement for computation time for both training and testing. In the following experiment, we give an example for the optimization using NSGA-II for heterogeneous ensemble predictors for three wind turbines. The number of possible models is infinite and we choose again a simple subset for proof-of-concept. In the experiment, the predictors consist of $T_{SVR} + T_{DT} + T_{KNN}$ models and T_i are tuned with the evolutionary optimization approach along with the belonging sample sizes S_{SVR}, S_{DT}, S_{KNN}. Each T_i of the initial population is initialized with a random number from $\{1, \ldots, 100\}$, S_i from $\{1, \ldots, 2000\}$. The SVR models

Table 8.2. Selected RF solutions for Lancaster.

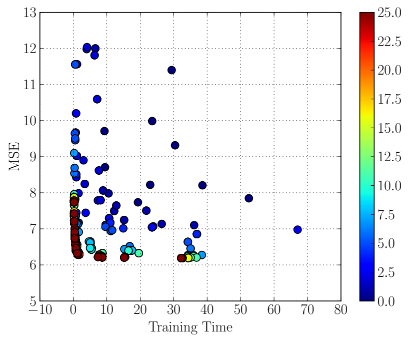
Pareto-Set	`n_estimators`	`max_features`	`max_depth`	t	E
min(t)	10	1	1	0.07	26.31
median(t)	10	4	4	0.39	10.59
max(t)	296	7	4	18.24	10.21
min(E)	296	7	4	18.24	10.21
median(E)	10	4	4	0.40	10.65
max(E)	10	1	1	0.07	26.31

All Solutions	`n_estimators`	`max_features`	`max_depth`	t	E
min(t)	10	1	1	0.06	27.21
max(t)	415	9	4	33.65	10.24
min(E)	315	7	4	19.86	10.19
max(E)	10	1	1	0.07	27.23

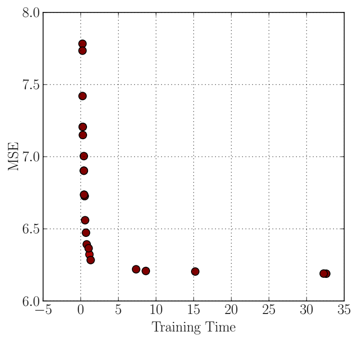
use an RBF-Kernel with $\sigma = 10^{-4}$ and $C = 1,000$. k-NN uses $k = 5$ nearest neighbors. Of course, in future work these parameters can be optimized with the EMOA as well.

Figure 8.4 shows the results for three test turbines near Palm Springs, Lancaster, and Las Vegas. For each turbine, all 500 solutions depicted in the left plot show various different possible solutions. The right plots shows the Pareto from evolved in the 25^{th} generation. It can be seen that these are the preferred solutions for the machine learning practitioner from which he now can choose based on his preferences. For turbine Lancaster, we show the parameters and objective functions' values for some important solutions in Table 8.3. It can be seen that balancing runtime and prediction error can be solved with the EMOA technique and a heterogeneous composition of the ensemble can be beneficial.

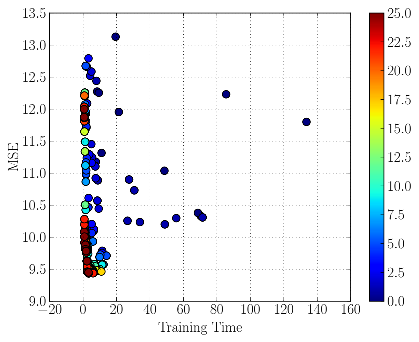
In Figure 8.5, we give a comparison of the evolved SVR ensembles, Random Forests, and heterogeneous ensembles for the test turbine near Lancaster. Although not a completly fair comparison because of some simplified assumptions, the results demonstrate that there exist very good solutions for all three algo-



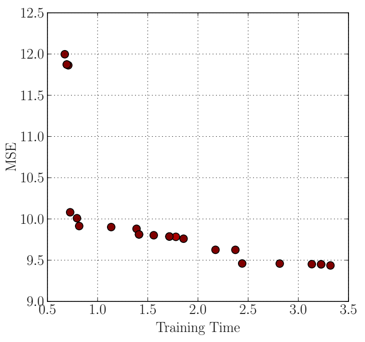
(a) Palm Springs



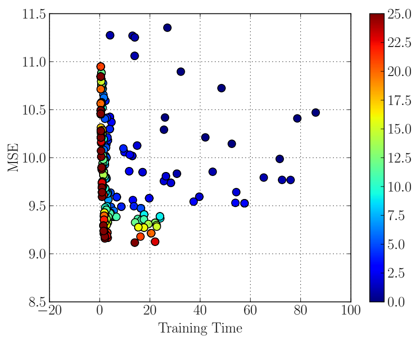
(b) Palm Springs (Generation 25)



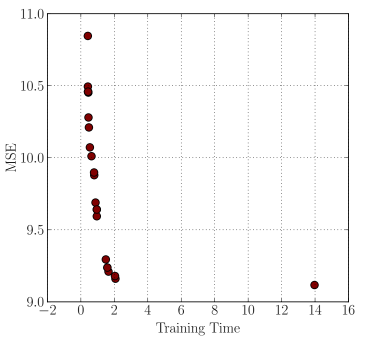
(c) Lancaster



(d) Lancaster (Generation 25)



(e) Las Vegas



(f) Las Vegas (Generation 25)

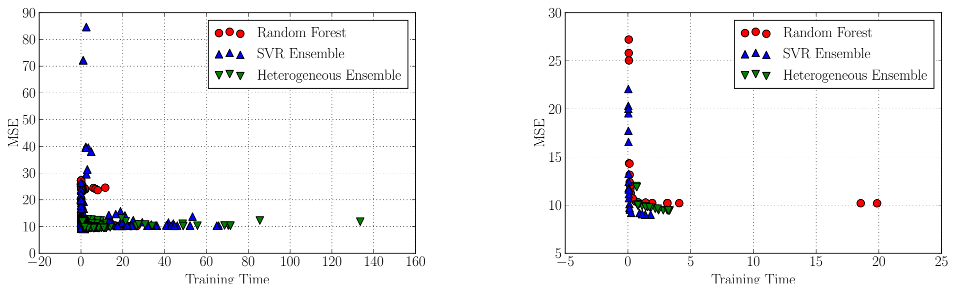
Fig. 8.4. Heterogeneous ensemble solutions evolved by NSGA-II in 25 generations.

Table 8.3. Selected heterogeneous ensemble solutions for Lancaster.

Pareto-Set	T_{SVR}	S_{SVR}	T_{DT}	S_{DT}	T_{KNN}	S_{KNN}	t	E
min(t)	3	121	0	223	100	1753	0.79	10.19
median(t)	3	156	3	332	25	1836	1.81	11.95
max(t)	48	124	67	242	108	1728	3.84	9.48
min(E)	55	147	6	224	111	1884	3.79	9.47
median(E)	3	125	0	226	108	80	1.16	10.11
max(E)	2	117	3	224	25	70	0.93	12.01

All Solutions	T_{SVR}	S_{SVR}	T_{DT}	S_{DT}	T_{KNN}	S_{KNN}	t	E
min(t)	3	117	2	224	25	60	0.67	12.00
max(t)	82	1379	79	1972	20	398	133.55	11.80
min(E)	56	124	116	226	118	1755	3.32	9.44
max(E)	10	1698	22	839	7	1523	19.44	13.13

rithms. The found SVR ensembles yield the best prediction error in the shortest time. However, the sets of evolved solutions for both RF and SVR ensembles contain solutions with relatively a poor prediction error. In contrast, the Pareto front evolved for the heterogeneous ensembles only contains solutions that provide a good balance between the two objectives.

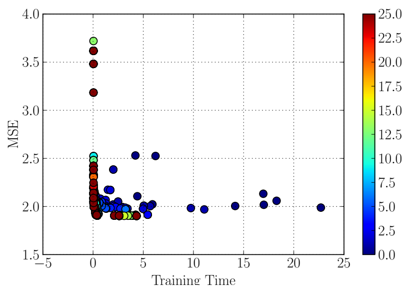


(a) Lancaster (b) Lancaster (Generation 25)

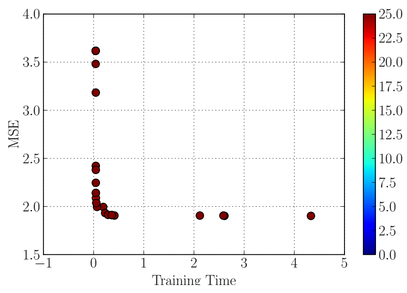
Fig. 8.5. SVR (blue), RF (red) and heterogeneous ensemble (green) solutions evolved by NSGA-II in 25 generations.

8.7 Experiments on DWD Dataset

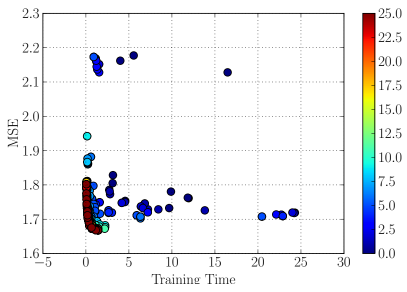
The DWD dataset consists of wind speed measurements from weather stations in Germany, see Appendix A. Although only hourly data is available, a prediction is possible. The data from the whole year 2015 is used, resulting in 8760 measurements. We use feature window of $\mu = 6h$ and a forecast horizon $\Delta t = 3h$. With the same experimental settings and ensemble parameters as in Section 8.6, we show that the multi-objective optimization yields good results for this use case, too. Figure 8.6 shows the results two weather stations in Bremen and Gütersloh.



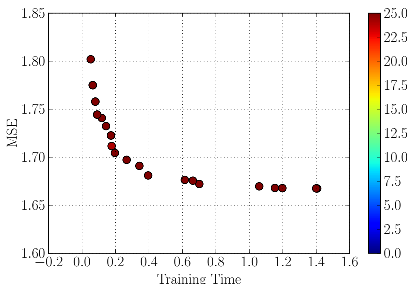
(a) Bremen



(b) Bremen (Generation 25)



(c) Gütersloh



(d) Gütersloh (Generation 25)

Fig. 8.6. Heterogeneous ensemble solutions evolved by NSGA-II in 25 generations (DWD dataset)

Although the training times are relatively low for all evolved solutions because of the relatively small training data set, the results show that a multi-objective optimization with non dominated sorting genetic algorithm II (NSGA-II) yields good models and a balanced choice of heterogeneous ensembles is possible.

8.8 Conclusion

In this chapter we have shown the usefulness of an multi-objective optimization using EA techniques. When dealing with supervised learning, not only the prediction performance is objective to parameter tuning, but also the reduction of huge computation times to a feasible level. Here, machine learning ensembles render the possibility of finding the balance between the two objectives for the practitioner. Using biologically-inspired heuristic search, exploring the tremendous search space of possible solutions can be done efficiently. The result is a Pareto-front of preferable solutions from which the practitioner can select one solution based on his preferences. With the experiments, we give a proof-of-concept. However, the possibilities of the evolutionary multi-objective optimization can be further exploited in future work, considering ensemble weights, neighbor turbine selection, and the algorithms' parameter choice as a wider range of objective variables. This increases the search space but also yields new possibilities for finding better solutions w.r.t the given objectives. The application of EMOAs to the field of prediction could also be helpful when aiming at hybrid estimators employing both data-driven and meteorological models.

Part IV

Summary

9 Summary and Outlook

To achieve the goal of integrating wind power into the power grid, we need a prediction method that computes precise and reliable forecasts. Another important requirement for practical use is an efficient computation. In this thesis, we present a framework for short-term wind power prediction using ensemble models to meet these requirements. This chapter is structured as follows. In Section 9.1, we summarize the research contributions. The conclusion is given in Section 9.2. In Section 9.3, we give an outlook on possible future research.

9.1 Contributions of this Thesis

A good alternative to the well-known machine learning algorithms is the use of ensembles, i.e. combining several basic models to an ensemble model. The prediction error can be decreased and ensembles offer a short computation time. We investigate four different aspects of ensemble prediction for wind power, which are summarized in the following.

1. SVR Ensembles (Chapter 5)

We propose the use of SVR ensembles for wind power prediction. The main objective is an improvement of the forecast quality compared to state-of-the-art machine learning algorithms. The second objective is the reduction of computation time, which is important for a practical, scalable application of machine learning methods in the industry. Instead of using one single support vector regressor, we train a number of weak SVR regressors with a weighted bagging approach.

We investigate different parameter choice strategies as well as different weighting methods. A random parameter choice combined with an error-based weighting of the ensemble members achieves the best results. Further, the runtime of the random approach without optimization is very short compared to grid search based parameter optimization.

The runtime of the proposed algorithm mainly depends on the number T of estimators and the size S of the training sample each estimator is trained on. By using relatively small sample sizes for the single estimators, the $\mathcal{O}(N^3)$ runtime complexity of the SVR algorithm is exploited. Therefore, the proposed algorithm yields a very efficient computation. The algorithm is embarrassingly parallel and we derived a multicore implementation in PYTHON that benefits directly from the number of CPU cores available.

The prediction error of the SVR ensemble depends on T and S. Given large enough sample size and number of estimators, a very low prediction error can be achieved. On the other hand, both T and S should be chosen as small as possible for reduction of the runtime required. The trade-off between the optimization of prediction error and a short computation time is also continued in the chapters 7 and 8.

Finally, a comparison to SVR is given. Compared to the state-of-the-art, SVR ensembles perform better on five out of five test turbines. Improvements of MSE from 1.9% up to 33% are achieved. The training times are reduced to a value between 10% and 50% of the SVR algorithm. Another experiment suggests that SVR ensembles are better suited to an increased forecast horizon than SVR. The experimental results shows hat a precise, robust, and very efficient prediction can be computed using our proposed approach.

2. Combination of Speed and Power Features (Chapter 6)

The use of appropriate input features is very important for the success of machine learning techniques. We analyze the behavior of various regressors, which are trained using patterns composed of different available features. For predicting the power output, we employ as features power output measurements, wind speed measurements, and computed differences of two successive measurements of each. On ten test turbines, we compare k-NN, SVR, and RF algorithms that turned out to be very successful in various applications. Both RFs and SVRs perform very well using power and speed features. The RF algorithm yields a slightly lower MSE than SVR. In the experimental results, it can be seen that the computed differences between two successive time steps help to substantially improve the prediction.

Next, we analyze the behavior of the three algorithms when all available features are used as one big pattern. In the most cases, the RF approach performs best w.r.t. both CV error and test error. For some turbines, the SVR yields better results. The k-NN algorithm always performs worse.

The further experiments focus on the question if combinations of different features used and combinations of different regression algorithms can help to improve the prediction error further. Two different approaches are analyzed. First, RF and SVR predictors that are based on one type of features are combined: One regressor is trained on the power time series and one is trained on the speed time series. While the combination can help to decrease the prediction error for all ten test turbines, there is no clear answer which combination of the predictors is the best. The second approach analyzed combines one RF and one SVR predictor based on all available features with a mixture coefficient α. For eight out of ten test turbines, this approach achieves the lowest prediction error compared to the first combination approach. Summing up the results of the chapter 6, we strongly recommend the use of all available features and also consider different prediction algorithms as well as combinations of these. The idea of heterogeneous ensembles is continued in Chapter 7 and Chapter 8.

3. Heterogeneous Ensembles for Wind Power Prediction (Chapter 7)

Since diversity among the ensemble members is crucial for the accuracy of the ensemble, we propose the use of heterogeneous ensemble predictors consisting of different types of base predictors for wind power prediction. We first analyze ensembles consisting of k-NN, SVR, and DT. While DT and SVR perform well, the k-NN algorithm seems not very well-suited for the ensemble method.

Going further, we analyze different ensembles employing heterogeneous estimators relying on different base algorithms. Our comprehensive experimental results show that a mixed ensemble consisting of a number of DTs and a number of SVRs yields better results than the analyzed homogeneous predictors while offering a very good runtime behavior.

A similar approach with a slightly different point of view is the combination of two ensemble models. Here, we combine one SVR ensemble with one DT ensemble. For the two ensemble members, the number T of estimators employed and the sample size S must be chosen and affects both the prediction error and the runtime. We show that the space of possible solutions offers a wide range of possibilities and enables to select a parameter setting that results in a good balance between the objectives of error optimization and decreasing the runtime.

Compared to state-of-the-art machine learning techniques, our approach renders better results within a shorter computation time. Our heterogeneous ensemble approach yields improvements of up to 37% compared to SVR and speed-ups for training time by factors from $1.60\times$ to $8.78\times$.

In another experiment, we show that the proposed heterogeneous ensembles are very well-suited for using large numbers of neighboring turbines and past measurements and improve the prediction performance. While the majority of our experimental studies focusses on the prediction for single turbines, we show that a good prediction for wind parks is possible, too.

The application of heterogeneous ensembles to the field of wind power prediction can help to improve both the prediction accuracy and the computation time requirements. Therefore, we encourage the machine learning practitioner to stay open for different algorithms and give combinations of these a try.

4. Evolutionary Multi-Objective Optimization (Chapter 8)

The success of machine learning models highly depends on the choice of algorithm and the parameter settings, which are difficult tasks. These aspects also influence the runtime behavior. Since we have to deal with a vast number of possible combinations and choices for model selection, we employ a nature-inspired heuristic. We show that ensemble models for wind power prediction can be optimized and selected using the EMOA algorithm NSGA-II in order to find the best trade-off between runtime and prediction error.

In the experimental evaluation, SVR ensembles, RFs, and heterogeneous ensembles are balanced with the EMOA technique. It can be seen that a Pareto front is evolved that offers various solutions that render a good compromise between the two objectives. The practitioner can choose from these solutions based on his preferences. For instance, a very short runtime can be preferred with the drawback of a slightly worse prediction error or vice versa. The practitioner could also select a solution which offers a moderate runtime and a decent prediction error. These properties are very desirable for the application in different scenarios and therefore the results are highly relevant. We suggest using a machine learning ensemble approach combined with multi-objective optimization because both the solution quality and the optimization efficiency are well-suited for a practical application.

9.2 Conclusion

The result of this thesis is a powerful and comprehensive framework for precise and efficient wind power prediction using machine learning ensembles. The research of the thesis is highly relevant because nowadays good forecasting methods are required and computation time is a limited resource for practical applications. SVR ensembles and RF offer a good prediction accuracy for a data-driven prediction of the power output to expect. The use of appropriate features helps to improve the prediction. By employing heterogeneous ensemble predictors, both the prediction accuracy can be improved and the required computation time can be decreased.

The balancing of the two objectives can be done with evolutionary optimization, which offers an appealing approach for the multi-objective optimization problem at hand.

9.3 Future Work

In this chapter, we briefly discuss further aspects that should be considered for detailed research in future work. We sketch our ideas that are out of scope of this thesis but require further research.

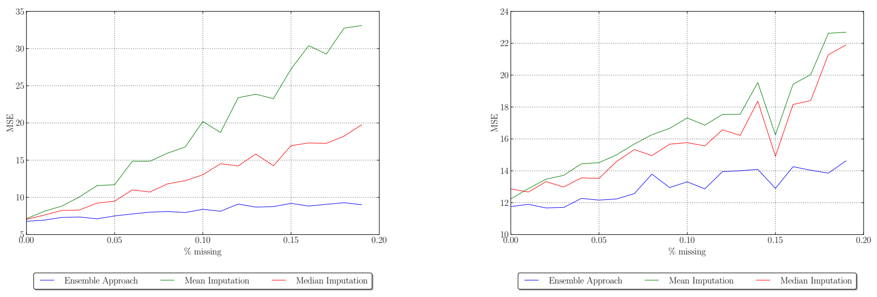
9.3.1 Handling of Missing Values

Real world data is often incomplete and some values are missing. In the field of wind power prediction with machine learning, one has to face the problem of missing values because of dropout, malfunction or corruption of sensors. When using the spatio-temporal regression model, the missing values would stop the model from working.

The problem of missing data is usually solved by using imputation methods [4, 89]. The imputation method is looking at the available values and computes the most likely value for the missing values. For the use during regression time, the commonly used imputation approach has several drawbacks. If a sophisticated imputation algorithm is employed, additional computation time is needed. For both imputation and supervised learning the algorithms and their parameters have to be chosen well to maximize prediction accuracy.

Saar-Tsechansky and Provost [95] investigate the use of reduced-models approaches for classification, which process the patterns with missing values directly without imputation. The focus is on decision tree based algorithms. One could also provide a more straightforward approach: We assume that the ensemble members do not tolerate missing values by themselves and want to show that the ensemble technique can help to increase accuracy when missing values occur. While imputation methods compute a likely value for the missing features by using the available features, we propose to use only the available features for the prediction itself while exploiting the ensemble approach for doing so. We suggest to train ensemble regression models in a way that they are still working well when data is missing. Objective of the research is to reduce the prediction error under these adverse conditions as far as possible while providing a practical implementation. Since ensembles employ models based on different samples or different feature subsets, the ensemble does not necessarily rely on all features. Further, an ensemble can use only a subset of the available estimators for the prediction. The method can easily be extended to special situations in the spatio-temporal regression of wind power. For example, when a neighbor turbine drops out or sensors fail, a model without that very turbine can be employed if available.

One can train a regression ensemble consisting of a large number of arbitrarily chosen weak predictors, all of which could use different feature subsets. When a pattern with missing values is used as test pattern, the ensemble models selects on the fly which estimators to use for that particular pattern of missing features. Because *a priori* you cannot know which combinations of missing features appear, every possible one has to be considered during training. For a dimensionality d and the number m of missing values, the number of possible combinations is given by $\sum_{m=1}^{d} \binom{d}{m}$. For instance, for $d = 30$, there are more than one billion combinations. Obviously, only a small subset of these can actually be used, considering time and memory limitations. However, with the following design goals in mind, for each combination of missing values a prediction can be given: We must be able to give a prediction result for every combination of missing values. By employing estimators that use only very few values available, we can ensure that at least one predictor could handle the data. On the other hand, estimators predictors using more features should increase the regression for many cases. Usually, ensembles with the random subspaces method use a large portion of the available features, determined by a fixed number (e.g. 90%). Instead, one can use a variable number of used features. A balanced choice with different amounts of features used would be the best, if a missing feature probability is not known beforehand.



(a) Tehachapi (b) Cheyenne

Fig. 9.1. Wind Power Prediction with the problem of missing values.

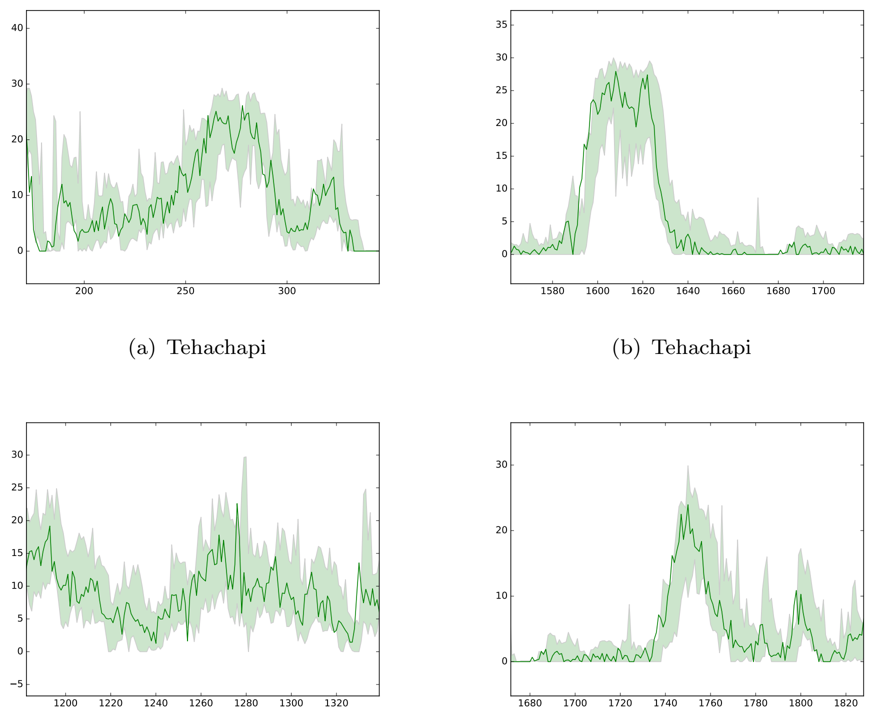
In a preliminary experiment, we show that the algorithm itself seems to work well. For a wind turbine near Tehachapi from the NREL dataset, we use $N = 1,000$ training and $N_t = 1,000$ test patterns with $m = 5$ neighbor turbines and $\mu = \Delta t = 6$, resulting in $d = 36$ dimensions. Like most imputation algorithms, we assume the values are *missing at random* (MAR) [39, 74]. Here, features are missing randomly and not depending on the actual value of the feature. Very simple and popular methods for imputation are taking the mean or median of the missing feature. Both are here used as comparison to the ensemble approach without imputation. Our approach trains $T = 200$ estimators. For each feature, one estimator is trained only using that particular feature. In a worst-case scenario, it is ensured that at least one or few estimators are available. Further, there are d estimators f_i with $i \in 1 \ldots 36$ trained which make use of all d features but feature i. The remaining $T - 2d$ estimators are trained using random features subspaces. This selection is a first straightforward attempt and shall be optimized in the future. The results are shown in Figure 9.1. It can be clearly seen that the approach outperforms the mean and median imputation methods which are used most widely. A reasonable prediction can be given even with higher rates of missing features. In the future, research can go into further detail and compare the method to more sophisticated algorithms.

9.3.2 Interval Prediction

In this thesis we present an ensemble method for wind power prediction that only gives one single value for a given point in time and a given horizon. For practical applications, it could be desirable to give a confidence value for the prediction or a prediction interval. The model computes not only one value but an upper bound and a lower bound as expected interval in which the real value lies with a certain probability, For instance, a requirement can be given that the bounds must be correct with a probability of 95%.

There exist some approaches in literature dealing with quantile regression using bagging. Heskes [45] presented a bagging approach providing prediction intervals or confidence intervals for ANN showing very good results even for small datasets. Lee and Yang [72] investigate equal-weighted and Bayesian Model Averaging weighted bagging methods for interval prediction on time series. Wager *et al.* [117] presented the Jackknife and the Infinitesimal Jackknife methods for estimating the Confidence Intervals for Random Forests. Meinshausen [81] proposed the method quantile regression forest (QRF), which considers the distribution of the results of the single decision trees instead of only their mean. These can be used for computing accurate prediction intervals. Further, the algorithm can detect outliers in the data. The approaches introduced in this thesis can be easily enhanced by one of these approaches.

Because an implementation of Meinshausens QRF idea is relatively straightforward, we conducted a preliminary experiment. For two NREL turbines, we compute an interval prediction using year 2004 for training, 5 neighbor turbines, and $\mu = \Delta t = 3$. The trained Random Forest consisted of 500 estimators, of which the distribution has been used for estimating the prediction intervals. The results are shown in Figure 9.2. These show that the computation of prediction intervals is possible with ensemble methods. Because of the relevance of the research problem, the application of these techniques and combination with the findings of this thesis could be a fruitful topic for future work.



(a) Tehachapi (b) Tehachapi

(c) Cheyenne (d) Cheyenne

Fig. 9.2. Quantile Regression Forests for wind power prediction yield accurate prediction intervals with a given percentile, here 90%.

9.3.3 Other Ensemble Techniques and Deep Learning

There are plenty of extensions of machine learning, neural networks, and ensemble techniques that can further improve the prediction performance. The possibilities for future work are virtually endless. While we stick to bagging methods, an obvious thing to do is the investigation of boosting algorithms. Further, employing other base estimators like ANN or time series methods could yield interesting research questions.

Another promising ensemble approach is given by Wolpert [122]: Stacked Generalization (Stacking) is a meta-learning model trained on the output labels of a set of estimators. Instead of computing an average of the outputs, the behavior of the estimators itself is also considered as helpful information depending on the situation. Here, we see a similarity to the deep learning paradigm. An interesting aspect of ensemble prediction for wind power that can be investigated is online learning. Because in operational service, new measurements are available every few minutes, the prediction system could be extended to the new data regularly. In ensemble models, old ensemble members could be deleted or a error-based weighting could help integrating new models.

Our research on SVR ensembles in Chapter 5 deals with the question how the ensemble techniques can improve the prediction accuracy of SVR. As second objective, we deal with the question how to reduce the computation time. A related approach is the Cascade SVM approach by Graf *et al.* [37], where an efficient parallel algorithm for training SVM classifiers is used. Like our SVR ensembles, the algorithm works in a divide-and-conquer manner by dividing the training dataset into partitions. For each partition, the support vectors are computed. In an iterative process, the partitions are merged pair-wise. From the found support vectors, the support vectors are computed again. This is done until all partitions are merged. The found support vectors can now be used for classification. The training process is faster and can be parallelized. Graf *et al.* [37] prove theoretically that the support vectors found are the same as with a standard SVM, however not with the best achievable runtime. As pointed out by Kramer [61], dimensionality reduction techniques can be integrated to the cascaded machine learning architecture for classification and regression. A good accuracy is achieved while the computation times are reduced further. We suggest that future work considers the similarities between ensembles and Cascade SVMs and possibly derives hybrid implementations. Looking into the other direction, one could also take our SVR ensemble approach and look into the support vectors of the ensemble members.

A recent trend of the last years is the paradigm of deep learning [7]. A long-held dream of researchers in the domain of neural networks was the training of deep neural networks (DNNs), which are ANNs with many layers. Until a few years ago, it was not possible to train DNNs because not enough processing power and a lack of the adequate algorithms. When training a deep neural network with standard backpropagation algorithms, the so-called vanishing gradient problem occurs [47]. While training the DNN, the weights of the neurons in the hidden layers cannot be updated in a reasonable way because their effect on the cost function gets lost because of the high number of layers and neurons. New algorithms have been proposed to overcome these difficulties. For instance, Deep Belief Networks [70] or Stacked Denoising Auto-Encoders [115], yield very good results. Beyond the task of supervised learning, deep architectures offer further applications and increased ability of generalization and therefore bring research closer to a general puropse artificial intelligence inspired by the brain. They are usually trained in a hierarchical way and employing layer-wise unsupervised training, yielding different problem representations on different layers. Crucial for success is the availability of large training datasets. For future work, it is thinkable to implement DNNs for wind power prediction. In particular, a GPU based implementation is desirable.

At this point, we want to point out that we see similarities between the ensemble way of thinking and deep architectures, which renders possible interesting research aspects to explore further. There exist hybrid algorithms yet. Tang proposes Deep Learning using linear SVMs [105]. Kontschieder *et al.* analyze the similarity between RF and DNNs and present a transformation that converts a RF estimator into a neural networks which can then be fine-tuned [59]. These approaches could also yield new findings for machine learning in the field of data analytics in the energy domain.

9.3.4 Power Ramp Prediction

The prediction of power ramp events is critical for the integration of wind power into the grid. A ramp event is the sudden change of the produced power in a short time interval. It is defined by a threshold θ and a forecast horizon Δt, cf. Kramer *et al.* [66]. With a power measurements $p(t)$ for a point in time t, ramp events can be defined as:

$$Ramp(t) = \begin{cases} Up & \text{if } p(t + \Delta t) - p(t) > \theta \\ Down & \text{if } p(t + \Delta t) - p(t) < -\theta \\ No & \text{else} \end{cases}$$

The problem can be solved with classification algorithms. Kramer *et al.* [66] use SVM and different dimensionality reduction techniques and achieve a good detection rate for the occurring power ramps. Unfortunately, the rate of false positives is rather high and therefore, improvements of the classification are needed. The main reason is seen in the imbalanced dataset, i.e., rare occurrence of ramps in the training data set. Kramer *et al.* suggest that a classification approach based on regression techniques could be possible method for improvements. The ensemble regression techniques proposed in this work can help to yield a better ramp prediction with regression or be extended for classification. Further, ensembles can help to increase the accuracy for imbalanced datasets [119]. The method could also be extended with undersampling or oversampling techniques, e.g. [54], would integrate straightforwardly into the bagging approach and heterogeneous ensembles used in this work. Because of the relevance of the topic, further research based on this thesis dealing with the problem of wind power ramp prediction seems promising.

Part V

Appendices

A

Datasets

A.1 NREL

In our experiments, we use the *NREL Western Wind Resources Dataset* available at `http://wind.nrel.gov/`. It consists of simulated wind power output for $32,043$ wind turbines in the US, given in 10-minute time resolution for the years 2004 - 2006, which are also included in the WindML framework. For each turbine, there are $157,680$ wind speed and power output measurements available. Because the turbines in the datset are relatively dense, the neighbor turbines are used for implementing the spatio-temporal regression model because of high correlations. The IDs of the test turbines in the NREL dataset are:

Name	ID
Casper	23167
Cheyenne	17423
Hesperia	2028
Lancaster	2473
Las Vegas	6272

Name	ID
Palm Springs	1175
Reno	11637
Tehachapi	4155
Vantage	28981
Yucca Valley	1539

A.2 AEMO

Australia's largest gas and electricity markets and power systems are operated by the Australian Energy Market Operator (AEMO, `http://www.aemo.com.au/`). AEMO has provided a public dataset of 22 wind farms in south-east, offering 5-minute wind power data for the years 2011 and 2012, which are also included in the WindML framework. Because the turbines are relatively sparse, we only use the univariate time series for prediction. Some single values are missing and have been imputed using Last Observation Carried Forward (LOCF).

A.3 DWD

The German Weather Service (DWD) has made available several measurements for Germany. We use the dataset called "Historische stündliche Stationsmessungen der Windgeschwindigkeit und Windrichtung", available at `ftp://ftp-cdc.dwd.de/pub/CDC/observations_germany/climate/hourly/wind/historical/`. It provides hourly measurements of weather stations in Germany. Altough all experiments with the other datasets in this work predict the produced wind power, using this dataset we only look into the prediction of wind speed. Only the wind speed measurements of the target turbine itself are used.

B

Implementation Details

All implementations in this work are using

- PYTHON 2.7 (`http://python.org`)
- SCIKIT-LEARN 0.17 (`http://scikit-learn.org/`)
- NUMPY 1.10 (`http://numpy.org`)

Chapter 8 makes use of the EVOALGOS package providing the NSGA-II implementation (`https://ls11-www.cs.tu-dortmund.de/people/swessing/evoalgos/doc/`).
We want to thank all the contributors of these open source packages for their passionate work.

C

Publications

- **Justin Heinermann**, Jörg Lässig, and Oliver Kramer, "Evolutionary Multi-Objective Ensembles for Wind Power Prediction", in Proc. ECML Workshop DARE, 2016. In print.
- **Justin Heinermann**, Oliver Kramer, "Machine learning ensembles for wind power prediction", in *Renewable Energy*, Volume 89, April 2016, Elsevier
- **Justin Heinermann**, Oliver Kramer, "Short-Term Wind Power Prediction with Combination of Speed and Power Time Series", in Proc. KI, 2015
- Stefan Oehmcke, **Justin Heinermann**, and Oliver Kramer, "Analysis of Diversity Methods for Evolutionary Multi-Objective Ensemble Classifiers", in Proc. EvoStar, 2015.
- **Justin Heinermann** and Oiver Kramer, "On Heterogeneous Machine Learning Ensembles for Wind Power Prediction", in Proc. AAAI Workshops, 2015.
- Oliver Kramer, Fabian Gieseke, **Justin Heinermann**, Jendrik Poloczek, and Nilse André Treiber, "A Framework for Data Mining in Wind Power Time Series", in Proc. ECML Workshop DARE, 2014.
- **Justin Heinermann** and Oliver Kramer, "Precise Wind Power Prediction with SVM Ensemble Regression", In Proceedings of the 24th International Conference on Artificial Neural Networks (ICANN). Lecture Notes in Computer Science, Springer, 2014.

- **Justin Heinermann**, Oliver Kramer, Kai Lars Polsterer, and Fabian Gieseke, “On GPU-Based Nearest Neighbor Queries for Large-Scale Photometric Catalogs in Astronomy”, in Proc. Advances in Artificial Intelligence (KI). Lecture Notes in Artificial Intelligence, Springer, 2013.

D

Figures and Tables

List of Figures

List of Tables

Acronyms

ANN artificial neural network. 5, 6, 49, 127–129

ARIMA autoregressive integrated moving average. 36

ARMA autoregressive moving average. 36

bagging bootstrap aggregating. 48, 49, 52

CART classification and regression trees. 21, 23, 85

CFD computational fluid dynamic. 33

CPU central processing unit. 42, 83

CV cross-validation. 26, 55, 74, 106, 107, 121

DB database. 43

DBMS database management system. 43

DNN deep neural network. 129, 130

DT decision tree. 6, 8, 10, 18, 26, 45, 49, 81, 83, 86, 88, 90, 91, 95, 97, 99, 102, 109, 110, 113, 122, 145

EA evolutionary algorithm. 103, 105, 115

EMOA evolutionary multi-objective optimization algorithm. 7, 101, 102, 105, 111, 115, 123

EPEX European Power Exchange. 29

EU European Union. 4, 29

GPU graphics processing unit. 69

k-NN k-nearest neighbors. 5, 6, 8–10, 18–20, 26, 34, 36, 39, 40, 71–75, 77, 79, 81, 85, 86, 91–95, 98, 99, 103, 111, 121, 122

LOOCV Leave-one-out cross-validation. 25, 26

ML machine learning. 15, 43

MNIST Mixed National Institute of Standards and Technology database. 16

MOP multi-objective optimization problem. 101, 105

MSE mean squared error. 17, 23–25, 39, 40, 56, 61–63, 65, 67, 73, 76, 83, 87–95, 97, 99, 106, 120, 142

NSGA-II non dominated sorting genetic algorithm II. 115, 123

NWP numerical weather prediction. 4, 8, 43

QRF quantile regression forest. 127

RF random forest. 9, 21, 46, 49, 51, 52, 71–79, 106, 113, 121, 123, 130, 142

RSM random subspace method. 49, 52

SVM support vector machine. 18, 34, 56, 57

SVR support vector regression. 5, 6, 9, 10, 17, 18, 26, 34–36, 39, 40, 52, 55, 57–63, 66, 67, 69, 71–79, 81, 84–99, 102, 103, 106–111, 113, 119–123, 129, 142, 143, 145

References

1. S. Aggarwal and M. Gupta. Wind power forecasting: A review of statistical models. *International Journal of Energy Science*, 3, 2013.
2. E. Alpaydin. *Introduction to Machine Learning.* Adaptive computation and machine learning. MIT Press, 2010.
3. A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51:117–122, 2008.
4. G. E. Batista and M. C. Monard. A study of k-nearest neighbour as an imputation method. *HIS*, 87:251–260, 2002.
5. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36:105–139, 1999.
6. BDEW Bundesverband der Energie- und Wasserwirtschaft e.V. *Erneuerbare Energien und das EEG: Zahlen, Fakten, Grafiken (2016)* *https://www.bdew.de/internet.nsf/res/7BD63123F7C9A76BC1257F61005AA45F/£file/160218_Energie-Info_Erneuerbare%20Energien%20und%20das%20EEG_2016_final.pdf*. 2016.
7. Y. Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2:1–127, 2009.
8. J. L. Bentley. Multidimensional Binary Search Trees Used For Associative Searching. *Communications of the ACM*, 18:509–517, 1975.
9. J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13:281–305, 2012.
10. A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of the 23 International Conference on Machine Learning*, pages 97–104. ACM, 2006.
11. C. Bishop. *Pattern Recognition and Machine Learning.* Information Science and Statistics. Springer, 2006.
12. L. Breiman. Bagging predictors. *Machine learning*, 24:123–140, 1996.
13. L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

14. L. Breiman, J. Friedman, C. Stone, and R. Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.
15. G. Brown, J. Wyatt, R. Harris, and X. Yao. Diversity creation methods: a survey and categorisation. *Information Fusion*, 6:5–20, 2005.
16. B. Bustos, O. Deussen, S. Hiller, and D. Keim. A graphics hardware accelerated algorithm for nearest neighbor search. In *Proc. International Conference on Computational Science (ICCS'06) Part IV*, volume 3994 of *LNCS*, pages 196–199. Springer, 2006.
17. S. Cagnoni, R. Poli, G. D. Smith, D. Corne, M. Oates, E. Hart, P. L. Lanzi, E. J. Willem, Y. Li, B. Paechter, et al. *Real-World Applications of Evolutionary Computing: EvoWorkshops 2000: EvoIASP, EvoSCONDI, EvoTel, EvoSTIM, EvoRob, and EvoFlight, Edinburgh, Scotland, UK, April 17, 2000 Proceedings.* Springer Science & Business Media, 2000.
18. F. Castellani, M. Burlando, S. Taghizadeh, D. Astolfi, and E. Piccioni. Wind energy forecast in complex sites with a hybrid neural network and cfd based method. *Energy Procedia*, 45:188–197, 2014.
19. P. Chakraborty, M. Marwah, M. F. Arlitt, and N. Ramakrishnan. Fine-Grained Photovoltaic Output Prediction Using a Bayesian Ensemble. In *AAAI Conference on Artificial Intelligence*, 2012.
20. M. Chen, S. Mao, and Y. Liu. Big data: A survey. *Mobile Networks and Applications*, 19:171–209, 2014.
21. C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.
22. A. Costa, A. Crespo, J. Navarro, G. Lizcano, H. Madsen, and E. Feitosa. A review on the young history of the wind power short-term prediction. *Renewable and Sustainable Energy Reviews*, 12:1725–1744, 2008.
23. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6:182–197, 2002.
24. T. Dietterich. *Ensemble Methods in Machine Learning*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin Heidelberg, 2000.
25. H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. *Advances in neural information processing systems*, 9:155–161, 1997.
26. A. Eiben and J. E. Smith. *Introduction to Evolutionary Computing.* Springer, 2015.
27. EPEX Spot. *EPEX Spot and ECC successfully reduce lead time on all intraday markets.* *https://www.epexspot.com/en/press-media/press/details/press/EPEX_SPOT_and_ECC_successfully_reduce_lead_time_on_all_intraday_markets https://windeurope.org/wp-content/uploads/files/about-wind/statistics/EWEA-Annual-Statistics-2015.pdf.* 2015.
28. M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15:3133–3181, 2014.

29. A. M. Foley, P. G. Leahy, A. Marvuglia, and E. J. McKeogh. Current methods and advances in forecasting of wind power generation. *Renewable Energy*, 37:1–8, 2012.
30. Y. Freund, R. E. Schapire, et al. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, volume 96, pages 148–156, 1996.
31. F. Friedrichs and C. Igel. Evolutionary tuning of multiple svm parameters. *Neurocomputing*, 64:107–117, 2005.
32. L. Fugon, J. Juban, G. Kariniotakis, et al. Data mining for wind power forecasting. In *Proceedings European Wind Energy Conference & Exhibition EWEC 2008*, 2008.
33. F. Galton. Vox populi (the wisdom of crowds). *Nature*, 75:450–451, 1907.
34. V. Garcia, E. Debreuve, and M. Barlaud. Fast k nearest neighbor search using GPU. In *CVPR Workshop on Computer Vision on GPU*, Anchorage, Alaska, USA, June 2008.
35. F. Gieseke, J. Heinermann, C. Oancea, and C. Igel. Buffer kd trees: processing massive nearest neighbor queries on GPUs. In *Proceedings of The 31st International Conference on Machine Learning*, pages 172–180, 2014.
36. T. Gneiting, A. E. Raftery, A. H. Westveld, and T. Goldman. Calibrated Probabilistic Forecasting Using Ensemble Model Output Statistics and Minimum CRPS Estimation. *Monthly Weather Review*, 133:1098–1118, 2005.
37. H. P. Graf, E. Cosatto, L. Bottou, I. Durdanovic, and V. Vapnik. Parallel Support Vector Machines: The Cascade SVM. In *NIPS*, 2004.
38. S. Hassan, A. Khosravi, and J. Jaafar. Examining performance of aggregation algorithms for neural network-based electricity demand forecasting. *International Journal of Electrical Power & Energy Systems*, 64:1098 – 1105, 2015.
39. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition.* Springer Series in Statistics. Springer, 2009.
40. J. Heinermann and O. Kramer. Precise Wind Power Prediction with SVM Ensemble Regression. In *Artificial Neural Networks and Machine Learning–ICANN 2014*, pages 797–804. Springer, 2014.
41. J. Heinermann and O. Kramer. On heterogeneous machine learning ensembles for wind power prediction. In *AAAI Workshops*, 2015.
42. J. Heinermann and O. Kramer. Short-term wind power prediction with combination of speed and power time series. In *KI 2015: Advances in Artificial Intelligence*, pages 100–110. Springer, 2015.
43. J. Heinermann and O. Kramer. Machine learning ensembles for wind power prediction. *Renewable Energy*, 89:671–679, 2016.
44. J. Heinermann, O. Kramer, K. L. Polsterer, and F. Gieseke. On GPU-based nearest neighbor queries for large-scale photometric catalogs in astronomy. In *KI 2013: Advances in Artificial Intelligence*, pages 86–97. Springer, 2013.

45. T. Heskes. Practical confidence and prediction intervals. In *Proc. of the 1996 Conf. Advances in Neural Information Processing Systems*, volume 9, page 176, 1997.
46. T. K. Ho. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20:832–844, 1998.
47. S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, 1998.
48. Q.-H. Hu, D.-R. Yu, and M.-Y. Wang. Constructing rough decision forests. In *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 147–156. Springer, 2005.
49. Q. Huang, J. Z. Kang, X. Wang, P. Guo, and X. Huang. A review of wind power forecasting models. *Energy Procedia*, 12:770 – 778, 2011.
50. G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.
51. J. Juban, L. Fugon, and G. Kariniotakis. Probabilistic short-term wind power forecasting based on kernel density estimators. In *European Wind Energy Conference and exhibition, EWEC 2007*, pages http–ewec2007proceedings, 2007.
52. C. Junk, L. Delle Monache, S. Alessandrini, G. Cervone, and L. von Bremen. Predictor-weighting strategies for probabilistic wind power forecasting with an analog ensemble. *Meteorologische Zeitschrift*, 24:361–79, 2015.
53. S. Kalogirou. Applications of artificial neural networks in energy systems. *Energy Conversion and Management*, 40:1073–1087, 1999.
54. P. Kang and S. Cho. EUS SVMs: Ensemble of under-sampled svms for data imbalance problems. In *International Conference on Neural Information Processing*, pages 837–846. Springer, 2006.
55. R. G. Kavasseri and K. Seetharaman. Day-ahead wind speed forecasting using f-arima models. *Renewable Energy*, 34:1388–1393, 2009.
56. H.-C. Kim, S. Pang, H.-M. Je, D. Kim, and S. Y. Bang. Constructing support vector machine ensemble. *Pattern Recognition*, 36:2757–2767, 2003.
57. R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.
58. R. Kohavi, D. H. Wolpert, et al. Bias plus variance decomposition for zero-one loss functions. In *ICML*, volume 96, pages 275–83, 1996.
59. P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1467–1475, 2015.
60. O. Kramer. *Self-adaptive heuristics for evolutionary computation*, volume 147. Springer, 2008.
61. O. Kramer. Cascade support vector machines with dimensionality reduction. *Hindawi Applied Computational Intelligence and Soft Computing*, 2015:216132:1–216132:8, 2015.

62. O. Kramer and F. Gieseke. Short-Term Wind Energy Forecasting Using Support Vector Regression. In *SOCO*, pages 271–280, 2011.
63. O. Kramer, F. Gieseke, J. Heinermann, J. Poloczek, and N. A. Treiber. A framework for data mining in wind power time series. In *Data Analytics for Renewable Energy Integration - Second ECML PKDD Workshop, DARE 2014, Nancy, France, September 19, 2014, Revised Selected Papers*, pages 97–107, 2014.
64. O. Kramer, F. Gieseke, and B. Satzger. Wind energy prediction and monitoring with neural computation. *Neurocomputing*, 109:84–93, 2013.
65. O. Kramer, N. A. Treiber, and F. Gieseke. Machine Learning in Wind Energy Information Systems. In *EnviroInfo*, pages 16–24, 2013.
66. O. Kramer, N. A. Treiber, and M. Sonnenschein. Wind power ramp event prediction with support vector machines. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 37–48. Springer, 2014.
67. T. Krishnamurti. Numerical weather prediction. *Annual review of fluid mechanics*, 27:195–225, 1995.
68. A. Kusiak, H. Zheng, and Z. Song. Short-term prediction of wind farm power: a data mining approach. *Energy Conversion, IEEE Transactions on*, 24:125–136, 2009.
69. M. Lange and U. Focken. New developments in wind energy forecasting. In *Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, pages 1–8. IEEE, 2008.
70. H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin. Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, 10:1–40, 2009.
71. Y. LeCun and C. Cortes. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2010.
72. T.-H. Lee and Y. Yang. Bagging binary and quantile predictors for time series. *Journal of econometrics*, 135:465–497, 2006.
73. M. Lei, L. Shiyan, J. Chuanwen, L. Hongling, and Z. Yan. A review on the forecasting of wind speed and generated power. *Renewable and Sustainable Energy Reviews*, 13:915–920, 2009.
74. R. Little and D. Rubin. *Statistical Analysis With Missing Data.* Wiley Series in Probability and Statistics - Applied Probability and Statistics Section Series. Wiley, 1987.
75. H. Liu, H.-q. Tian, and Y.-f. Li. Comparison of two new arima-ann and arima-kalman hybrid methods for wind speed prediction. *Applied Energy*, 98:415–424, 2012.
76. G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts. Understanding variable importances in forests of randomized trees. In *Advances in Neural Information Processing Systems*, pages 431–439, 2013.
77. P. C. Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55, 1936.

78. W. P. Mahoney, K. Parks, G. Wiener, Y. Liu, W. L. Myers, J. Sun, L. Delle Monache, T. Hopson, D. Johnson, and S. E. Haupt. A wind power forecasting system to optimize grid integration. *IEEE Transactions on Sustainable Energy*, 3:670–682, 2012.
79. S. Marsland. *Machine learning: an algorithmic perspective.* Chapman & Hall/CRC machine learning & pattern recognition series. CRC Press/Taylor Francis, Boca Raton, Mass. [u.a.], 2009. XVI, 390 S. ; 24 cm : Ill., graph. Darst.
80. I. Martí, M. San Isidro, D. Cabezón, Y. Loureiro, J. Villanueva, E. Cantero, and I. Pérez. Wind power prediction in complex terrain: from the synoptic scale to the local scale. In *CD-Rom Proceedings of the Conference: The Science of making Torque from Wind, Delft, The Netherlands*, 2004.
81. N. Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999, 2006.
82. I. Mierswa. Controlling overfitting with multi-objective support vector machines. In *Genetic and Evolutionary Computation Conference (GECCO)*, pages 1830–1837, 2007.
83. M. Mohandes, T. Halawani, S. Rehman, and A. A. Hussain. Support vector machines for wind speed prediction. *Renewable Energy*, 29:939–947, 2004.
84. F. Molteni, R. Buizza, T. N. Palmer, and T. Petroliagis. The ecmwf ensemble prediction system: Methodology and validation. *Quarterly journal of the royal meteorological society*, 122:73–119, 1996.
85. S. Oehmcke, J. Heinermann, and O. Kramer. Analysis of diversity methods for evolutionary multi-objective ensemble classifiers. In *Applications of Evolutionary Computation - 18th European Conference, EvoApplications 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings*, pages 567–578, 2015.
86. N. C. Oza and K. Tumer. Classifier ensembles: Select real-world applications. *Information Fusion*, 9:4–20, 2008.
87. J. Palomares-Salas, J. De la Rosa, J. Ramiro, J. Melgar, A. Aguera, and A. Moreno. Arima vs. neural networks for wind speed forecasting. In *2009 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, pages 129–133. IEEE, 2009.
88. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
89. J. Poloczek, N. A. Treiber, and O. Kramer. KNN regression as geo-imputation method for spatio-temporal wind data. In *International Joint Conference SOCO'14-CISIS'14-ICEUTE'14 - Bilbao, Spain, June 25th-27th, 2014, Proceedings*, pages 185–193, 2014.
90. J. R. Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.
91. J. R. Quinlan. *C4. 5: programs for machine learning.* Morgan Kaufmann, 1993.

92. P. Ramasamy, S. Chandel, and A. K. Yadav. Wind speed prediction in the mountainous region of india using an artificial neural network model. *Renewable Energy*, 80:338 – 347, 2015.
93. I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologishen Evolution.* 1973.
94. L. Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33:1–39, 2010.
95. M. Saar-Tsechansky and F. J. Provost. Handling missing values when applying classification models. *Journal of Machine Learning Research*, pages 1623–1657, 2007.
96. S. Salcedo-Sanz, J. Rojo-Álvarez, M. Martínez-Ramón, and G. Camps-Valls. Support vector machines in engineering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4:234–267, 2014.
97. A. Scherbart and T. W. Nattkemper. The diversity of regression ensembles combining bagging and random subspace method. In *International Conference on Neural Information Processing*, pages 911–918. Springer, 2008.
98. G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing).* MIT Press, 2006.
99. J. Shi, X. Qu, and S. Zeng. Short-term wind power generation forecasting: direct versus indirect arima-based approaches. *International Journal of Green Energy*, 8:100–112, 2011.
100. A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.
101. J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.
102. S. S. Soman, H. Zareipour, O. Malik, and P. Mandal. A review of wind power and wind speed forecasting methods with different time horizons. In *North American Power Symposium (NAPS), 2010*, pages 1–8. IEEE, 2010.
103. M. Stone. Cross-validation and multinomial prediction. *Biometrika*, 61:509–515, 1974.
104. J. Stubbemann, N. A. Treiber, and O. Kramer. Resilient propagation for multivariate wind power prediction. In *ICPRAM 2015 - Proceedings of the International Conference on Pattern Recognition Applications and Methods, Volume 2, Lisbon, Portugal, 10-12 January, 2015.*, pages 333–337, 2015.
105. Y. Tang. Deep learning using linear support vector machines. In *In ICML*. Citeseer, 2013.
106. The European Wind Energy Association. *Wind in power – 2015 European statistics.* *https://windeurope.org/wp-content/uploads/files/about-wind/statistics/EWEA-Annual-Statistics-2015.pdf.* 2016.
107. T. L. Thorarinsdottir and T. Gneiting. Probabilistic forecasts of wind speed: ensemble model output statistics by using heteroscedastic censored regression. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 173:371–388, 2010.

108. N. A. Treiber, J. Heinermann, and O. Kramer. Aggregation of features for wind energy prediction with support vector regression and nearest neighbors. In *European Conference on Machine Learning, Workshop Data Analytics for Renewable Energy Integration*, 2013.
109. N. A. Treiber and O. Kramer. Evolutionary feature weighting for wind power prediction with nearest neighbor regression. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 332–337. IEEE, 2015.
110. N. A. Treiber, S. Späth, J. Heinermann, L. von Bremen, and O. Kramer. Comparison of numerical models and statistical learning for wind speed prediction. In *ESANN*, 2015.
111. I. W. Tsang, A. Kocsor, and J. T. Kwok. Diversified SVM Ensembles for Large Data Sets. In *European Conference on Machine Learning*, pages 792–800, 2006.
112. K. Upadhyay, A. Choudhary, and M. Tripathi. Short-term wind speed forecasting using feed-forward back-propagation neural network. *International Journal of Engineering, Science and Technology*, 3:107–112, 2011.
113. P. A. Valdes-Sosa. Spatio-temporal autoregressive models defined over brain manifolds. *Neuroinformatics*, 2:239–250, 2004.
114. S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13:22–30, 2011.
115. P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.
116. W. Waegeman and L. Boullart. An ensemble of Weighted Support Vector Machines for Ordinal Regression. *Transactions on Engineering, Computing and Technology*, 12:71–75, 2006.
117. S. Wager, T. Hastie, and B. Efron. Confidence intervals for random forests: the jackknife and the infinitesimal jackknife. *Journal of Machine Learning Research*, 15:1625–1651, 2014.
118. J. Wang, J. Hu, K. Ma, and Y. Zhang. A self-adaptive hybrid approach for wind speed forecasting. *Renewable Energy*, 78:374–385, 2015.
119. S. Wang and X. Yao. Diversity analysis on imbalanced data sets by using ensemble models. In *Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on*, pages 324–331. IEEE, 2009.
120. K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2005.
121. I. Witten, E. Frank, and M. Hall. *Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques.* The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2011.
122. D. H. Wolpert. Stacked generalization. *Neural networks*, 5:241–259, 1992.

123. X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14:1–37, 2008.
124. L. Xiao, J. Wang, Y. Dong, and J. Wu. Combined forecasting models for wind energy forecasting: A case study in china. *Renewable and Sustainable Energy Reviews*, 44:271–288, 2015.
125. A. Zameer, A. Khan, S. G. Javed, et al. Machine learning based short term wind power prediction using a hybrid learning model. *Computers & Electrical Engineering*, 45:122–133, 2015.
126. H. Zhang, L. Chen, Y. Qu, G. Zhao, and Z. Guo. Support vector regression based on grid-search method for short-term wind power forecasting. *Journal of Applied Mathematics*, 2014, 2014.
127. P. Zikopoulos and C. Eaton. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Osborne Media, 1st edition, 2011.

Acknowledgements

I would like to thank Jun.-Prof. Dr. habil. Oliver Kramer for the supervision of my work and the invaluable support. Further, I want to thank the ministry of science and culture of Lower Saxony for supporting me with a scholarship in the PhD program *System Integration of Renewable Energies*.