Machine Learning Concepts in Predictive Analytics

A Case Study on Wind Turbine Data

Chair of Information Technology
Montanuniversität Leoben

Submitted by:
Elmar Steiner
0935078

Advisors:
Prof. Peter Auer
Niklas Goby

A thesis submitted for the degree of
Master of Science
Leoben 2017

Affidavit

I declare in lieu of oath, that I wrote this thesis and performed the associated research myself, using only literature cited in this volume.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich diese Arbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe.

Leoben, _____ _____

Date Signature

Confidentiality clause

The presented thesis with title

Machine Learning Concepts in Predictive Analytics

contains confidential data of *ZF Friedrichshafen AG*. For this reason and personal interests the work may only be made available to the first and second reviewers and authorized members of the board of examiners of the Montanuniversität Leoben. Any publication, duplication or transmission to elecronic data processing systems - even in part - is prohibited. An inspection of this work by third parties is explicitly forbidden and requires permission of both author and the company *ZF Friedrichshafen AG*.

Elmar Steiner

Abstract

Due to current structural change in energy systems (*energy transition*), the construction of wind turbines in order to provide a sustainable production of energy rose to prominence in the last couple of years. While investment in renewable energy has been supported by governments in one way or the other in the past, ceased subsidy may strongly influence profitability in the future. The costs of construction as well as for operations and maintenance (O&M) can be significant. Unscheduled maintenance, often caused by breakdown of the turbine, has been identified as a major part of overall (O&M) costs, owing to increased logistical expenditure and lost revenue. As a consequence *condition monitoring systems* (CMS) more and more have manifested themselves to measure behavior of the turbine or individual components in order to provide diagnostic information. This in turn facilitates the scheduling of repair work as well as allows the prediction and consecutive avoidance of component failures that may lead to a breakdown. CMS can have access to various data sources in form of time series, that include vibration, voltage levels or performance related information. Common CMS approaches and related research likewise are often focused on *anomaly detection* in time series by applying physical models and various transformations on the data (e.g. Wavelet transformation). The usage of *machine learning* algorithms for prediction lacks a more thorough investigation.
The subject of this thesis is to contribute to the closure of this gap by applying two different machine learning algorithms to performance related data of a wind turbine in order to detect anomalies and predict events in defined future time intervals. Despite the technical nature of this specific case, we aim to present the applied approach as valuable procedure for any time series prediction as often occurring in the domain of *predictive analytics*. Besides physical time series data also sequential event data were provided. Because of it's proclaimed applicability for modeling time series, a *restricted Boltzmann machine* (RBM) with Gaussian visible units has been used. As it is a generative model the trained RBM can estimate the probability of a given set of input variables, that are the sensor values at a specific point in time. In case the probability falls below a threshold the occurrence of an event is predicted. The second applied algorithm was a *support vector machine* (SVM), a binary linear classifier which is trained through supervised learning. To examine the performances of the methods for the respective time interval a *receiver operating characteristic* (ROC) was used. Thereby not only false event predictions but also the number of missed events could be investigated. The results substantiate the approach to be valuable for data analysis, although further improvements are possible. Furthermore, the derived predictions can be enhanced by decision prescriptions in the sense of *prescriptive analytics*. A link to entrepreneurial circumstances could enlarge practical benefits.

Kurzfassung

Im Zuge der strukturellen Umstellung zu einer nachhaltigen Energieversorgung (*Energiewende*) hat der Einsatz von Windkraftanlagen in den vergangenen Jahren an Bedeutung gewonnen. Die Kosten sowohl für Errichtung als auch für Wartung und Betrieb können erheblich sein. Es konnte festgestellt werden, dass außerplanmäßige Wartung einen wesentlichen Anteil der Kosten ausmacht, welche erhöhten logistischen Aufwänden und Umsatzverlusten geschuldet sind. Infolgedessen haben sich sogenannte *Condition Monitoring Systems* (CMS) etabliert, welche das Verhalten der Anlage oder einzelnen Komponenten messen um diagnostische Informationen über den Zustand liefern zu können. Dadurch kann die Planung der Wartungsarbeiten verbessert, sowie eine Vorhersage und folgerichtige Vermeidung von Schäden ermöglicht werden. CMS haben Zugriff auf unterschiedliche Zeitreihendaten, die z.B. Vibration oder Stromgrößen. CMS und damit verbundene Forschungsarbeiten fokussieren oft die *Anomalie-Erkennung* in Zeitreihen mithilfe physikalischer Modelle und verschiedener Transformationen. Der hingegen geringere Einsatz von Algorithmen des *maschinellen Lernens* bedarf einer intensiveren Untersuchung. Der Inhalt vorliegender Arbeit soll dazu beitragen diese Lücke zu schließen, indem zwei verschiedene Algorithmen des maschinellen Lernens auf leistungsbezogene Zeitreihendaten angewandt werden, um Ereignisse in zukünftigen Zeitintervallen vorherzusagen. Ungeachtet des technischen Charakters des Anwendungsfalls, ist die Vorgehensweise auf jedwede Vorhersage von Zeitreihen - wie häufig im Bereich von *Predictive Analytics* erforderlich - anwendbar. Aufgrund von Empfehlungen gängiger Fachliteratur wurde eine *Restricted Boltzmann Maschine* (RBM) mit Gauß'schen sichtbaren Knoten implementiert. Da sie ein generatives Modell ist, kann eine trainierte RBM die Auftrittswahrscheinlichkeit von Eingangsvariablen, d.h. den Sensorwerten zu einem bestimmen Zeitpunkt, abschätzen. Fällt die Wahrscheinlichkeit unter einen gewissen Schwellwert, wird eine Anomalie vorhergesagt. Der zweite verwendete Algorithmus ist eine *Support Vector Maschine* (SVM), ein linearer Klassifikator, welcher durch *supervised learning* trainiert wird. Um die Leistungsfähigkeit der Methoden zu prüfen, wurde für verschiedene Zeitintervalle eine *Receiver-Operating-Characteristic* (ROC) angewandt. Die Ergebnisse bestätigen, dass die Vorgehensweise für Datenanalyse nützlich ist, obwohl Verbesserungen möglich sind. Überdies kann die gesamte Methodik im Sinne von *Prescriptive Analytics* erweitert werden, indem die Prädiktion mit darauf aufbauenden Handlungsvorschlägen ergänzt wird. Verknüpfungen zu Rahmenbedingungen könnten so größeren praktischen Nutzen ermöglichen.

Contents

List of Figures

List of Tables

1 Introduction

The introduction of this thesis shall give an appropriate overview of the context the work is related to, as well as define and explain the terms in use. An extensive treatise of the topics would by far exceed the scope of the thesis, so that for any additional information it shall be referred to the referenced literature.

1.1 Predictive analytics

Predictive analytics is regarded as the second phase of *business analytics*[34], which in turn refers to an extensive conglomeration of data-based practices to explore and investigate business performance and to improve business planning.[4]

As can be seen in figure 1.1 predictive analytics shall be understood as to build on the preceding phase, thus inheriting techniques as well as methodology and leads to the final phase, *prescriptive analytics*. However, the term of prescriptive analytics is not well defined so far and thus the distinction to predictive analytics remains ambiguous and scrutinized.[56]

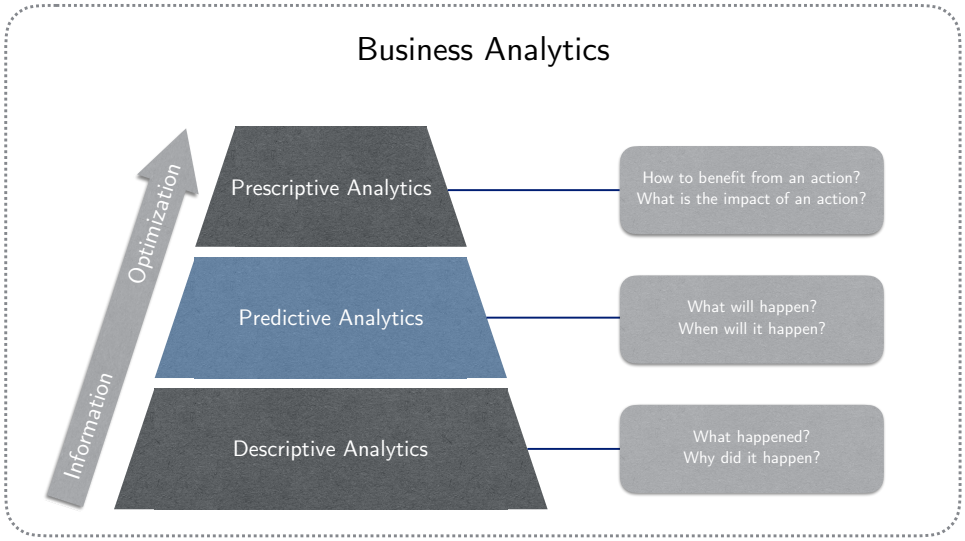


Figure 1.1: Phases of Business Analytics.

While *descriptive analytics* strives to explain past behavior by quantifying relationships in data, predictive analytics seeks to predict trends and behavior patterns using *data mining* and *machine learning* techniques. If not deployed commercially the latter usually is accompanied - if not even used synonymously - by the term *predictive modeling* for the purpose of emphasizing the modeling of relationships prior to prediction.[40]

Consequently prescriptive analytics is now considered to enhance the derived predictions with meaningful action/decision prescription in order to optimize one or more predefined metrics by linking them to given entrepreneurial circumstances (strategic directions, business processes, side conditions etc.). Figure 1.1 further depicts the idea that evolving from descriptive to prescriptive analytics is paralleled by changing the procedure's objective from the gaining of information to optimization.
As stated this extension does not quite serve for proper differentiation due to vagueness, so that an additional perspective seems reasonable.

Given the characteristic of data-based decision support one may also integrate prescriptive analytics into the realm of *data science* in general and *data-driven decision making* (DDD) in particular as both have well established descriptions.

Figure 1.2 illustrates the connection between these two and explicitly separates them from disciplines, which may be mistakenly confused to be data science, but belong to different activities - here jointly referred to as *data engineering and processing*.
The overall objective of data science is to facilitate and improve decision making by involving certain principles, processes and techniques. DDD addresses the modus operandi of basing decisions on data analysis rather than on pure intuition.[1]

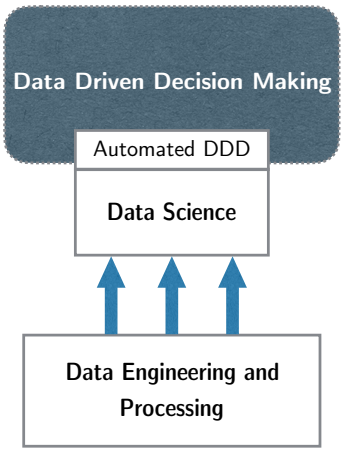


Figure 1.2: Data science in the context of data-related processes.

Finally an overlap between DDD and data science exists, that represents business decisions which are not only data driven but also partially or fully automated.[44]

With regard to our initial attempt of categorization it becomes evident that the discipline of prescriptive analytics can belong to this overlap, whereas predictive analytics cannot. While the former automatically extends the range of application for information based on prediction, the latter strictly pertains to data science, because the final use of the prediction's results maintains

[1] It has been shown extensively that enterprises following this practice turn out to be more productive. Actually there exists a positive correlation with return on assets, return on equity, asset utilization and market value.[10]

to be user-driven.However, prescriptive analytics does not explicitly match the definition of automated DDD as analytics, in general, aims at facilitating decisions in business planning rather than fully automating processes.[2]

One may conclude that a differentiation is legitimate due to a reduced user-driven decision making by optimization with respect to user-defined metrics, while a full automation is not the general purpose of analytics.

Nevertheless, predictive analytics as well as prescriptive approaches find a use in a myriad of different application fields including marketing, financial services, actuarial science or pharmaceuticals.

1.1.1 Related Disciplines

To conduct predictive analytics one makes use of various analytical techniques or process models (see section 1.1.2) of related disciplines. However, it has to be stated, that the scientific community provides different perspectives in regard to the relationship of the individual disciplines to each other. While this thesis considers[3] predictive analytics as part of business analytics irrespective to data mining, which is in turn a sub-field of computer science[16], Finlay *et.al.*(2014) proposes, that the former is actually a sub-area of the latter.[19]

Data mining

'Data mining is the process of automatically discovering useful information in large data respositories'[53]. It seeks to discover useful and novel patterns (*features*) using methods of *machine learning* and statistics. Furthermore it comprises the capability of prediction. Though the term is used as a buzzword for any information discovery involving large-scale databases, and even for preparatory steps, the distinction to *information retrieval* is clear and well-defined. In fact information retrieval is relying on traditional computer science techniques and focuses on more trivial features and the reorganization of data. Data mining uses methods, whose origin is statistics and machine learning, such as estimation, hypothesis testing, search algorithms or modeling. The tasks are generally separated into two major classes.[53]

- **Predictive tasks** aim to predict values (*targets*) of particular attributes based on others (*explanatory variables*).
- **Descriptive tasks** on the other hand seek to discover patterns such as *correlations*, *clusters* or *anomalies* in data.

Within these classes one can in turn identify four core tasks, which are *cluster analysis*, *association analysis*[4], *predictive modeling* and *anomaly detection*, whereas the latter two will take a central role in this thesis.

[2] Although this is in dispute as some professionals consider them to be synonyms (e.g. [59]).
[3] The considerations were motivated by ideas from Provost (2013)[44].
[4] *Association rule learning* is a very prominent method of data mining to discover interesting relationships between variables, which is known for its application in retail business (*market basket analysis*)[1].

Machine Learning

An often quoted definition of machine learning has been proposed by Mitchell (1997)[37]:

> *A computer program is said to learn from experience $\mathcal{E}$ with respect to some class of tasks $\mathcal{T}$ and performance measure $\mathcal{P}$, if its performance at tasks in $\mathcal{T}$, as measured by $\mathcal{P}$, improves with experience $\mathcal{E}$.*

Historically machine learning can be seen as a branch of *artificial intelligence*, although the strong statistical line of research of the former caused a rift between the two. In machine learning systems are designed and trained to learn from data, improve with experience and can be used to predict outcomes based on previous learning. The manifold algorithms in machine learning can be generally categorized based (1) on their characteristic learning type or (2) their application, that is the desired outcome.[5][46]
The learning type is depending on the form of the learning signal the system has access to and falls into the following classes:[46]

- **Supervised learning** refers to training with labeled data. That is, for every training example exists an explanatory variable (input) and the desired output value.[5]
- **Unsupervised learning** algorithms, on the contrary, cope without labels and seek to find hidden patterns or structure in the input data.
- **Reinforcement learning** comprises a different setting. In fact it studies the behavior of *agents* taking *actions* in a dynamic environment to maximize some *reward*.

As mentioned machine learning algorithms can also be classified according to the desired output which includes the following.[46]

- **Classification**
  Input data belong to different classes and the algorithm seeks to assign unseen inputs to one or more of them.
- **Clustering**
  Clustering aims to divide input data into groups, but on the contrary to classification, the groups are not known prior to training.
- **Regression**
  Hereby the output values are not discrete but continuous.

Examples for algorithms are *neural network*, *restricted Boltzmann machine Bayesian network*, *support vector machine*, *random forest*, *self-organizing map* or *Q-learning*.

[5] One furthermore distinguishes between the principle of reasoning from observed training cases to general rules (*transduction*) or reasoning from observed training cases to specific test cases (*Induction*).

1.1.2 Process model

Due to the magnitude of different problem cases, incorporated disciplines and potential solutions, a process model for standardization of the overall procedure is necessary to allow consistency, repeatability and objectiveness.
As no specific structure for predictive analytics applications exists, one may incorporate well-established models from related disciplines.

Cross Industry Standard Process for Data Mining

The *Cross Industry Standard Process for Data Mining* (CRISP-DM) is a non-proprietary, documented, freely-available data mining model established in 1996. As illustrated in figure 1.3 the model comprises six process phases with clearly defined tasks consolidated in an iterative design. The illustration as well as the following description is taken from Shearer (2000).[49]

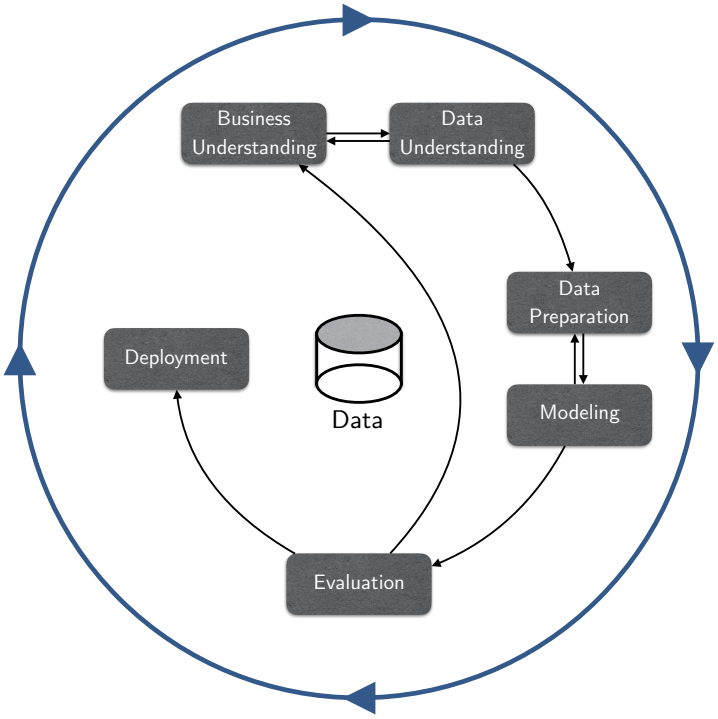


Figure 1.3: The CRISP-DM.

- **Business understanding**
  Considered as outstandingly important the phase's focus is firstly to determine the objectives and success criteria from a business perspective. Secondly the situation has to be properly assessed in regard to requirements, available resources, constraints, risks as well as costs and benefits. Finally one converts the knowledge into a data mining problem definition which in turn allows the draft of a project plan and an initial assessment of techniques.

- **Data understanding**
  This phase involves initial collection and description of data. Furthermore the familiarity

with the data is increased by exploration, gathering a-priori information and establishing hypotheses about hidden information. The verification of data quality (i.e. completeness, erroneousness) is also particularly significant.

- **Data preparation**
  Based on the predefined goals as well as technical and quality constraints (e.g. limits of data volume or type) tables, records or attributes of data are selected for the further analysis. This reduction is often coercive facing 'big data' volumes nowadays. However, many data mining techniques are also quite delicate to unnecessary but impairing variables.
  Further steps are cleaning of data, integration of data from multiple sources and transformation into an applicable format as required by the respective algorithm.

- **Modeling**
  The primary objective during this phase is to establish some sort of model or pattern capture of the data. Thus, one has to select the appropriate technique, test the validity and quality by running empirical tests,[6] then run it on the prepared data and eventually interpret the models according to the domain knowledge or the success criteria. This process is paralleled with the setting and adjustment of the model's parameters.

- **Evaluation**
  Before the final deployment can take place, a thorough evaluation of the model is necessary. While it's general accuracy and capability has been addressed in the previous phase this phase assesses to which extent the model achieves the business objectives and explicitly which objectives could not have been met.
  At this point also the whole process and its phases undergo an exhaustive review in order to detect deficiencies in the approach and as depicted in 1.3 one must now decide whether going to deployment or to initiate another process iteration.

- **Deployment**
  The very last phase resembles well-known procedures in project management and covers the definition of the deployment strategy, the planning of monitoring and maintenance (especially in case of a integration into daily business), production of a final report and finally the review of the whole project.

Figure 1.3 makes it explicit that iteration is a crucial component of this process model. It emphasizes that a complete and entire problem understanding is not premised but evolving during the process. So does the understanding of the given data or the appropriateness of the applied model. Actually even the evaluation may reset the business objectives and thus recommence the cycle.

[6] It shall be noted that determining the strength of a model is rarely trivial. In supervised data mining tasks this is usually done by somehow quantifying the error of the prediction.

The KDD Process

The process of *knowledge discovery in databases* (KDD) is based on a different, more holistic perspective. In contrary to CRISP-DM the KDD process understands data mining as a single step in the overall process of knowledge discovery (see figure 1.4), which is in turn defined as

> *The nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.*[18]

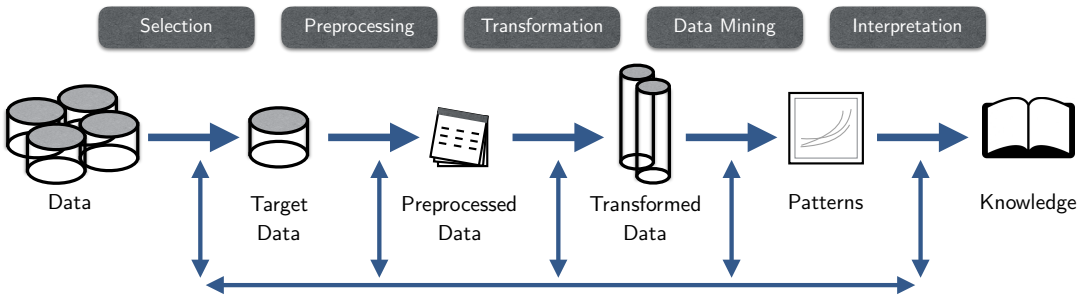


Figure 1.4: The steps of the KDD process.

The process is designed to be iterative and consists of the steps below[18], nevertheless the form of iterative transitions between steps are not as clearly defined as in CRISP.

- **Selection**
  This phase includes the acquisition of prior knowledge of the application domain and the establishment of business targets. Based on the captured information one selects relevant parts of the dataset or focuses on a reduced number of variables.

- **Pre-Processing**
  Processing of data comprises cleaning (e.g. filling missing data fields), noise reduction or the elimination of outliers if expedient.

- **Transformation**
  The characteristics of the data may require a dimensionality reduction or a transformation method to a different domain (e.g. Fourier tranformation).

- **Data mining**
  This essential part of the process comprises not only the actual search for patterns, but also the prior determination of function (e.g. classification, regression, clustering) and the appropriate algorithmic methods or models. The latter further includes parameter setting and optimization.

- **Interpretation and usage**
  Eventually the discovered patterns are interpreted and evaluated, which includes visualization and translation of the discovered results into a form understandable by the user. At this point - with the awareness of the actual outcome - one may also return to a previous step in order to improve the overall result.

The authors of the KDD process try to emphasize that preliminary or planning phases deserve equal attention and valuation as the actual data mining and are often overlooked in other process schemes.[18] However, it has to be stated that the contents, as described above, can be found in CRISP-DM as well, although the arrangement and the denotation of steps differ.[18] A comparison is illustrated in table 1.1[3].

KDD	CRISP-DM
Selection	Data preparation / Business understanding
Pre-processing / Transformation	Data understanding / Data preparation
Data mining	Modeling
Interpretation	Evaluation / Deployment

Table 1.1: Comparison of process steps.

1.2 Time series analysis

Time series are sequences of observations $X_t \in \{X_t\}$, that are listed in time order. Usually these observations are captured at successive equally spaced points in time. Domains of occurrence are manifold and include meteorology, economy and finance, marketing, industry or biology. The analysis of data in form of time series has multiple reasons also depending on the domain and can be generally categorized as follows.

- **Prediction** of future data points based on past observations
- **Understanding** of the underlying mechanism, which defines the time series
- **Control** of the process producing the series
- **Description** of non-trivial salient features of the data

It has to be stated that although time series often represent processes that change in a continuous way, in practice digital recording is done discretely in time. The measured values are considered to be a combination of a *systematic* part, which is a deterministic function of time, and a *stochastic* sequence (i.e. a *residual* term, also called noise). Furthermore one distinguishes univariate and multivariate time series, depending on the number of variables that are measured at each point in time.

In time series analysis the pivotal aspect is to determine how observations are related to each other in time, which is called *autocovariance*.[39] This is measuring the degree of second order variation, that is the covariance σ^2, between two elements at two different times. Formally the autocovariance between X_t and X_s of a process $\{X_t\}$ is defined as

$$\operatorname{cov}(X_t, X_s) = \mathbb{E}[\{X_t - \mathbb{E}(X_t)\}\{X_s - \mathbb{E}(X_s)\}]. \tag{1.1}$$

1.2.1 Stationarity

Simply put, the statistical properties of a *stationary* process do not change over time. Formally, a stochastic process is called *strictly stationary* when the joint statistical distribution of $X_{t_1}, ..., X_{t_l}$ is the same as the joint statistical distribution of $X_{t_1+\tau}, ..., X_{t_l+\tau}$ for all l and τ. This signifies, that all moments of all degrees are identical throughout the process. Consequently the joint distribution of (X_t, X_s) is the same as (X_{t+r}, X_{s+r}), showing clearly that it cannot depend on s or t but only on the distance between them.[39] A *weak stationary* process allows mean and variance to be independent of t and thus the autocovariance between X_t and $X_{t+\tau}$, with $\tau \in \mathbb{N}^+$, can only depend on the so called *lag* τ and hence

$$\gamma(\tau) = \text{cov}(X_t, X_{t+\tau}). \tag{1.2}$$

Another widely used measure is the *autocorrelation* of a process, which is a normalized version of autocovariance to values between -1 and 1:[39]

$$\rho(\tau) = \gamma(\tau)/\gamma(0), \tag{1.3}$$

with $\tau \in \mathbb{N}^+$ and $\gamma(0) = \text{cov}(X_t, X_t) = \text{var}(X_t)$.
However, one should consider stationary processes as models, which may not fit the needs of real-world problems.Time series can be non-stationary due to multiple reasons. Very common ones, however, are trends in mean, caused by the presence of a *unit root*[7] or a *deterministic trend*. While in case of a deterministic trend, referred to as *trend stationary process*, effects given a stochastic shock are transitory, in the case of a unit root they are permanent.
A trend stationary process can be transformed to a strictly stationary process by eliminating the underlying trend. Applying *differencing* can make processes with unit roots stationary.

Stationarity testing

There are multiple ways to determine whether a time series is stationary, whereas the respective effectiveness strongly coheres with the complexity of the stationarity violations. As mentioned quite common but already challenging effects are mean change, variance change or change in the autoregressive coefficients. Time series in engineering domains may constitute even more complex issues as *long range dependence*, *fractional integration* or *pink noise*. Hence, analyzing data in frequency instead of time domain may facilitate the determination. This so-called *spectral* approach is often implemented by applying *Wavelet* or *Fourier Transforms* to the data.

The basic idea of testing is to consider alternative hypotheses H_i for any assumption of stationarity (*null hypothesis* H_0). It has to be stated that albeit tests for complex effects have been developed (e.g. *Dickey-Fuller test*, *Priestley-Subba Rao test* (PSR)), formal hypothesis tests tend to concentrate on a specific type of alternative being insensitive to others at the same time.[39] As a logical consequence various tests are applied jointly on the data, which is called a *multiple hypothesis test* and in turn constitutes a severe drawback, referred to as *multiple comparison problem*.

[7] A stochastic process has a unit root if 1 is a root of the characteristic equation.

This problem arises when a set of statistical inferences are conducted simultaneously, since errors in inference - false positive hypothesis tests in this case[8] - are more likely to occur by multiple testing on identical samples. A common measure to counteract this accumulation of type I errors is the *Bonferroni correction*. It follows *Boole's inequality* and is based on the idea that the *familywise error rate* (FWER) is derived by testing each of the m hypotheses at a significance level of $\alpha * \frac{1}{m}$, where α is the desired overall significance level.

1.2.2 ARMA models

ARMA models are capable of describing (weakly) stationary processes and are a common approach for modeling univariate time series. Actually they are a concatenation of a *autoregressive* model (AR) and a *moving average* model. Those are in turn very basic probability models, which are commonly used to model time series.

Moving average models

A moving average model represents a linear combination of lagged elements of a process, $\{\epsilon_t\}$. In other words it specifies, that the output is linearly depending on the current and various past elements of a stochastic process. Formally, a moving average process $\{X_t\}$ is defined as

$$X_t = \mu + \theta_0\epsilon_t + \theta_1\epsilon_{t-1} + ... + \theta_q\epsilon_{t-q} = \mu + \sum_{i=0}^{q} \theta_i\epsilon_{t-i}, \tag{1.4}$$

where μ is the mean of the process, $\theta_0, ..., \theta_q$ are the model's parameters, q is the order of the model and $\epsilon_t, ..., \epsilon_{t-q}$ are *white noise*[9] error terms, which are considered to be independent, identically distributed (iid) variables sampled from a Gaussian normal distribution. It is usually denoted as MA(q).
Instantly one can derive interesting statistical properties of this model:[10]

$$\mathbb{E}(X_t) = \mathbb{E}\left(\mu + \sum_{i=0}^{q} \theta_i\epsilon_{t-i}\right) = \mu + \sum_{i=0}^{q} \theta_i\mathbb{E}(\epsilon_{t-i}) = \mu \tag{1.5}$$

$$\operatorname{var}(X_t) = \operatorname{var}\left(\mu + \sum_{i=0}^{q} \theta_i\epsilon_{t-i}\right) = \sum_{i=0}^{q} \theta_i^2 \operatorname{var}(\epsilon_{t-i}) = \sigma^2 \sum_{i=0}^{q} \theta_i^2 \tag{1.6}$$

$$\gamma(\tau) = \operatorname{cov}\left(\mu + \sum_{i=0}^{q} \theta_i\epsilon_{t-i},\ \mu + \sum_{j=0}^{q} \theta_j\epsilon_{t-\tau-j}\right) = \sigma^2 \sum_{i=0}^{q}\sum_{j=0}^{q} \theta_i\theta_j\delta_{j,i+\tau}. \tag{1.7}$$

$\delta_{u,v}$ represents the *Kronecker delta*, that is 1 for $u = v$ and zero otherwise. Given this property

[8] In statistical hypothesis testing formally referred to as *type I errors*.
[9] The term white noise describes a random signal with constant power spectral density.
[10] For the full derivations see [39].

yields

$$\gamma(\tau) = \sum_{i=0}^{q-\tau} \theta_i \theta_{i+\tau}, \tag{1.8}$$

which effectively is an *autoconvolution* of $\{\theta_i\}$. One can take from equation 1.8 that the autocovariance is obviously zero for lag $\tau > q$. This important feature can be used to estimate the model order q using the *sample autocovariance* function $\hat{\gamma}(\tau)$ for an actual time series $x_1, ..., x_n$, which is defined as

$$\hat{\gamma}(\tau) = \sum_{i=1}^{n-\tau} (x_i - \bar{x})(x_{i+\tau} - \bar{x}). \tag{1.9}$$

By examining the sample autocovariance (or the sample autocorrelation $\hat{\rho}$) for $\tau = 0, ..., n-1$ one is able to determine where it becomes negligibly different from zero, which is only the case for lags of $q+1$ or higher.

Autoregressive models

As the very name already implies, these models specify that the ouput value depends linearly on its own previous values and on a stochastic term. They are denoted as AR(p) and defined as

$$X_t = c + \varphi_1 X_{t-1} + \varphi_2 X_{t-2} + ... + \varphi_p X_{t-p} + \epsilon_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \epsilon_t. \tag{1.10}$$

Whereby $\varphi_1, ..., \varphi_p$ constitute the parameters of the model, c is constant and ϵ_t white noise. While an MA(q) is always stationary, autoregressive models may not as they can contain a unit root. If they are, however, statistical properties of the model can be derived likewise, although higher order p constitute substantial complexity.[11][39]

$$\mathbb{E}(X_t) = \mathbb{E}\left(c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \epsilon_t\right) = \frac{c}{1 - \sum_{i=1}^{p} \varphi_i} = \mu \tag{1.11}$$

$$\gamma(\tau) = \begin{cases} \sigma^2 + \sum_{i=1}^{p} \varphi_i \gamma(i) & \text{if } \tau = 0 \\ \sum_{i=1}^{p} \varphi_i \gamma(\tau - i) & \text{if } \tau \in \mathbb{N}^+. \end{cases} \tag{1.12}$$

Combination to ARMA models

It has to be noted that AR and MA model different types of stochastic dependence. While AR processes comprise a Markov-like behavior, MA processes combine elements of randomness from the past using a moving window. The combination of these to a ARMA(p,q) model was firstly described in 1951 by Peter Whittle and follows a simple concatenation s.t.

$$X_t = c + \epsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}. \tag{1.13}$$

[11] In order to do so the AR(p) process is actually turned to an MA(∞) process by recursively applying formula 1.10. For the full derivations see [39].

They rose to prominence in the 1970's, when George E. P. Box and Gwilym M. Jenkins postulated an effective iterative method for fitting of the models - the *Box-Jenkins* method. This procedure involves examining the sample autocorrelation functions to decide the order of MA or AR terms and further covers the elimination of deterministic trends or outliers.[39]

1.2.3 ARIMA models

An *autoregressive integrated moving average* (ARIMA) process can be seen as a generalization of an ARMA process. The 'integrated' corresponds to an initial differentiation of the time series of d times, which is reducing the non-stationarity. In order to make the formal distinction more intuitively accessible the definition of an ARMA(p,q) model given equation 1.13 can be equivalently written as

$$X_t - \varphi_1 X_{t-1} - ... - \varphi_{p'} X_{t-p'} = \epsilon_t + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} \tag{1.14}$$

and consequently

$$\left(1 - \sum_{i=1}^{p'} \varphi_i L^i\right) X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \epsilon_t, \tag{1.15}$$

where the left part of the equation represents the autoregressive model and the right part the moving average model. L represents the so-called *lag operator* that produces the previous element of a time series when applied to the current one (i.e. $LX_t = X_{t-1}$ for $t > 1$). In case of non-stationarity due to the presence of a unit root (a factor $(1 - L)$) of multiplicity d[12] in the characteristic polynomial $\left(1 - \sum_{i=1}^{p'} \varphi_i L^i\right)$ it can be written as

$$\left(1 - \sum_{i=1}^{p'} \varphi_i L^i\right) = \left(1 - \sum_{i=1}^{p'-d} \phi_i L^i\right) (1 - L)^d. \tag{1.16}$$

Combining this with equation 1.15 yields

$$\left(1 - \sum_{i=1}^{p'-d} \phi_i L^i\right) (1 - L)^d X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \epsilon_t, \tag{1.17}$$

which is commonly referred to as ARIMA(p, d, q) process. The key aspect when estimating the latter is to successively differentiate the data until the time series appears stationary given a specific value of d.[39]

1.2.4 Machine learning concepts for time series prediction

The classical statistical models (e.g. ARMA), constituting the established approach in predicting time series, more and more compete with machine learning techniques as support vector machines, neuronal networks or decision trees. While those methods primarily focused on the classification domain at first, their applications extended also to regression tasks and they proved

[12] In case the characteristic polynomial of a stochastic process has 1 as a root, commonly referred to as *unit root*, the process is integrated of order one. If 1 is a multiple root of order d, the process is integrated of order d.

their efficiency, though some perform better than others.[2]
Ahmed *et.al.*(2010) conducted an empirical comparison of machine learning models applied on the *M3* time series competition data of the *International Institute of Forecasters.*[13]
The comparison sought to quantitatively determine the efficiency of selected methods, which included *multilayer perceptron*, *Bayesian neural networks*, *K-nearest neighbor regression*, *regression trees*, *support vector regression*, *generalized regression neural networks* (also called kernel regression), *radial basis functions* and *Gaussian processes.* As a measure of error served the symmetric mean absolute percentage error (SMAPE) and the average rank $\bar{R}$, and for the case that no special pre-processing was conducted, the results are as illustrated in table 1.2[14]. While SMAPE is intuitively comprehensible and given in equation 1.18, the average rank $\bar{R}$ requires more explanation. The crucial aspect is to determine whether some methods outperform others at specific tests significantly, although they might not in the overall performance. After computing the performance rank of each method q on each time series p, $R_q(p)$ (1=best, 8=worst), the average performance rank of each model $\bar{R}_q$ can be obtained by averaging $R_q(p)$ over all p.

$$\text{SMAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{(|\hat{y}_i| + |y_i|)\frac{1}{2}} \tag{1.18}$$

The $\alpha\%$ confidence intervals given the number of methods Q as well as the amount of time series (i.e. tests) P are defined as[2]

$$\bar{R}_q \pm 0.5 q_{\alpha Q}\sqrt{\frac{Q(Q+1)}{12P}} \tag{1.19}$$

where $q_{\alpha Q}$ is the upper α percentile of the range of Q independent standard normal variables.[15]

Model	SMAPE	Mean rank
MLP	0.0857	2.78
GP	0.0947	3.17
SVR	0.0996	4.19
BNN	0.1027	3.42
KNN	0.1035	5.10
RT	0.1205	6.77
GRNN	0.1041	5.24
RBF	0.1245	5.33

Table 1.2: Comparison of selected machine learning methods for time series forecasting on M3 data.

Another concept, which has received attention more recently, is the *Conditional Restricted Boltzmann Machine* (see section 4.3), which is especially suitable for the prediction of multivariate time series.[31]

[13]The M3 data has become an important benchmark for testing forecasting models.
[14]The results are based on testing of a number of methods $Q = 8$ on an amount of time series $P = 1045$.
[15]For the detailed derivation and the preconditions for this test see Koning *et.al.*(2005)[30]

1.3 Wind turbine

Since this work addresses analytics applied to *wind turbines* and their data (for the detailed problem specification see section 2) - the following section shall give a brief overview of the major aspects. A complete description including formulas or derivations would exceed the scope of this work. Thus, it shall be referred to the extensive and detailed treatises of Hau (2008) and Gasch *et.al* (2005) on which this overview is based if not stated otherwise. [25][22]

1.3.1 Construction design

As all wind turbines are energy converters, regardless of constructional layout they share the property of transforming the *kinetic* energy of moving air masses into *mechanical rotationa*l energy. However, one generally distinguishes based on the exploited aerodynamic principle, *resistance* or *buoyancy*. The former comprises a low level of efficiency[16] and therefore has no relevance for technical applications except anemometers. The buoyancy runners in turn can be differentiated with respect to the orientation of their power train. *Vertical* turbines do not need any yaw control for wind direction tracking, but constitute severe disadvantages when applied to greater dimensions. Consequently only buoyancy runnners with *horizontal* power train axis have been widely accepted. Figure 1.5 illustrates two different wind turbines with vertical axis, while the one on the left, the *Savonius* rotor, is based on resistance the other, a two-blade *Darrieus* rotor, on buoyancy.

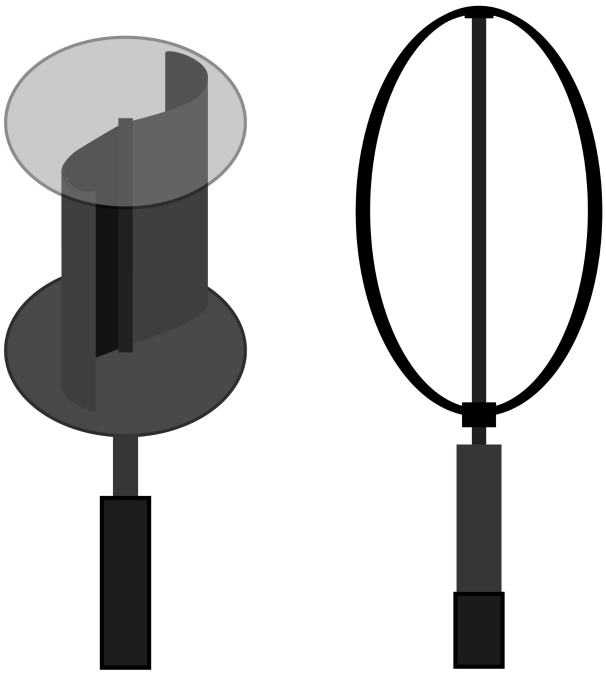


Figure 1.5: Wind turbines with vertical axis

The prevailing design principle of wind turbines with horizontal axis (HAWT) is a blade oriented

[16] In fact it is easy to see that peripheral speed of resistance runners - the relevant component of the resulting force - can not exceed wind speed and is consequently limiting potential power output.

architecture of the rotor as depicted in figure 1.6[25].
The figure also schematically illustrates the principle components of a turbine, albeit their design as well as their setup may vary in practical construction.

Rotor

The rotor constitutes the centerpiece of the turbine, converting the wind energy into kinetic energy of rotation.
The *rotational speed* Ω represents a crucial design parameter and is connected to wind speed v over the so-called *tip-speed ratio* λ. This is in turn the proportion between the velocity of circulation at the end of a *blade* and the inflowing wind speed, s.t. $\lambda = \Omega R/v$ with the length of the blade R. The tip-speed ratio serves as basis for the aerodynamic design of blades as well as for the construction of the *nave*, since it strongly influences the arising torsional moment.
The number of blades is usually set to three, because, due to a more even distribution of the inertia and aerodynamic forces over the swept area, the operation is smoother and thus reduces the stress on the components. The blades themselves are nowadays made of carbon or glass fiber reinforced plastic and differ in dimensions and geometric profile, depending on a.o. usage and design parameters.
There are two different concepts for *power limitation*, to prevent the turbine from operating beyond their design. The first approach, already introduced in the 80's, is based on the *stall effect* and was applied to rotors that operated at constant rotational speed. At high wind speeds the angle of attack at the blade would get too big and consequently lead to stalling. Due to the disadvantageous inertia of the concept, automatic controls that *pitch to stall* also exists. The second approach, *pitch to feather*, similarly relies on adjustment of the rotor blade, but in the opposite direction. Since the air stream is in contact throughout the regulation, this limitation control operates smoother, though the angle of adjustment is bigger.

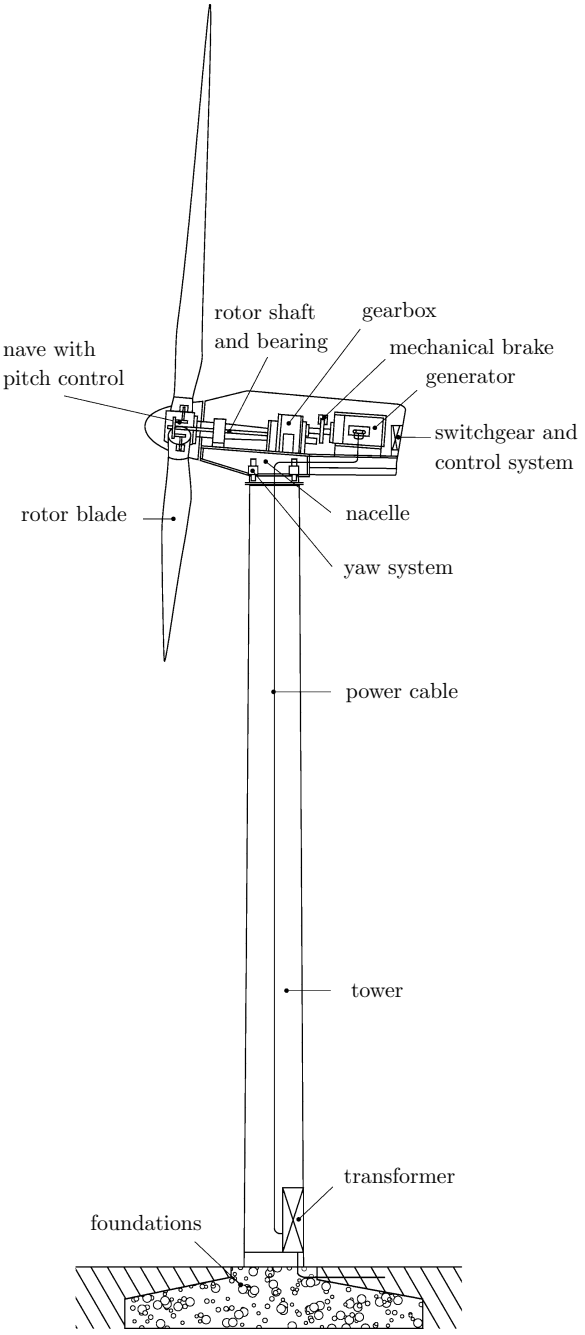


Figure 1.6: Schematic structure of a wind turbine with vertical axis.

Nacelle

Inside the nacelle all remaining parts of the drive train are located, that usually includes the *bearings* of the shaft, *gearbox*, *couplings* and *brakes* as well as the *generator* and the *control system*.

The arrangement can broadly differ depending on manufacturer or conditions of use. Nevertheless one distinguishes between disintegrated or integrated architectures. A very prominent example of the former are the gearless drop-shaped wind turbines of the company ENERCON. In this case the rotor shaft is directly attached to the generator omitting any gearbox, brakes or couplings. The generator's spinning rate is then equivalent to the rotor's, which is compensated by increased diameters and a higher number of magnets.
The nacelle also houses sensors measuring wind speed or vibration of the shaft and the *yaw control*, that moves the turbine into the correct direction by adjusting the azimuth angle.

Control system

The control system comprises the *switchgear* and the *programmable logic controller*(PLC), that controls and monitors the system. Commonly a supervisory control and data acquisition (SCADA) system exists on top, which allow the remote monitoring and process data acquisition (such as wind speed, rotational speed, electrical flows, azimuth angle etc.).

Tower

The height of the tower usually depends on the geographic conditions of use and on the size of the rotor blades. Since wind speed is increasing with height and hub heights beyond the *turbulent bottom boundary layer* generate higher energy output the height constitutes a major profitability criteria.
While *off-shore* wind turbines are typically lower, comprising a ratio between height and diameter of 1.0 to 1.4, *on-shore* turbines exhibit a ration between 1.2 and 1.8. The maximum height is further limited by the availability of cranes for the erection. Modern installations use towers about 65m high.
Common constructions concepts include *conical tubular steel towers* or the use of high-performance concrete. Hybrid versions of pre-stressed concrete and steel have the advantage of combining the good *structure damping* of concrete towers with the easier assembly of steel towers. They furthermore constitute a solution to transport issues arising with big steel segments.

1.3.2 Costs and profitability

The review of costs and profitability of wind turbines can naturally be only a snapshot, as they are subject to constantly changing economical, juridical and technical circumstances (e.g. demand, manufacturing processes). Nevertheless, the costs clearly depend on some characteristic properties. Conventional considerations link the construction costs to the performance and therefore employ the relationship of building costs per kilowatt. The dimensions of the rotor and the height of the tower are also commonly used as parameters, though the ratio between investment costs and energy production is still important for profitability assumptions. Furthermore, a comparison to conventional installations is only valid if the operating life per year is comparable.
The construction costs, however, are said to amount for $1.0 * 10^6$ EUR/MW for on-shore and about $1.5 * 10^6$ EUR/MW for off-shore projects[57], which has also been shown analytically

through Hau (2008) by considering component costs and cubic capacity.

Modern inland wind turbines have powers that range from 2.5 MW up to 6.0 MW, off-shore turbines are built up to 8.0 MW. In order to compete with traditional/fossil sources in regard to power production[17], the erection of a multitude of turbines becomes obligatory (e.g. the *East Anglia THREE* is a planned wind park in the Northern Sea comprising 172 turbines producing 1200 MW[48]).
The operating costs can differ significantly, depending on the location and the technological standard of the turbine. Gasch *et.al.* (2005) indicate costs of energy (COE) of 6.5 ct/kWh, according to Walford (2006)[57] modern turbines can create COE of 3.5 to 6 ct/kWh.
Investment in renewable energy production has been strongly supported by governments in the last couple of years. While at the beginning this was effectuated in form of funding, today *statutory feed-in compensation* prevails in Germany through the *Erneuerbare-Energien-Gesetz* (EEG). This law basically sets a yearly fixed minimum energy price for each kWh produced by a wind turbine, differing between on- and off-shore locations (see table 1.3 for the 2016 feed-in compensations)[11].

Type	First operation	Basic compensation [ct/kWh]	Initial compensation [ct/kWh]
On-shore	2016	4.58	8.53
Off-shore	2016	3.90	15.40

Table 1.3: Statutory feed-in compensation in DE 2016 according to EEG. The initial compensation is valid for 12 years.

1.3.3 Operation and maintenance

Operation and maintenance (O&M) constitute a major portion of costs, representing around 75-90% of a turbine's overall costs based on a 20-year life cycle - although the costs may decrease with higher turbine dimension. The costs can be separated into categories of *operations*, *scheduled maintenance* and *unscheduled maintenance*, whereas the latter is the most difficult to predict and can represent 30-60% of the total O&M. The variety and complexity of the turbine's components encourages malfunctions, that may lead to a shut-down of the whole turbine.[57]
The related cost of a breakdown can in turn be divided into *direct* and *indirect* costs: The direct ones include labor and equipment costs that arise through the repair or replacement. The latter are associated with the lost revenue due to downtime. They depend on the total repair time involving any processes connected to the repair, such as acknowledgment, diagnosis or the actual replacement activity. The costs also depend on the economic and meteorological circumstances. That is, they depend on energy price and appropriate wind speed downtime.[57] Another influence on the costs represents the arrangement of components in the turbine. Specifically, in case of an integrated design, a breakdown can cohere with the replacement of the whole rotor, which evidently correlates with increased expenditure.

[17] A typical coal power station can reach up to 1000 MW per block-unit. The conceptual study of a 600 MW black coal power station in Nordrhein-Westfalen calculated construction costs of $478.8 * 10^6$ EUR[35].

Reduction of maintenance costs

Walford (2006) has postulated the following measures to prevent and reduce maintenance costs[57], though they are partially valid for any industrial plant:

- **Improving system reliability**
  This involves identifying critical components in order to focus on their monitoring, inventory and logistic issues they may comprise. A further essential action to improve stability is to see failure as opportunity for improvement. That is, one determines the root cause of a problem and evaluates the defective part for redesign.

- **Logistics plan**
  A logistics plan can facilitate the efficiency of the repair process, optimizing the schedule of necessary tasks or the execution of sub-processes.

- **Improving maintainability**
  Maintainability is commonly referred to the ease and the efficiency of performing maintenance. It may include improving accessibility to components that are prone to defects or easements for the process of replacement as markings. The perspective of maintenance should be considered even in the very early stage of design.

- **Condition monitoring**
  Condition monitoring systems more and more manifest themselves in *preventive maintenance concepts*. The basic idea is to measure the behavior of the turbine or its individual components to provide diagnostic information. This significantly facilitates and optimizes the scheduling of repair work as well as allows the prediction and consecutive avoidance of component failures that may lead to a breakdown and lost revenue. Condition monitoring can either be on- or offline, whereby the former can be incorporated into SCADA systems. Common measured data include vibration, temperature, voltage levels or performance related information and are usually provided as time series.

2 Problem Statement

The problem statement shall firstly provide a profound motivation for the content of this work and secondly states and concisely describes the objectives of the thesis.

2.1 Motivation

The motivation for this thesis consists of three considerations, of which two are devoted to the specific problem case and the remaining to machine learning in analytics.

1. The prognosis of overall energy demand expects a worldwide increase by a third by 2040 according to the *World Energy Outlook*, whereas the entire growth is caused outside of OECD countries[29].[18] Due to current structural change in energy systems (*energy transition*), the construction of wind turbines in order to feed this increasing demand in a sustainable way rose to prominence in the last couple of years. In fact the produced wind energy in Germany reached the amount of 85 TWh in 2015 (see figure 2.1). While energy production using fossil energy sources declined, wind energy rose by 50% with respect to the preceding year (see figure 2.2). The total installed capacity worldwide is expected to more than double by 2020[21].

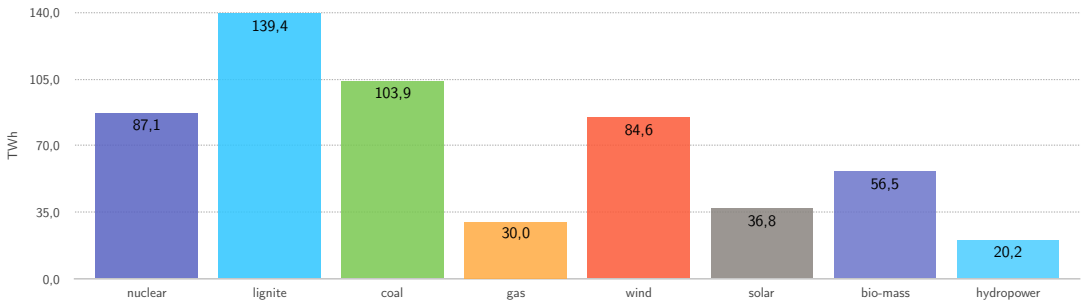


Figure 2.1: Electricity generation in Germany 2015 by source of energy.[12]

2. As outlined in the introductory sections 1.3.2 and 1.3.3 costs of operation and maintenance of wind turbines can crucially affect profitability of the investment. Constituting up to 60% of maintenance costs, unscheduled maintenance is in turn a major cost factor. As a consequence *condition monitoring systems* (CMS) more and more manifested themselves to measure behavior of the turbine or individual components in order to provide diagnostic information. This facilitates the scheduling of repair work as well as allows the prediction

[18] Because of higher efficiency in consumption the EU will reduce consumption by 15% , Japan by 12% and the U.S. by 3%.[29]

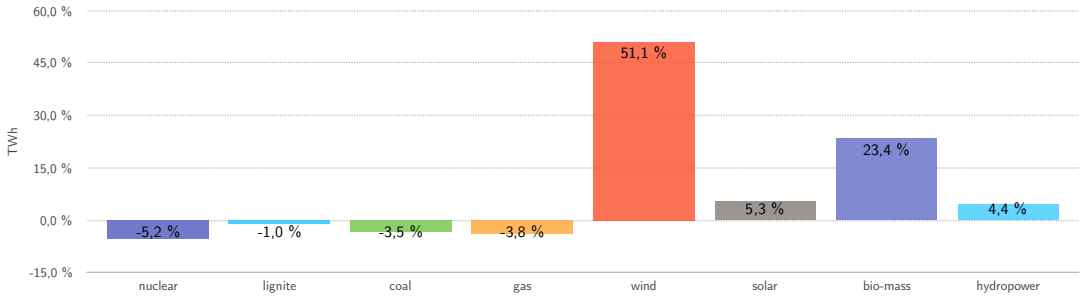


Figure 2.2: Change of electricity generation in Germany from 2014 to 2015.[12]

and consecutive avoidance of component failures that may lead to a breakdown (*preventive maintenance*).[57]
CMS in wind turbines typically measure vibration, acoustic emission, strain or electrical effects. However, the usual approach and related research likewise mainly focuses - simply put - firstly on the technical implementation of measurement, secondly on signal processing in form of *Fourier transformation* or *Wavelet transformation* and thirdly on the application of *physical models*, the technical process shall obey.[21][24] Subsequently and given the model one computes from a set of observations the causal factors that produced them, which is called an *inverse problem*. Alternatively one may apply a more abstract model and train it with machine learning, which has been done (e.g.[45][36]) to a lesser extent, although they might capture more non-trivial patterns or effects in the data.
Furthermore the temporal horizon of the prediction (i.e. the earliness of the anomaly detection with respect to the moment when the anomaly is affecting the system) often lack a more thorough investigation.

3. The modeling and prediction of time series takes a central role in predictive analytics. Established methods such as ARIMA may not always be applicable (see introductory section 1.2). According to literature the machine learning concept of a *restricted Boltzmann machine* (RBM) has proven usefulness in different domains[31][32], albeit the applicability has only been investigated to a limited extent so far.

Since the amount of installed capacity of wind energy production is increasing significantly in future and profitability is more and more determined by unscheduled maintenance costs, evolved CMS technology receives a pivotal role (which holds in general for various technical applications). As machine learning is currently not in the focus of CMS-related research, the content of this thesis shall firstly contribute to the closure of this gap. Secondly, giving the fact that the RBM has not been extensively investigated yet, the work examines another specific application for the model. Finally we seek to present the applied approach, despite the technical nature of this case, as valuable procedure for any time series prediction as often occurring in the domain of predictive analytics.

2.2 Objectives

The major objective of this work is to apply a restricted Boltzmann machine to a large amount of performance-related time series data provided by on-shore wind parks in order to predict the occurrence of events. To allow evaluation of the performance the concept will be compared with another machine learning algorithm - the *support vector machine*. Due to the different character of the methods a secondary goal shall be to develop an appropriate setting of implementation. The methods' performance shall be compared in regard to their capability of detecting events based on time series data as well as to the earliness of detection. However, there will be no categorization of events, whether being a breakdown or of a different kind, but the outcome of the work shall serve as a basis for further research in preventive maintenance with RBMs and as a contribution to the investigation of applicability of the concept of RBM to time series data in analytics.

For this purpose the remainder of this thesis is organized as follows. Part 3 describes the available data in detail and by its exploration shall deliver a valid indication for the usage of machine learning algorithms. Part 4 firstly gives a general overview of the approach and secondly describes the methods and their peculiarities in appropriate detail. Part 5 presents and discusses the results, while part 6 concludes the thesis and gives implications for future work in practice and research likewise.

3 Data Exploration

As already outlined in section 1.2 time series are, because of the *temporal autocorrelation* between data points, formally speaking a special type of *sequential data*, which are in turn *ordered data*. That is, records of both, sequential and time series, are associated to a time, though data points of time series comprise a relationship in time and often are separated through a constant time interval.[53] The data provided in this very case are of two different kinds: For each wind turbine of each wind park the implemented SCADA system is measuring (1) performance related time series data in a time interval of 10 minutes and (2) any occurring technical event (breakdown, defect) in form of sequential data. An overview of the characteristics of the provided data is given in table 3.1.

Number of parks	1
Number of turbines	3
Time interval	May to July 2015
Number of records each turbine	13140

Table 3.1: Data description.

This section will give a brief description of the involved data, necessary pre-processing steps and examines the stationarity of the time series in order to investigate the appropriateness of established modeling techniques (ARIMA).

3.1 Time Series Data

The time series data obtained from the SCADA system comprises 35 attributes for each wind turbine, that are grouped in parks all over Germany. Not considering turbine ID, park ID, date, time stamp and working hours so far (time in operation), only 11 are usable, whereas the rest of the attributes are not changing over time. The quantity of data points for the respective wind turbine depends on activation and amounts to 52560 points for each year.
Concretely the SCADA system provides the cumulated maximum, minimum and average value every 10 minutes for *wind speed* (v_w), *rotational speed* of the rotor (v_{rot}) and *power output* (P). Furthermore it measures *azimuth angle* of the nacelle ($\measuredangle$) and performed mechanical work so far (W) (see table 3.2). As one can see this data is undeniably *low-dimensional*, so that an enhancement in terms of *feature engineering* will be applied in this work (see section 4.2).
While one typically takes basic statistics of continuous data such as measures of spread (variance and range) or location (mean and median) in order to obtain an insight into the characteristics

$\overline{v_w}$	$max(v_w)$	$min(v_w)$	$\overline{v_{rot}}$	$max(v_{rot})$	$min(v_{rot})$	$\overline{P}$	$max(P)$	$min(P)$	∢ azimuth	W
8.8	10.8	6.4	32.32	35.4	27.30	202	340	117	-511	12107520
8.6	10.2	6.9	31.52	34.31	28.06	184	244	130	-512	12107552
8.5	10.1	6.3	32.41	34.86	26.97	202	282	110	-512	12107586
8.4	10.7	6.4	30.6	34.46	25.81	169	251	99	-511	12107607
...	...	...	...	...	...	...	...	...	...	...

Table 3.2: Time series of a wind turbine.

(see Tan (2014)[53]), in this case we also investigate the stationarity of the time series.

3.1.1 Stationarity testing

For this reason at first eventual seasonality or trend (invalidating first-order stationarity) will be modeled via moving averages (MA) and eliminated from the time series using the `decompose` method provided in programming language `R`.[19] Consequently two second-order statistically rigorous hypothesis tests are applied.

Priestley-Subba-Rao (PSR) second-order testing

The Priestley-Subba Rao (PSR) test examines the homogeneity of a set of spectral density function (SDF) estimates over time or frequency or both.[43] The `stationarity` method of the `R` package `fractal`, which has been used in this thesis, is a slightly differing implementation of the original algorithm proposed by Priestley *et.al.*(1969). In fact the localized lag window SDF estimators have been replaced by averages of *multitaper* SDF estimates.

SDF estimator	Multitaper
Number of (sine) tapers	5
Centered	TRUE
Number of blocks	13
Block size	1019
p-value for T	0
p-value for I+R	4.609947e-05
p-value for T+I+R	6.375234e-12

Table 3.3: Results of PSR second-order test for rotational speed.

The output of the method applied to 3-month rotational speed data can be seen in table 3.3. Since the *p-value* for T, which is measuring variation over time, is essentially zero, there is strong evidence to reject the null hypothesis of stationarity.[20]

[19] It has to be stated, that decomposing time series is without doubt non-trivial and there are multiple different ways to do so (e.g. decomposition using *local polynomial regression fitting*).

[20] The p-value indicates whether the statistical summary of the sample is similar or more extreme as for actual observed results and $0 \leq p \leq 1$. The closer to 0, the more one should reject the null hypothesis. For decision support one usually establishes a significance level α, such as $\alpha = 0.05$, below which the null hypothesis will be rejected.

Wavelet second-order testing

Secondly we apply a wavelet-based multiple hypothesis test for investigating secondary-order stationarity as outlined in Nason (2013)[38]. Concretely the method firstly computes an evolutionary wavelet spectral estimate and consequently computes the *Haar* wavelet coefficients for each scale of the spectral estimate. Since large coefficients indicate non-stationarity, the test secondly investigates if any Haar coefficient is large enough to reject the null hypothesis.

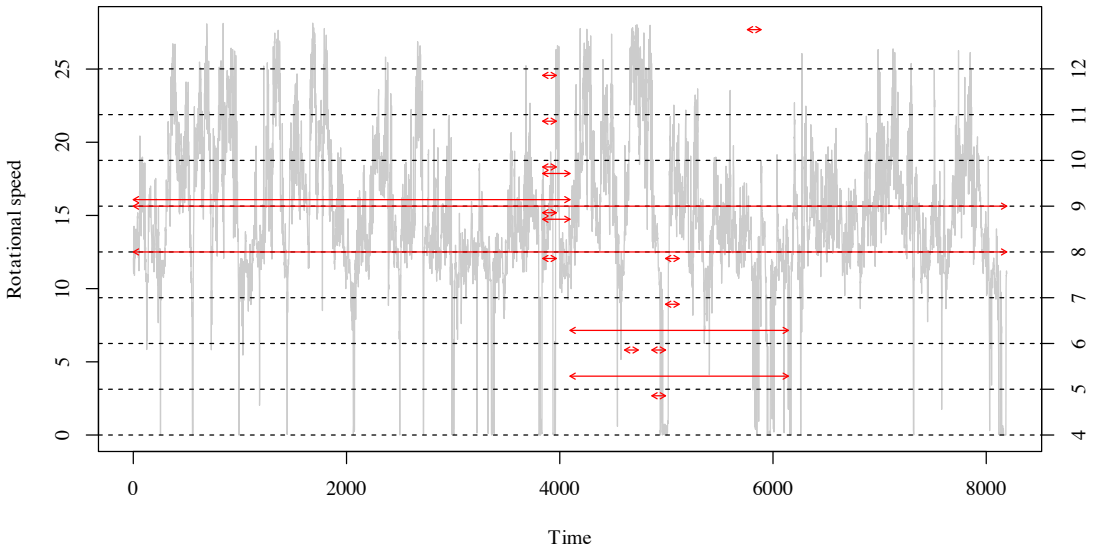


Figure 3.1: Wavelet-based test of secondary-order stationarity.

Applied to the very same set of 3-month rotational data, using the `R` function `hwtos2` with $\alpha = 0.05$ and *Daubechies wavelets*, provides again strong evidence to reject the null hypothesis of stationarity. In fact 1270 hypothesis have been tested and the measurement of significance of the multiple hypothesis test (outlined in section 1.2.1) has been twofold. The test rejects 18 hypothesis against *false discovery rate* (FDR) assessment and 7 according to Bonferroni correction. Figure 3.1 shows the plot of the rotational data and each red arrow corresponds to one of the FDR non-stationarities identified by the test. The length of the arrow corresponds to the scale of the wavelet coefficient whose null hypothesis was rejected and the location of that wavelet coefficient is fixed by the mid-point of the arrow. The summary of the test can bee seen in table 3.4.

α	0.05
# hypothesis tests	1270
# rejections (FDR)	18
p-value (FDR)	0.0006882402
# rejections (Bonferroni)	7
p-value (Bonferroni)	3.937008e-05

Table 3.4: Results of wavelet-based test for rotational speed.

The time series in any case is evidently not stationary, such that the usage of modeling or prediction methods, which are premising at least weak stationarity, would be certainly inappropriate

and thus the application of machine learning algorithms seems more promising.

3.2 Sequential Data

The provided sequential data comprises 7 attributes, which clearly characterize each event. As exemplified in table 3.5 these include the *serial number*, the *date* of occurrence, description of the incident and the *operating state* of the turbine.

Serial #	Date	Notification	Status	Add. info	warning info	Add. info
782241	23.05.2016 20:00	SCADA S:0/0 W:58/2		operating	failure lubr.sys.	no press. (90)
782241	22.05.2016 20:52	SCADA S:0/0 W:58/2		operating	failure lubr.sys.	no press. (90)
782242	03.08.2015 17:02	SCADA S:0/1 W:90/91		starting	prot. switch trigg.	tower fan (14)
...	...	...	...	...	...	...

Table 3.5: Service notifications of a wind park.

4 Methods

This section shall describe the *concept*, *preliminary steps*, and the *applied methods* in order to accomplish the objectives given in the problem statement (see section 2).

4.1 Concept overview

As mentioned the investigation of the restricted Boltzmann machine takes a central role. However, for the purpose of relative performance measurement a different model shall be implemented. The *support vector machine*, that has been referred to already as potential machine learning alternative for time series prediction (see section 1.2.4), will serve for comparison. Consequently an appropriate conceptual layout for the application of the prediction models is required, which has to cover three major issues: **training**, **testing** and **verification of results**. Since the algorithms differ in their methodology, so does the layout as illustrated in figure 4.1 and 4.2. Initially the data has to be divided in both cases into training and testing examples, where the former typically amounts to 70% of the sample.
As depicted in section 4.4 in more detail, a support vector machine is a supervised learning model, which needs the corresponding *label* for each training example, indicating the correct classification. In this case the label refers to the occurrence of an event in a defined time horizon t. After training the SVM serves as a linear classifier by predicting, whether an event will occur in time horizon t given a testing example or not.

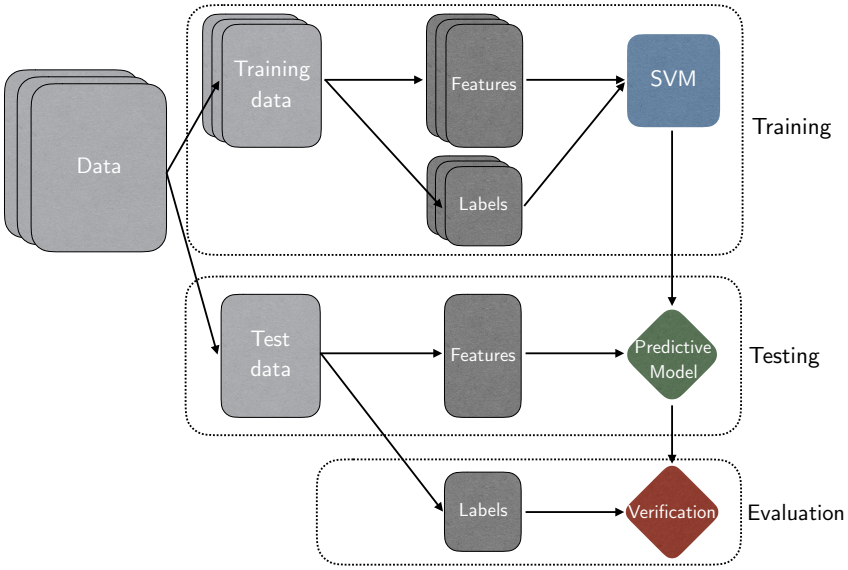


Figure 4.1: Conceptual layout for the application of a SVM.

On the contrary the RBM (see section 4.3) shall not be implemented as a linear classifier per se, but after training estimates the likelihood of the occurrence of a testing example. Subsequently, given a pre-defined threshold τ_{th} for the deviation between actual and expected values, a prediction can be made, based on **the assumption that unlikely or anomalous data coheres with the occurrence of events**.
Finally the results are evaluated with the actual labels for both algorithms.

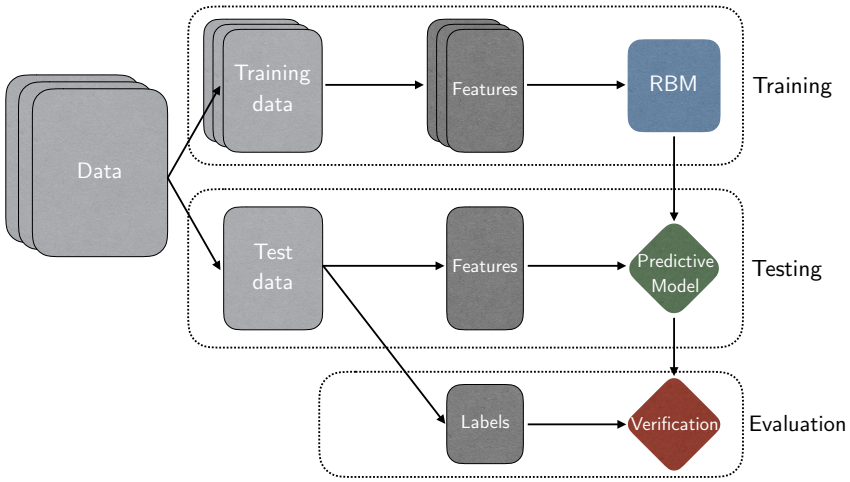


Figure 4.2: Conceptual layout for the application of a RBM..

The algorithms expect the features in a distinct shape, so that the input data has to be formatted appropriately. Furthermore, as stated in the data exploration section 3, we seek to qualitatively augment the data by means of feature engineering.

4.2 Data Preparation

4.2.1 Selection and formatting

Initially the data sets of time series and sequential data have to be combined and formatted in order to be usable jointly as input for the proposed algorithms. Therefore we had to overcome the following issues:

- To combine the different types of ordered data, the sequential data set has to be transformed to time series data by assigning each sequential record to the respective time series elements. However not all turbines can be connected to events.
- Given the objective to predict the occurrence of an event under time horizon t, the target values of the classification have been established through aggregation as illustrated in figure 4.3. For a better understanding the operation can be seen in pseudo-code 1.

The data manipulation was performed using programming language `R`, any further programming was conducted in `Python`.
The investigation of all wind turbines and parks ($>$ 300 units) was limited by the reduced

accessibility of the control system linked to the data source, so that we had to select data from three of the most promising turbines, i.e. units with a high amount of events to predict.

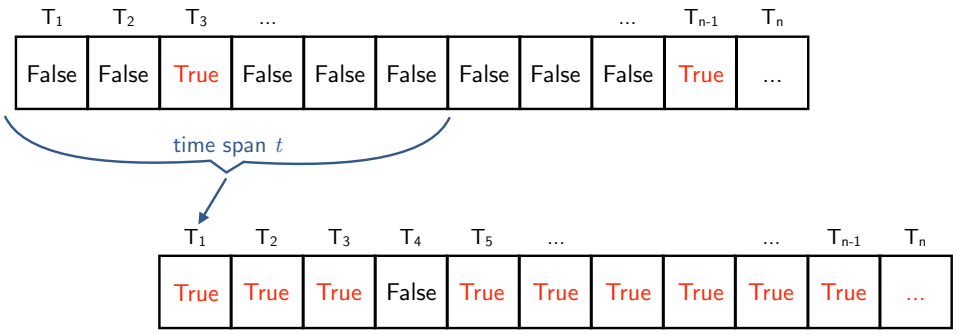


Figure 4.3: Target value (label) aggregation for a time span t.

Algorithm 1 Label aggregation.

Input: Data label vector l of size n, time horizon of steps t

```
for i, ..., (n − 1) do                          ▷ No aggregation for last element
    for j = i, ..., min(n − 1, t) do            ▷ Minimum of t and remaining steps in array
        if l[j]==True then
            l[i] ← True
```

4.2.2 Feature Engineering

To enlarge the amount of attributes in the time series data set, the four *statistical moments* and the *standard deviation* of each primary attribute has been computed and added to the set. In statistics, a moment is a quantitative measure of the shape of a probability density. The k-th moment is in general defined as

$$\mu_k := \mathrm{E}\left((\mathrm{X} - \mu)^k\right) \tag{4.1}$$

where E(X) represents the *expected value* of a variable. While the zeroth moment is the total probability, the first moment is the *mean*, the second is the *variance*, the third the *skewness* and the fourth is the so-called *kurtosis*. The standard deviation is defined as $\sigma = \sqrt{\mathrm{Var(X)}}$.[42]
To obtain the moments for each point t in time the *unbiased sample variance* (eq. 4.2), *sample excess skewness* (eq. 4.3) and *sample kurtosis* (eq. 4.4) have been computed for each based on the t data points before. The named sample moments are obtained through

$$\sigma_t^2 = \frac{1}{t-1}\sum_{\tau=1}^{t}(x_\tau - \overline{x}_t)^2, \tag{4.2}$$

$$b_t = \frac{\frac{1}{t}\sum_{\tau=1}^{t}(x_\tau - \overline{x}_t)^3}{\left[\frac{1}{t-1}\sum_{\tau=1}^{t}(x_\tau - \overline{x}_t)^2\right]^{3/2}}, \tag{4.3}$$

$$k_t = \frac{\frac{1}{t}\sum_{\tau=1}^{t}(x_\tau - \overline{x}_t)^4}{\left[\frac{1}{t-1}\sum_{\tau=1}^{t}(x_\tau - \overline{x}_t)^2\right]^{2}} - 3, \tag{4.4}$$

where $\overline{x}_t$ denotes the *sample mean* given the t data points of the sample, i.e. the currently observed fraction of the time series in our case. The computed values are added as additional columns to the dataset.

4.3 Restricted Boltzmann Machine

Restricted Boltzmann Machines (RBMs) received an increase in popularity in the last couple of years[21] with a wide range of applications for dimensionality reduction, classification, feature learning and topic modeling. As it is extensively used in the presented work this section shall give a outline over the employed mathematical structure and is based on the tutorial of Fischer and Igel[20] if not stated otherwise.

Formally, an RBM is a probabilistic undirected graphical model, which can be interpreted as generative stochastic neuronal network. As the name implies the RBM is derived from the *Boltzmann Machine*, a parametrized model representing a probability distribution, that can learn relevant characteristics of an unknown *target distribution* based on samples from this distribution.

Learning Boltzmann Machines is computationally demanding and can be simplified by restricting the network topology of the neurons such that they must form a bipartite graph. Consequently one distinguishes between two groups of units, commonly referred to as the *visible* and the *hidden* units. These groups can be thought of as being arranged in two layers, with symmetric connections between but without any connections within the layers. While the visible units represent the first layer and correspond to an observation, the hidden units model the dependencies between the components of an observation and thus can be seen as non-linear feature detectors. The RBM in figure 4.4 consists of n visible units $\boldsymbol{V} = (v_1, ..., v_n)$ and m hidden units $\boldsymbol{H} = (h_1, ..., h_m)$.

The basic idea is to learn a closed-form representation of the distribution underlying the training data and in turn make use of the generative characteristic of the model, that is sampling from a learned distribution. This property may be used to complete a partial observation by fixing some of the visible units (treat them as constants) and sample the remaining components, for example for the recognition of anomalous behavior.

[21] Initially the RBM has been invented under the name *Harmonium* by Paul Smolensky in 1986[51], and gained in significance after the development of fast learning algorithms amongst others by Geoffrey Hinton and rise of computational power in the mid-2000s.

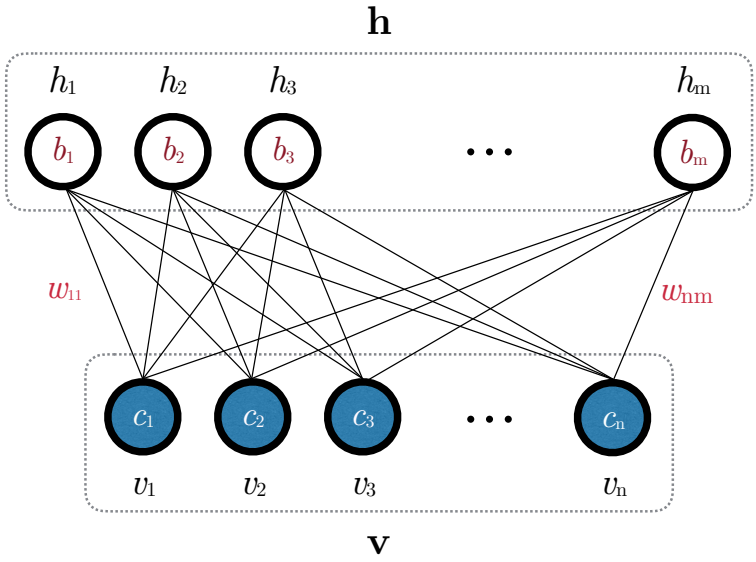


Figure 4.4: The network graph of a Restricted Boltzmann Machine.

Regarding RBMs as probabilistic undirected graphical models, also referred to as *Markov Random Fields* or *Markov Random Networks* provides access to established training algorithms and theoretical results. Unsupervised learning of MRFs, training of RBMs respectively, is commonly conducted by gradient-based maximization of the *likelihood* by adjustment of the model's parameters given the training data $D = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_l\}$.The likelihood $\mathcal{L} : \Theta \to \mathbb{R}$ maps parameters $\boldsymbol{\theta}$ from a parameter space Θ to

$$\mathcal{L}(\boldsymbol{\theta} \mid D) = \prod_{i=1}^{l} p(\boldsymbol{x}_i \mid \boldsymbol{\theta}). \tag{4.5}$$

Maximizing the likelihood is equal to maximizing the log-likelihood given by

$$\ln \mathcal{L}(\boldsymbol{\theta} \mid D) = \ln \prod_{i=1}^{l} p(\boldsymbol{x}_i \mid \boldsymbol{\theta}) = \sum_{i=1}^{l} \ln p(\boldsymbol{x}_i \mid \boldsymbol{\theta}). \tag{4.6}$$

To emphasize the consistency between the objective of modeling the unknown distribution underlying the training data and maximizing the likelihood, it can be shown as in [50] that the latter corresponds to minimizing the *Kullback-Leibler divergence* (KL divergence). The KL divergence between the unknown distribution q underlying D, which is represented by the currently observed sample, and the (current) distribution p of the RBM is given by

$$\mathrm{KL}(q \parallel p) = \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \ln \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})} = \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \ln q(\boldsymbol{x}) - \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \ln p(\boldsymbol{x}) \tag{4.7}$$

and is a (non-symmetric) measure of the difference between the actual distribution and the approximation.

Due to its complexity the computation of the maximum likelihood of an RBM is not feasi-

ble in appropriate time (which holds in general for undirected graphical models) and therefore numerical approximations have to be applied. One approach to find parameters through approximation is *gradient ascent*.
That is, iteratively updating the parameters $\boldsymbol{\theta}^{(t)}$ to $\boldsymbol{\theta}^{(t+1)}$ based on the gradient of the log-likelihood as shown in the following update-rule.

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \underbrace{\eta \left[\frac{\partial}{\partial \boldsymbol{\theta}^{(t)}} \left(\ln \mathcal{L}(\boldsymbol{\theta}^{(t)} \mid D) \right) - \lambda \boldsymbol{\theta}^{(t)} \right] + \nu \Delta \boldsymbol{\theta}^{(t-1)}}_{\Delta \boldsymbol{\theta}^{(t)}} \tag{4.8}$$

The constant $\eta \in \mathbb{R}^+$ in equation 4.8 is the *learning rate*. The term $\lambda \boldsymbol{\theta}^{(t)}$ with $\lambda \in \mathbb{R}_0^+$ represents a method called *weight decay*, which is penalizing high parameter values. The final extension $\Delta \boldsymbol{\theta}^{(t-1)}$, referred to as the *momentum*, is controlled by $\nu \in \mathbb{R}_0^+$ and seeks to avoid the occurrence of oscillations in the iterative update procedure, while additionally speeding up the learning process. The correct setting of the constants, which is in fact a crucial performance factor and highly non-trivial, is dealt with in section 4.3.4.

Returning to the mentioned categorization of units into two groups (visible and hidden) in an RBM, allows us to describe a distribution over the visible variables by means of conditional distributions. The so-called *Gibbs distribution* of an MRF describes the joint probability distribution of visible $\boldsymbol{V}$ and hidden variables $\boldsymbol{H}$. The marginal distribution of the former is consequently given by

$$p(\boldsymbol{v}) = \sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \sum_{\boldsymbol{h}} \mathrm{e}^{-E(\boldsymbol{v}, \boldsymbol{h})} \tag{4.9}$$

where Z is a *partition function* defined as the sum of $\mathrm{e}^{-E(\boldsymbol{v},\boldsymbol{h})}$ of all possible configurations, that is, a normalizing constant. $E(\boldsymbol{v}, \boldsymbol{h})$ refers to the *energy function*, computing a scalar value associated with each state of a network. As described, training algorithms are based on gradient ascent on the log-likelihood which combined with the model of 4.9 yields

$$\ln \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{v}) = \ln p(\boldsymbol{v} \mid \boldsymbol{\theta}) = \ln \frac{1}{Z} \sum_{\boldsymbol{h}} \mathrm{e}^{-E(\boldsymbol{v},\boldsymbol{h})} = \ln \sum_{\boldsymbol{h}} \mathrm{e}^{-E(\boldsymbol{v},\boldsymbol{h})} - \ln \sum_{\boldsymbol{v},\boldsymbol{h}} \mathrm{e}^{-E(\boldsymbol{v},\boldsymbol{h})}. \tag{4.10}$$

As a consequence the gradient is[22]

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{v})}{\partial \boldsymbol{\theta}} = \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} - \sum_{\boldsymbol{h}} p(\boldsymbol{h} \mid \boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}}. \tag{4.11}$$

Equation 4.11 expresses the difference between the expected values of the energy function under the model distribution and those under the conditional distribution of the hidden variables. It is easy to see that a simple calculation of the sums over all possible configurations turns out to be computational expensive, in fact exponential in the numbers of variables. A therefore common approximation is drawing samples from the distributions by applying *Markov chain Monte Carlo* (MCMC) techniques.

[22] For the full derivation of the gradient see Fischer (2014).

So far we neglected the detailed definition of parameter vector $\boldsymbol{\theta}$ and energy function $E(\boldsymbol{v}, \boldsymbol{h})$ for the sake of simplicity, we also did not define which values the units can assume.
As pictured in Figure 4.4 an RBM is - apart from it's network topology - configured by real-valued weights w_{ij} for $i \in \{1, ..., n\}$, $j \in \{1, ..., m\}$ associated with the connection between units V_i and H_j and real-valued bias terms c_i and b_j associated with each visible and hidden variable. Hence $\boldsymbol{\theta} = (W, \boldsymbol{c}, \boldsymbol{v})$. In *binary* (or *Bernoulli*) RBMs the units $(\boldsymbol{V}, \boldsymbol{H})$ can take values $(\boldsymbol{v}, \boldsymbol{h}) \in \{0, 1\}$ and considering the interpretation of the RBM as stochastic neural network the conditional probability of a variable being one is the firing rate of a neuron with the *sigmoid activation function* $\text{sig}(x) = 1/(1 + \text{e}^{-x})$, s.t

$$p(H_j = 1 \mid \boldsymbol{v}) = \text{sig}\left(\sum_{i=1}^{n} w_{ij} v_i + b_j\right) \tag{4.12}$$

and

$$p(V_i = 1 \mid \boldsymbol{h}) = \text{sig}\left(\sum_{j=1}^{m} w_{ij} h_j + c_i\right). \tag{4.13}$$

The energy function of the Gibbs distribution $p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z}\text{e}^{-E(\boldsymbol{v}, \boldsymbol{h})}$ in turn is given by

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} v_i h_j - \sum_{i=1}^{n} c_i v_i - \sum_{j=1}^{m} b_j h_j. \tag{4.14}$$

4.3.1 Learning Algorithms

Inserting the energy function of an RBM in equation 4.11 of the log-likelihood gradient of a MRF reveals that the second term (i.e. the expected values of the energy gradient under the conditional distribution of the hidden variables) factorizes w.r.t. to w_{ij} s.t.[23]

$$\sum_{\boldsymbol{h}} p(\boldsymbol{h} \mid \boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial w_{ij}} = p(H_j = 1 \mid \boldsymbol{v}) v_i = \text{sig}\left(\sum_{i=1}^{m} w_{ij} v_i + b_j\right) v_i. \tag{4.15}$$

Rewriting the first term of equation 4.11 as $\sum_{\boldsymbol{v}} p(\boldsymbol{v}) \sum_{\boldsymbol{h}} p(\boldsymbol{h} \mid \boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}}$ the derivative of the log-likelihood w.r.t. w_{ij}, c_i and b_j respectively, becomes

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{v})}{\partial w_{ij}} = p(H_j = 1 \mid \boldsymbol{v}) v_i - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) p(H_j = 1 \mid \boldsymbol{v}) v_i, \tag{4.16}$$

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{v})}{\partial c_i} = v_i - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) v_i, \tag{4.17}$$

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{v})}{\partial b_j} = p(H_j = 1 \mid \boldsymbol{v}) - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) p(H_j = 1 \mid \boldsymbol{v}). \tag{4.18}$$

As noted earlier summing over all values of visible variables when computing the second term

[23] For the full derivation of the factorization see Fischer (2014).

of equation 4.16 comprises exponential complexity and thus the log-likelihood gradient is approximated by sampling from the model distribution with MCMC techniques. A prominent representative, which is commonly applied, is *Gibbs sampling*. A detailed description of this technique would exceed the scope of this work, thus please refer to Geman (1984)[23]. The basic idea, however, is to construct a Markov Chain by updating each variable with the respective conditional probability and the state of the others. Given the conditional independence between variables in the same layer allows to sample all variables in one layer jointly instead of sampling new values for all variables subsequently and thereby simplifies Gibbs sampling significantly. In fact, the procedure is reduced to only two steps: Sampling state $\boldsymbol{h}$ based on $p(\boldsymbol{h} \mid \boldsymbol{v})$ and sampling state $\boldsymbol{v}$ based on $p(\boldsymbol{v} \mid \boldsymbol{h})$. Unfortunately the sampling procedure has to be repeated until the Markov chain converges at the stationary point requiring a large number of steps, and further no reliable method exists to determine whether equilibrium has been actually reached. The resulting considerable computational effort can be reduced applying common RBM learning techniques as *Contrastive Divergence* (CD) or *Parallel Tempering*.

Contrastive Divergence

The idea of k-Contrastive Divergence, which was introduced by Hinton[27], is instead of running the Markov Chain implemented by Gibbs sampling to equilibrium one runs it only for k steps (and usually $k = 1$). Subseqently the parameters are updated to reduce the tendency of the chain to move away from the data distribution q^0. More formally, instead of minimizing $\mathrm{KL}(q^0 \parallel q^\infty)$[24], with q^∞ the distribution at equilibrium, we minimize $\mathrm{KL}(q^0 \parallel q^k)$, where q^k is the distribution over the k-step reconstructions of the data vectors computed by Gibbs sampling. The approximation of the log-likelihood gradient for one training vector is thus generally given by

$$\mathrm{CD}_k(\boldsymbol{\theta}, \boldsymbol{v}^{(0)}) = \sum_{\boldsymbol{h}} p(\boldsymbol{h} \mid \boldsymbol{v}^{(k)}) \frac{\partial E(\boldsymbol{v}^{(0)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} - \sum_{\boldsymbol{h}} p(\boldsymbol{h} \mid \boldsymbol{v}^{(0)}) \frac{\partial E(\boldsymbol{v}^{(k)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \tag{4.19}$$

Algorithm 2 shows the procedure in detail for training data set D making use of derivative equations 4.16, 4.17 and 4.18.

[24] Which is equivalent to maximize the log-likelihood as described earlier.

Algorithm 2 k-step contrastive divergence

Input: Training data D

Output: Gradient approximations Δw_{ij}, Δc_i and Δb_j

$\Delta w_{ij}, \Delta c_i, \Delta b_j \leftarrow 0$ for $i = 1, ..., n$, $j = 1, ..., m$ ▷ Initialization
for all $\boldsymbol{v} \in D$ **do**
  $\boldsymbol{v^{(0)}} \leftarrow \boldsymbol{v}$
  for $t = 1, ..., k$ **do** ▷ Gibbs sampling for k steps
    for $j = 1, ...m$ **do**
      sample h_j^t from $p(h_j \mid \boldsymbol{v^t})$
    for $i = 1, ...n$ **do**
      sample v_i^{t+1} from $p(v_i \mid \boldsymbol{h^t})$
  for $i = 1, ..., n$, $j = 1, ..., m$ **do**
    $\Delta w_{ij} \leftarrow \Delta w_{ij} + p(H_j = 1 \mid \boldsymbol{v^{(0)}})v_i^{(0)} - p(H_j = 1 \mid \boldsymbol{v^{(k)}})v_i^{(k)}$ ▷ Eq. 4.16
  for $i = 1, ..., n$ **do**
    $\Delta c_i \leftarrow \Delta c_i + v_i^{(0)} - v_i^{(k)}$ ▷ Eq. 4.17
  for $j = 1, ..., m$ **do**
    $\Delta b_j \leftarrow \Delta b_j + p(H_j \mid \boldsymbol{v}^{(0)}) - p(H_j \mid \boldsymbol{v}^{(k)})$ ▷ Eq. 4.18

In order to avoid looping over variables it is common practice to treat them as vector elements and consequently apply matrix multiplication, which further improves computation time.

4.3.2 Continuous-Valued Restricted Boltzmann Machine

Restricting the visible units to only assume binary values is severely limiting the capability of RBMs and their applications fields likewise. Especially when considering real-valued/continuous input data, as time series data, adjustments become inevitable and there is a variety of ways to deal with them.

Continuous Restricted Boltzmann Machine

Incipient work is constituted by Chen and Murray[14] introducing 'continuous stochastic units', where zero-mean Gaussian noise is added to the input of sampled sigmoid units.
Hinton[28] demonstrated, that a continuous-valued input can be modeled with binary units by firstly replacing the state space of visible units $\{0, 1\}$ by $[0, 1]$ and scaling the input data to the very same interval. Then the expectation $p(V_i = 1 \mid \boldsymbol{h})$ - the probability of the visible variable to be one - is regarded as the current state of the unit and substitutes the samples in the subsequent procedure.
Another quite common and simple approach is to use *truncated exponential* units. That is, allowing v to take any value in a given Interval I, here $I = [0, 1]$, the conditional density becomes

$$p(v_l \mid \boldsymbol{h}) = \frac{\mathrm{e}^{-v_l a_l(\boldsymbol{h})}\mathbf{1}_{v_l \in I}}{\int \mathrm{e}^{-v_l a_l(\boldsymbol{h})}\mathbf{1}_{v_l \in I}\mathrm{d}v_l} \tag{4.20}$$

with v_l denoting the lth visible unit and $a_l(\boldsymbol{h}) = -\sum_{j=1}^{m} w_{lj}h_j - c_l$. For $I = [0, 1]$ the normalizing integral yields $\frac{\mathrm{e}^{-a_l(\boldsymbol{h})}-1}{a_l(\boldsymbol{h})}$. The conditional expectation E of v_l given $\boldsymbol{h}$ is further sigmoid-like and

comprises monotone non-linearity:

$$E[v_l \mid \boldsymbol{h}] = \frac{1}{1 - \mathrm{e}^{-a_l(\boldsymbol{h})}} - \frac{1}{a_l(\boldsymbol{h})} \tag{4.21}$$

Samples from this truncated exponential are obtained from a uniform sample U, employing the inverse cumulative F^{-1} of the conditional density $p(v_l \mid \boldsymbol{h})$ s.t.

$$F^{-1}(U) = \frac{\ln(1 - U \times (1 - \mathrm{e}^{a_l(\boldsymbol{h})}))}{a_l(\boldsymbol{h})}. \tag{4.22}$$

An RBM with these type of units is commonly referred to as *Continuous Restricted Boltzmann Machine*.[7] It is important to note, that while the sampling step for visible units has been changed, the learning process remains the same as with binary hidden units.

Gaussian-Bernoulli Restricted Boltzmann Machine

The also widely appreciated model of *Gaussian-Bernoulli* or *Gaussian-Binary* RBMs (GBRBM) involves normally distributed visible variables and binary hidden units. That is, the linear pre-activation values of the visible units serve as means for drawing samples from a Gaussian distribution. The energy function is then augmented with quadratic terms and can be written as

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_j \frac{v_i}{\sigma_i^2} - \sum_{i=1}^{n} \frac{(v_i - c_i)^2}{2\sigma_i^2} - \sum_{j=1}^{m} b_j h_j \tag{4.23}$$

where σ_i denotes the standard deviation of the Gaussian distribution for visible unit i. In case the input data is normalized, one may set $\boldsymbol{\sigma} = 1$. Otherwise the model can be forced to learn the standard deviations itself, which are then effectively an additional vector of biases one must optimize. This supplementary task, however, can be challenging with Contrastive Divergence and requires a careful setting of learning parameters.
The corresponding conditional probability of a visible neuron is consequently given by

$$p(v_i \mid \boldsymbol{h}) = \mathcal{N}\left(v_i \,\middle|\, \sum_{j=1}^{m} h_j w_{ij} + c_i, \sigma_i^2\right), \tag{4.24}$$

where $\mathcal{N}(\cdot \mid \mu, \sigma^2)$ is the probability density of a Normal distribution with mean μ and variance σ^2. The conditional probability of a hidden neuron to be one is then

$$p(h_j \mid \boldsymbol{v}) = \mathrm{sig}\left(\sum_{i=1}^{m} w_{ij} \frac{v_i}{\sigma_i^2} + b_j\right) \tag{4.25}$$

and can be reduced to equation 4.12, when setting $\boldsymbol{\sigma} = 1$.[15][58]

4.3.3 Conditional Restricted Boltzmann Machine

Regardless of considering continuous values or not, an ordinary RBM (i.e. with a classic network topology) does not include any regional inter-dependencies of subsequent data points. That is,

the models presented so far model static frames of data, but cannot incorporate any temporal information. When dealing with time series data, however, short-term temporal dependencies and longer-term temporal structures should be taken into account[31]. In order to overcome this shortcoming Taylor et.al.[55] introduced the *conditional* RBM (cRBM) by adding two types of direct connections to previous data points (Figure 4.5). That is, a defined number n of past observations have an influence not only to the current hidden structure but also to current visible units. The weighting of these connections are also trained with contrastive divergence.

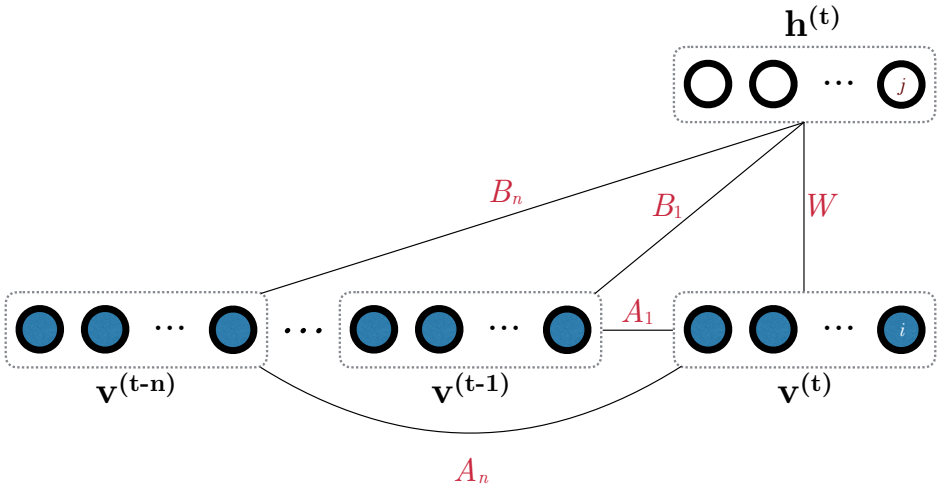


Figure 4.5: The network graph of a conditional Restricted Boltzmann Machine.

Practically this means treating data from previous time steps as a dynamically changing biases, which are defined as

$$\hat{c_i} = c_i + \sum_{i=1}^{k} A_i \boldsymbol{v}^{(t-i)} \tag{4.26}$$

and

$$\hat{b_j} = b_j + \sum_{i=1}^{l} B_i \boldsymbol{v}^{(t-i)} \tag{4.27}$$

where k denotes the number of auto-regressive connections between visible layers $\boldsymbol{v}$ at time $(t-i)$ and the current visible layer and l the number of connections to the current hidden layer likewise. k and l may differ, but for the sake of simplicity we set $k = l = p$.[31]
The conditional probabilities for a entirely binary cRBM are then given by

$$p(h_j \mid \boldsymbol{v}) = \text{sig}\left(b_j + \sum_{i=1}^{n} w_{ij} v_i + \sum_{o=1}^{p} \sum_{i=1}^{n} B_{ijo} v_i^{(t-o)}\right) \tag{4.28}$$

$$p(v_i \mid \boldsymbol{h}) = \text{sig}\left(c_i + \sum_{j=1}^{m} w_{ij} h_j + \sum_{o=1}^{p} \sum_{i=1}^{n} A_{ijo} v_i^{(t-o)}\right). \tag{4.29}$$

The parameters A and B have to be trained similarly to the classic RBM using Contrastive Divergence.[32]In practice it is more convenient to calculate the dynamic biases (Equ. 4.26, 4.27) seperately and insert them into the original equations of conditional probabilities instead of computing equations 4.28 and 4.29. This holds especially when applying non-binary units. Generally it must be stated, that although the marginal conceptual change might suggest minor increase in implementation complexity likewise, it still comes with some obstacles.

4.3.4 Parameter Setting

As already mentioned the parameter setting is doubtless non-trivial. Besides the values of numerical meta-parameters such as learning rate, momentum or weight-decay, one must also determine an adequate amount of hidden units or consider the initialization and the size of the data mini-batches during training phase. Furthermore a beneficial setting for one problem case might not be useful at all for another. Thus practical expertise to set and consequently relate failures in learning to the setting is a necessity. However, there has been extensive research in the last few years in this field, so one can benefit from shared knowledge as given by Geoffrey Hinton and collaborators [26].

Size of mini-batch

Although one can update the weights and biases based on the estimated gradient on a single observation, it is more efficient to split the training set into *mini-batches*, which are a group of observations. The update is then computed simultaneously for all observations in the batch. When applying gradient descent the size of the mini-batch can vary from 10 up to 100 training examples. In case *stochastic gradient ascent* is used, too large batches can be problematic.[26]

Learning rate

The learning rate $\eta \in \mathbb{R}^+$ represents the step range of the update and if chosen too large may let the reconstruction error increase significantly and weights may explode. However, choosing too small can result in slower learning.
It is common practice to implement a dynamic rate, which is reduced during the process and should correspond to the adjustment of the weights. Hinton (2010)[26] recommends a rate which is about 10^{-3} times the size of the weights.

Initial Weights and Biases

Usually the weights are initialized to small random values from a Gaussian distribution with $\mu = 0$ and standard deviation $\sigma = 0.01$. Larger values may speed up initial learning but can also deliver a worse model. In case binary visible units are used it is advantageous to initialize the bias of unit i to $\ln(p_i/(1-p_i))$ where p_i is the proportion of training vectors in which unit i is 1. Hidden biases are set to 0.[26]

Momentum

(Stochastic) gradient ascent - and descent likewise - experiences difficulties when navigating in ravines (i.e. areas where the gradient is a lot steeper in one dimension than in another), which are quite common around local minima[52]. As already mentioned the momentum $\nu\Delta\boldsymbol{\theta}^{(t-1)}_{\Delta\boldsymbol{\theta}^{(t)}}$ is controlled by $\nu \in \mathbb{R}^+_0$ and is a simple method to increase the speed of learning in case of existence of such ravines by accelerating (stochastic) gradient descent in the relevant direction and dampening oscillations. Metaphorically speaking one simulates a heavy ball rolling down a surface. The ball accumulates momentum as it rolls downhill and the momentum term enhances parameter updates for dimensions whose gradients point in the same directions. Conversely it reduces parameter updates for dimensions whose gradients change direction.

Hinton (2010)[26] recommends to start with a momentum of 0.5 and to raise it to 0.9 as soon as the main reduction of reconstruction error took place. This adjustment may cause a severe increase of error or a even lasting instability. The latter can be opposed by an adaptive learning rate.

Weight-decay

The term $\lambda\boldsymbol{\theta}^{(t)}$ in equation 4.8 is controlled by $\lambda \in \mathbb{R}^+_0$ and is the derivative of a function that penalizes large weights.[25] It reduces overfitting[26] and significantly improves the mixing rate of the alternating Gibbs Markov chain. According to Hinton (2010)[26], the penalty function 'L2' is the common and reasonable choice, and for a cost function C with energy function E it is defined as

$$C = E + \frac{\lambda}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}^2. \tag{4.30}$$

Applying the derivative of the weights yields:

$$\frac{\partial C}{\partial w_{ij}} = \frac{\partial E}{\partial w_{ij}} + \lambda w_{ij}. \tag{4.31}$$

The above equation intentionally just considers the weights between the units, while neglecting biases. Those in particular are typically less likely to cause overfitting and sometimes are required to be very large.
The value of the weight-cost coefficient λ usually ranges from 0.01 to 0.00001 depending on the amount of training data.[26]

Number of hidden units

Finally one must deal with the sensible number of hidden units, whose inappropriate setting can cause severe overfitting or tremendous computation time in training and testing. A common approach is to firstly estimate the number of bits that could describe each data vector $\boldsymbol{v}$, which is

[25] The simplified case where $\lambda = \nu = 0$ is commonly referred to as *Vanilla Gradient Descent*.
[26] In other words it prevents the network from using weights, which are not needed and severely improves generalization.

equivalent to $\int p(v) \log \frac{1}{p(v)} \, \mathrm{d}v$. Secondly this figure is multiplied by the number of training cases and eventually the designated amount of hidden units shall be about one order of magnitude smaller than that. However, if training cases show a high redundancy, which might be the case in big training sets of real-world scenarios, the amount should be even smaller.

4.4 Support Vector Machine

A *support vector machine* (SVM) is a supervised machine learning model and has been initially proposed by Vladimir Vapnik and collaborators in 1992.[6] The concept evolved, especially by introducing the *kernel trick* in order to create nonlinear classifiers. Developed in industrial environment, the research consistently focused on realistic application, maintaining it a prominent candidate for any real-world problem cases. This section provides an introduction to SVM, that is based on the tutorial of Burges (1998)[13] and the treatise of sparse kernel machines of Bishop (2006)[8] if not stated otherwise.

Formally speaking a SVM constructs a *hyperplane* or a set of hyperplanes in a high-dimensional space for the purpose of classification (which has been the primary motivation) or regression. The quality of separation is determined by the distance of the hyperplane to the nearest data point - the so-called *margin*, since the size of the margin coheres with the *generalization error*[27] of the method. Therefore the SVM is commonly referred to as *maximum margin classifier*.

The basic idea is to find a function $f(x)$ that, given training data $\{(x_1, y_1), ..., (x_l, y_l)\} \subset \mathcal{X} \times \mathbb{R}$, where $\mathcal{X}$ denotes the space of input patterns[28], as well as corresponding target values y_i, separates the data with largest possible margin. Misclassification permitted to a predefined extent, is known as *soft-margined*, while the opposite is called *hard-margined*.

In case f is supposed to be linear, it is defined as

$$f(x) = \langle w, x \rangle + b \text{ with } w \in \mathcal{X}, b \in \mathbb{R}, \tag{4.32}$$

where w represents the (not necessarily normalized) normal vector to the hyperplane and $\frac{b}{\|w\|}$ the offset of the origin along w.
As illustrated for a hard-margined linear separation of two classes in figure 4.6, maximizing the distance between the planes, which are parallel to the hyperplane and constructed through the nearest data point(s) of each class, is equivalent to minimizing $\|w\|^2$. Additionally one has to prevent data points from falling into the margin. The convex optimization problem can then be

[27] The generalization error is a common measure for the accuracy of the prediction made by supervised machine learning models.
[28] $\mathcal{X}$ needs to be a *Hilbert space*, so that there exists a dot product $\langle \cdot, \cdot \rangle$.

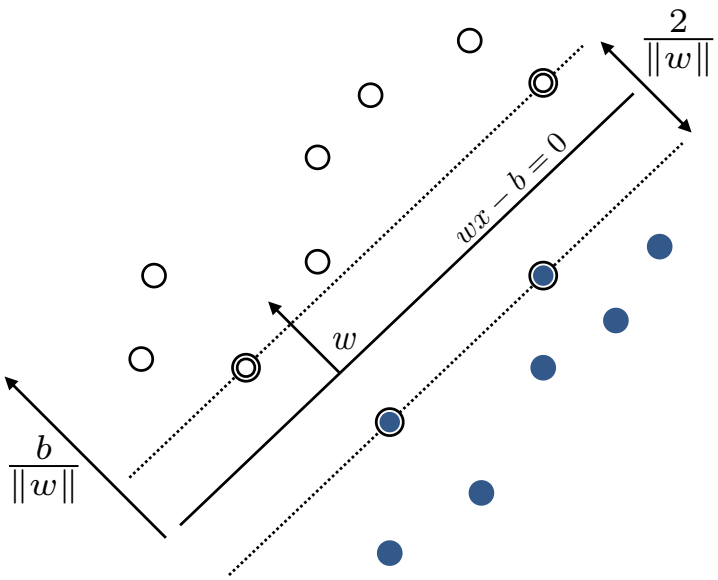


Figure 4.6: The linearly separating hyperplane for the separable case.

written as:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_i(\langle w, x\rangle + b) \geq 1 \ \ \forall i \end{aligned} \tag{4.33}$$

In order to solve this constrained optimization problem one makes use of the standard dualization method utilizing *Lagrange multipliers* and the *Karush-Kuhn-Tucker* (KKT) *conditions*. At first the *Lagrange function* Λ_P is build from the objective function and the constraint (as defined in equation 4.33), whereas for the latter a dual variable, the Lagrange multiplier λ_i, is introduced:

$$\Lambda_P = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \lambda_i [y_i(\langle w, x\rangle + b) - 1] \tag{4.34}$$

Since it can be shown that this function has a saddle point at the solution, consequently the partial derivatives of Λ_P with respect to the *primal* variables (b, w) have to vanish for optimality:

$$\frac{\partial \Lambda_P}{\partial w} = w - \sum_{i=1}^{l} \lambda_i y_i x_i = 0 \tag{4.35}$$

$$\frac{\partial \Lambda_P}{\partial b} = \sum_{i=1}^{l} \lambda_i y_i = 0 \tag{4.36}$$

$$\tag{4.37}$$

The derived conditions are equality constraints in the dual formulation, thus can be substituted

for w and b in equation 4.34, which then yields the *dual representation*

$$\begin{aligned} \text{maximize} \quad & \Lambda_D = \sum_{i=1}^{l} \lambda_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \lambda_i \lambda_j y_i y_j \langle x_i, x_j \rangle \\ \text{subject to} \quad & \begin{cases} \lambda_i \geq 0 \ \ \forall i \\ \sum_{i=1}^{l} \lambda_i y_i = 0. \end{cases} \end{aligned} \tag{4.38}$$

The dual problem is equivalent to the primal one, though w is never computed explicitly, but represented as linear combination of training examples x_i. The Support Vector method is named after a subset of the training data, for which elements $\lambda_i \neq 0$. Intriguingly the complexity of the function does not depend on $\mathcal{X}$, but on the amount of support vectors.

One way to compute the bias b is to make use of the KKT conditions, that inter alia state that the product between dual variables and constraints has to vanish. That is

$$\lambda_i [y_i f(x_i) - 1] = 0, \tag{4.39}$$

where $f(x_i)$ (see equation 4.32) can be expressed substituting w using equation 4.35 s.t.

$$f(x) = \sum_{i=1}^{l} \lambda_i y_i \langle x, x_n \rangle + b. \tag{4.40}$$

Now b can be easily computed by using an arbitrarily chosen support vector x_i, though it is numerically more stable to take the mean value resulting from equations of all support vectors.

There will be definitely situations where a convex optimization will be not feasible, as linear separation in classification tasks may sometimes not be possible either. In these cases one usually applies the *soft margin* loss function, introducing positive *slack variables* ξ_i to adjust the constraints of the optimization problem, which then is defined as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & \begin{cases} \langle w, x_i \rangle + b \geq 1 - \xi_i & \text{for } y_i = +1 \\ \langle w, x_i \rangle + b \leq -1 + \xi_i & \text{for } y_i = -1 \\ \xi_i \geq 0 \ \ \forall i. \end{cases} \end{aligned} \tag{4.41}$$

C controls the trade-off between the slack variable penalty and the margin. The linear soft-margin approach (equations 4.41) is illustrated in figure 4.7.
The corresponding Lagrangian to equation 4.41 is then given by

$$\Lambda_P = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \lambda_i [y_i (\langle w, x \rangle + b) - 1 + \xi_i] - \sum_{i=1}^{l} \eta_i \xi_i, \tag{4.42}$$

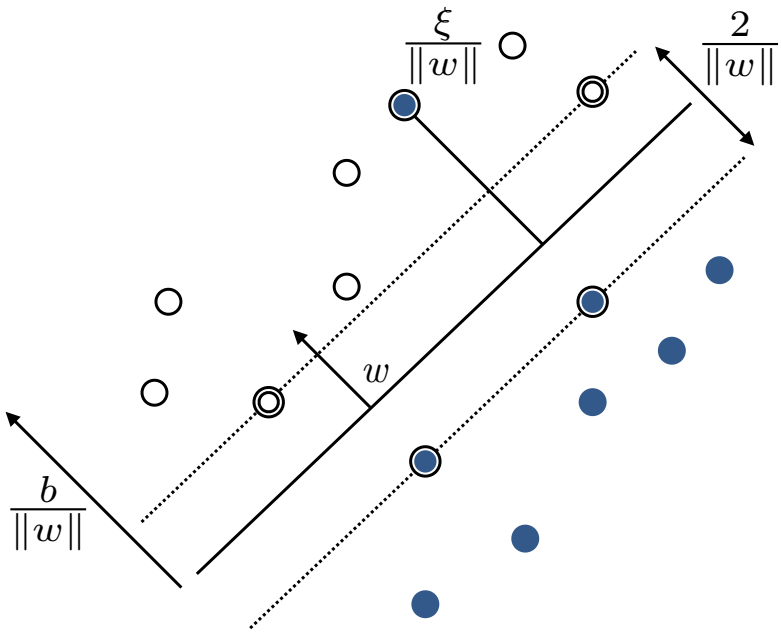


Figure 4.7: The linearly separating hyperplane for the non-separable case.

where η_i and λ_i are Lagrange multipliers. In order to obtain the dual optimization problem, the derivatives with respect to the primal variables are taken and substituted into the primal Lagrange objective function, which consequently yields[29]

$$\begin{aligned} \text{maximize} \quad & \Lambda_D = \sum_{i=1}^{l} \lambda_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \lambda_i \lambda_j y_i y_j \langle x_i, x_j \rangle \\ \text{subject to} \quad & \begin{cases} 0 \leq \lambda_i \leq C \ \ \forall i \\ \sum_{i=1}^{l} \lambda_i y_i = 0. \end{cases} \end{aligned} \tag{4.43}$$

Though the objective function, we seek to maximize, is identical to the soft-margined case, the constraints alternate.

However, so far we only covered a linear decision function, which severely limits the applicability for real-life scenarios.

4.4.1 Nonlinear Support Vector machines

The consecutive action is to make the SV algorithm nonlinear, which is usually achieved by applying a function to the training data x_i, $\Phi : \mathcal{X} \rightarrow \mathcal{F}$ prior to the actual training. The function maps the input into a higher-dimensional feature space $\mathcal{F}$, followed by the standard SV algorithm - combined allowing nonlinear separation.
However explicitly computing Φ can turn out to be computational expensive. Nevertheless it has been shown that it does suffice to simply know the dot product $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$, the

[29] For the complete derivations see [13] and [8].

so-called *kernel*, instead and substitute it into the optimization problem.[30] As a consequence it has been studied, which functions $k(x, x')$ correspond to a dot product in a feature space $\mathcal{F}$ (e.g.[47]). In fact one distinguishes different types such as *polynomial* or *Gaussian* or *radial basis function* (RBF) kernels. The definition of a polynomial kernel of degree d and a Gaussian kernel with width controlling parameter γ is given by equations 4.44, 4.45 respectively.

$$k(x, x') = (\langle x, x' \rangle + 1)^d \tag{4.44}$$

$$k(x, x') = \exp(-\gamma \|x - x'\|^2) \tag{4.45}$$

Due to the property, that the SVM depends on the data only through dot products, it formally belongs to the general category of *kernel methods*.[6]

4.4.2 Parameter Setting

The training of a SVM sets the parameters λ_i and b in order to find the large margin hyperplane. Nevertheless, the pre-setting of the so-called *hyperparameters*, such as the soft margin constant C or kernel depending parameter γ, has crucial effect on the decision boundary of the algorithm. A large value for parameter C results in a high penalty to margin errors, thus affecting the size of the margin of a SVM. The kernel parameters in turn control the flexibility of the classifier. If setting to high, however, it can result in *overfitting* (see following section). Finally the adjustment of one parameter may affect the other. For a Gaussian kernel the decrease of γ decreases the curvature of the decision boundary. An increase of C forces the curve to adjust. The standard method of visually exploring the accuracy of the classifier for different parameters is a *grid-search*. In this method one assigns for each point of a grid, which is corresponding to specific parameter values on the axes, an accuracy value.[6] Another popular approach for the optimization of parameter choice is the usage of *simulated annealing* algorithms (e.g. [41], [33]).

4.5 Overfitting

One fundamental concept of machine learning is *generalization*. This is the intuitively reasonable property of a model to be applicable also to data to which it has not been applied yet.
Overfitting in turn is the related effect, which may occur when a machine learning model starts to describe the noise present in the given data instead of the underlying relationship, i.e. the patterns. The reason may be, that the model is excessively complex with a large amount of parameters relative to the amount of training data. Consequently the model lacks robustness and overreacts to fluctuations in the data. Simply put: if we allow our machine learning model enough flexibility to find patterns, it certainly will discover some. However, these may just be chance occurrences in the data, which are useless given the objective of generalization. To counteract there have been proposed techniques such as adding a *sparsity term* to the objective function or the regularization method called *dropout*. Even though all procedures have the

[30] E.g. $\Phi : \mathbb{R}^2 \to \mathbb{R}^3$ with $\Phi(x_1, x_2) = (x_1^2, \sqrt{2x_1x_2}, x_2^2)$. Boser et. al. observed in this case that $k(x, x') = \langle (x_1^2, \sqrt{2x_1x_2}, x_2^2), ({x'_1}^2, \sqrt{2x'_1x'_2}, {x'_2}^2) \rangle = \langle x, x' \rangle^2$.[9]

tendency to overfit, it still can be a strong indication for an inappropriate model.[31][44]

4.6 Implementation

The implementation of the depicted algorithms was conducted in the programming language `Python`. The time series data (see section 3) required continuous hidden units, so that we implemented a **standard** as well as a **conditional Gaussian-Bernoulli restricted Boltzmann machine**. The re-implementation of the **support vector machine** has been omitted, since the used programming language provides well established librariers.The predictions have been derived based on a time horizon t of seven days, which corresponds to a number of 1008 time series records ($6\frac{records}{h} * 24\frac{h}{day} * 7days$).

4.6.1 Gaussian-Bernoulli restricted Boltzmann machine

In order to achieve optimal performance we initially scaled each feature x_j with the attribute's overall maximum and minimum:

$$x'_j = \frac{x_j - min(\boldsymbol{x}_j)}{max(\boldsymbol{x}_j) - min(\boldsymbol{x}_j)} \tag{4.46}$$

The parameter values and procedures for the contrastive divergence learning, which have been implemented, are based on the ideas of Hinton (2010)[26].

Dynamic parameters

The number of *Gibbs sampling* iterations is progressively increased over the number of learning *epochs* E, starting with one and ending with eight sampling iterations. The underlying idea is to enhance the accuracy at a later state of training process. Likewise the *learning momentum* and the *learning rate* are dynamically implemented, whereas the former starts with 0.5 and reaches 0.9 in the last epochs and the latter at first starts with 0.0010 and successively is halved as soon as the moment increases. All three parameters are illustrated in figure 4.8. However, the learning rate is additionally coupled to the *update target*, i.e. the desired adjustment of weights each epoch.
That is, the learning rate is adjusted depending on the update size of the weights and biases (see pseudo-code 3), until a predefined *decay period* p, $0 \leq p \leq 1$, at the end of training. After this point the learning rate is continuously reduced with a auto-regressive decay parameter α given a *decay target* δ and the total number of epochs ϵ (see equation 4.47).

$$\alpha = \delta^{1/pE}. \tag{4.47}$$

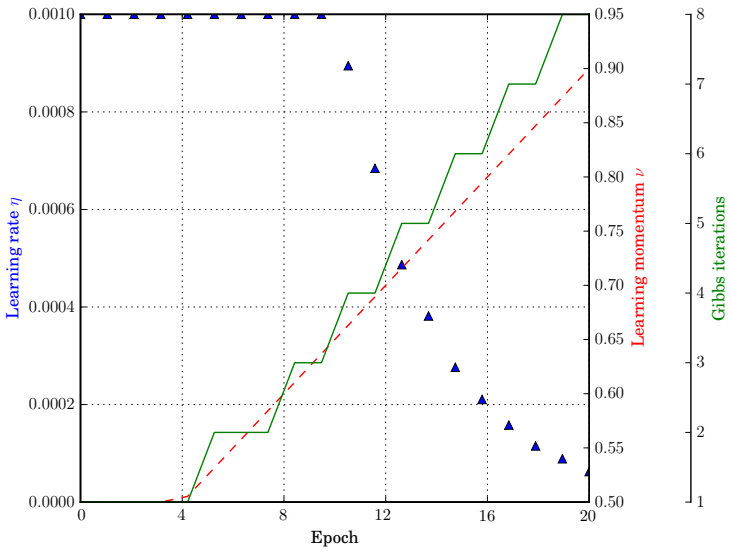


Figure 4.8: The RBM learning parameters.

Algorithm 3 Dynamic learning rate adjustment.

Input: weight updates in epoch e, update target τ, number of epochs E

if $e < \text{int}(E(1-p))$ **then** ▷ Check whether decay start is reached
  if $\max(\text{update}) < 0.1\tau$ **then**
    $\eta_{e+1} = 2\eta_e$
  else if $\max(\text{update}) > 10\tau$ **then**
    $\eta_{e+1} = 0.5\eta_e$
  else if $\max(\text{update}) < 0.9\tau$ **then**
    $\eta_{e+1} = 1.1\eta_e$
  else if $\max(\text{update}) > 1.2\tau$ **then**
    $\eta_{e+1} = 0.9\eta_e$
  else
    $\eta_{e+1} = \eta_e$
else
  $\eta_{e+1} = \alpha\eta_e$ ▷ Decay of learning rate

Static parameter values

The remaining parameter values, which have been set using the guideline of Hinton (2010)[26] (such as decay period or number of epochs), are given in table 4.1. [31]
Finally we have to determine a threshold τ_{th} for prediction purposes The underlying idea is to examine whether a new example, i.e. an actual observation, is probable in regard to the distribution, which has been learned by the RBM.

Prediction

As already stated, the model can be used to estimate the likelihood of occurrence of a given observation after it has learned the underlying distribution. For this purpose one may examine

[31] It has to be stated that other values despite the recommendations of Hinton have been used, yielding similar results.

Parameter	Value
Weight decay λ	0.0001
Epochs E	20
Batch size	100
Update target	0.1%
Decay target δ	10%
Decay period p	0.1
Reshuffling	every 10 examples

Table 4.1: RBM static parameter setting.

the reconstruction error computed with the model for one example. Concretely one firstly computes the binary hidden units based on the input (illustration 4.9 (a)) and consequently the values of the visible units based on the hidden units' values (illustration 4.9 (b)). Now the deviations between *initial* and *recalculated* values represent the reconstruction error. The smaller the error the more likely is the observation's occurrence.[32]. However, variation/error is accepted to a certain extent, which is in turn implemented with the before mentioned threshold τ_{th}. The threshold is actually a vector, which holds a *probable* deviation for each attribute. It has been obtained by computing the average reconstruction error for each feature of a single batch of observations with **10 Gibbs sampling iterations**. Additionally we allow deviation values during testing to be 20% larger as the initially computed value. This measure shall take the non-stationarity of the time series into account. That is, even when representing a valid approximation to the deviation in the training data set, the threshold might not be equally representative in the testing data set.

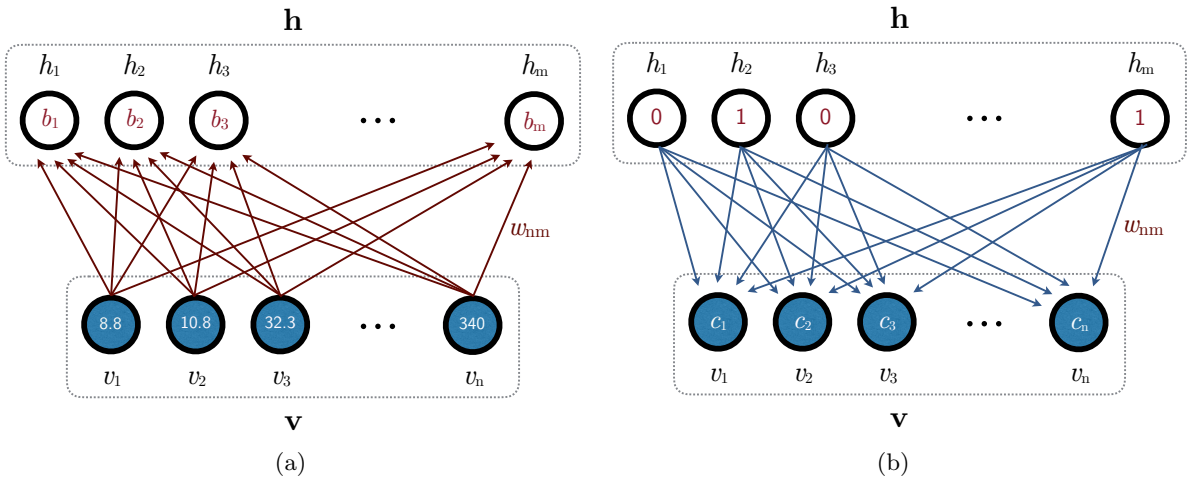


Figure 4.9: Recalculation of the input values with prior learned weights.

Equation 4.48 shows the prediction of an event (x_t) at time t in case the reconstruction error is larger than the *threshold* of at least one of a features (attributes) . The reconstruction error is defined as the absolute difference between input at the visible units and values which are

[32]This is essentially the procedure in any Gibbs sampling iteration. Although trivial, it has to be stated that a higher amount of Gibbs iterations yields a more likely output and therefore a more likely reconstruction error.

reconstructed by the RBM using weight and biases.

$$x_t = \begin{cases} 1 \text{ if } \exists i \in \{1,..,a\} : (v_{i,true} - v_{i,recon}) > \tau_{th} \\ 0 \text{ otherwise} \end{cases} \tag{4.48}$$

4.6.2 Conditional Gaussian-Bernoulli restricted Boltzmann machine

The parameter setting for the conditional Gaussian-Bernoulli restricted Boltzmann machine has been slightly adjusted. Since initial test results of the the training with 20 epochs showed much more fluctuation of the reconstruction error and no signs of convergence (see figure 4.10), firstly the number of epochs has been significantly enhanced. Secondly, to stabilize the learning process[33], the momentum has been set to 0 and the learning rate decoupled from the weight adjustment. In fact the learning rate has been set to various values, static as well as decreasing over time. Additionally we have to define the range of regional inter-dependency, i.e. the amount of past data points that influence the current hidden and current visible units (see section 4.3.3). The used values are given in table 4.2.

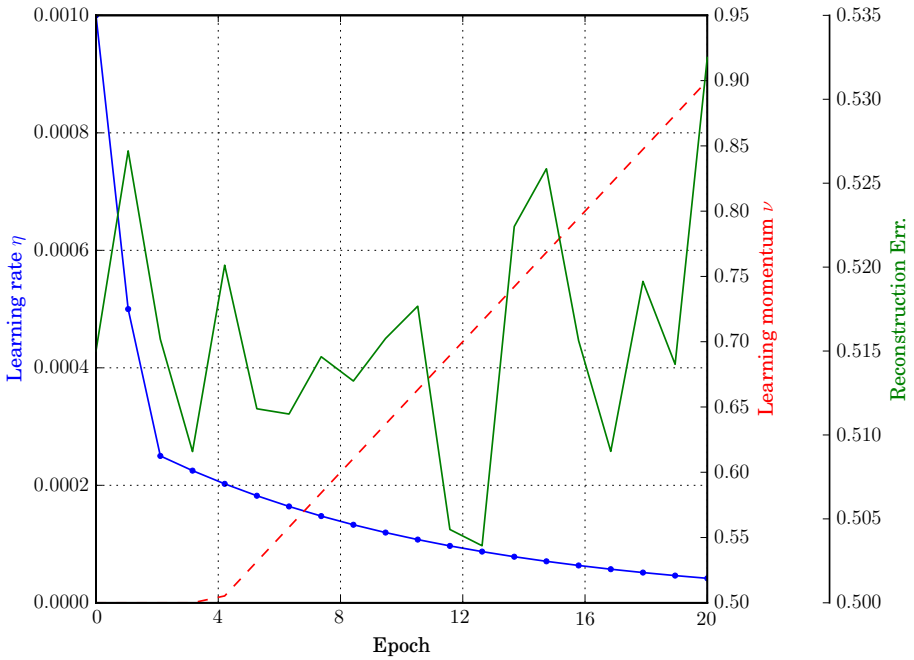


Figure 4.10: Initial training process of the conditional GBRBM.

4.6.3 Support vector machine

Since support vector machines assume that the input data is in a standard range, normalization of feature vectors prior to training is crucial. We applied 'hard' normalization, which signifies mapping of the minimum and maximum values of a given attribute to 0 and 1.

$$x'_j = \frac{x_j - min(\boldsymbol{x}_j)}{max(\boldsymbol{x}_j) - min(\boldsymbol{x}_j)} \tag{4.49}$$

[33] Surprisingly the merely prolongation of the learning process had no effect.

Parameter	Value
Weight decay λ	0.0001
Learning rate η	0.001 | 0.01 | 0.1 | decreasing
Momentum ν	0.0
Epochs E	300 | 1000
Batch size	100
Update target	0.1%
Decay target δ	10%
Decay period p	0.1
Reshuffling	every 300 examples
# conditional dependencies	1 - 5 n_c

Table 4.2: cRBM parameter setting.

The different configurations of the **support vector machine** are given in tables 4.3, 4.4. The degree of a the polynomial kernel and the error penalty parameter C have been varied to find optimal results.

Parameter	Value
Penalty parameter C	0.1-1.5
Kernel type	*radial basis functions*
γ value	$\frac{1}{\text{number of features}}$
Maximum iterations	unlimited

Table 4.3: SVM parameter setting for radial basis function kernel.

Parameter	Value
Penalty parameter C	0.1-1.5
Kernel type	*polynomial*
Degree	5 - 15
γ value	$\frac{1}{\text{number of features}}$
Maximum iterations	unlimited

Table 4.4: SVM parameter setting for polynomial kernel.

5 Results

This section describes and illustrates the findings of the case study and discusses the results. For this purpose firstly the training process of the restricted Boltzmann machine is examined and subsequently the prediction capability of both methods is investigated. The data has been chronologically split into *training* and *test data set* at the ratio of 70:30.

5.1 Training process of (c)GBRBM

The training process of the RBM should follow a successively improvement of the likelihood given the training data by adjusting the weights and biases. Hence the *reconstruction error* can be[34] used as an indicator for the refinement over the epochs. Figures 5.1, 5.2, 5.3 show this in detail for the GBRBM for three different turbines. The alteration of learning rate (blue) is depending on the extent of weight adjustment (see pseudo-code 3). The standard implementations show improvement as expected. Figure 5.1 furthermore depicts the dynamic learning rate: At an early epoch the rate is heavily increased, due to low weight updates, until a certain point where the increasing updates become to large. A decrease of the rate is the logical consequence until the decay period is reached.

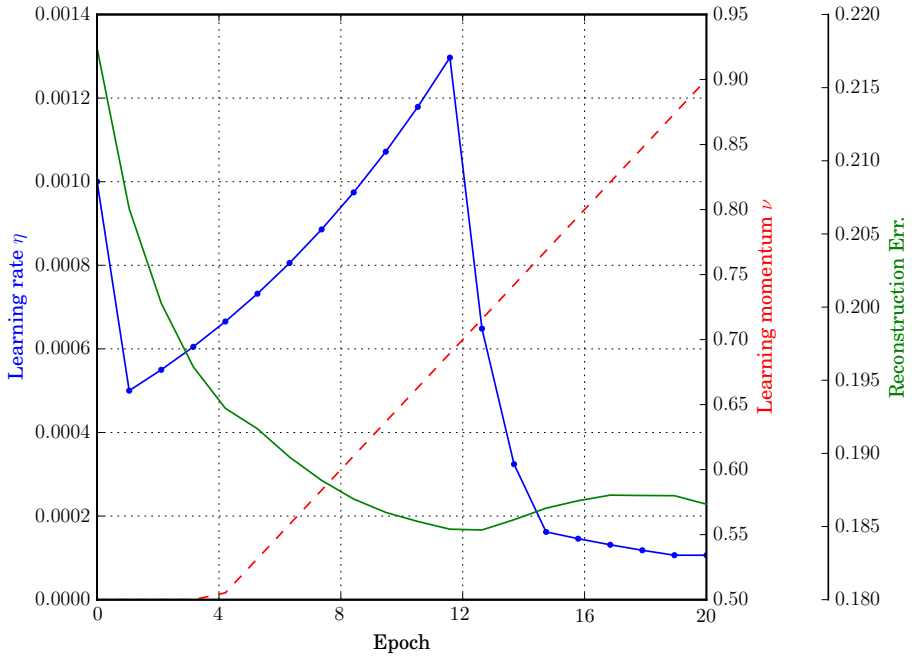


Figure 5.1: Training process of the GBRBM given data of turbine one.

[34] Though this is in dispute (see Hinton (2010)[26]).

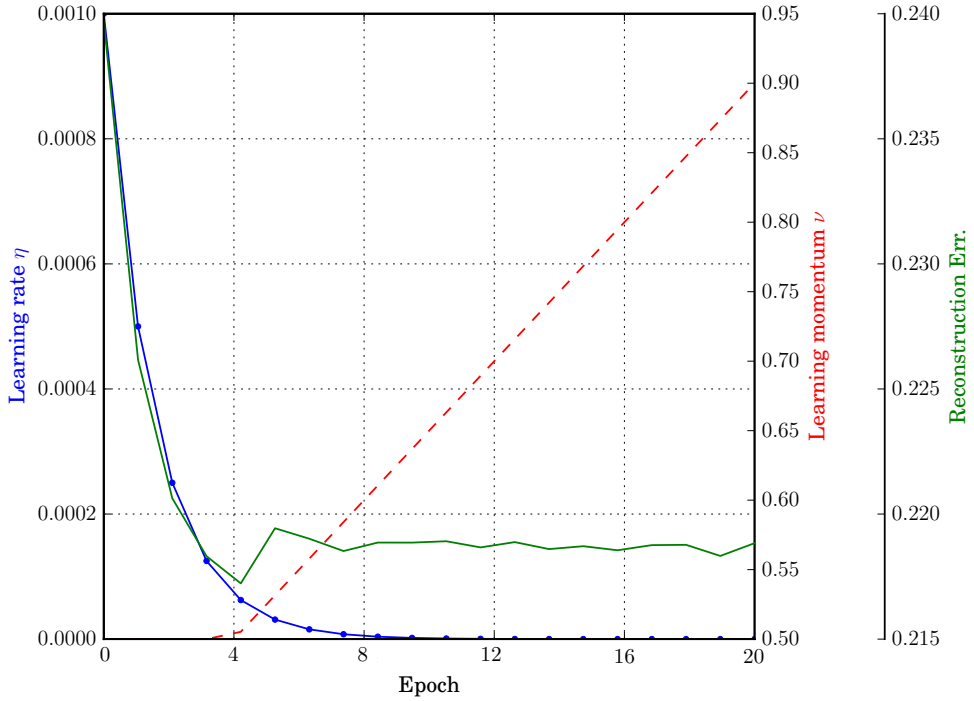


Figure 5.2: Training process of the GBRBM given data of turbine two.

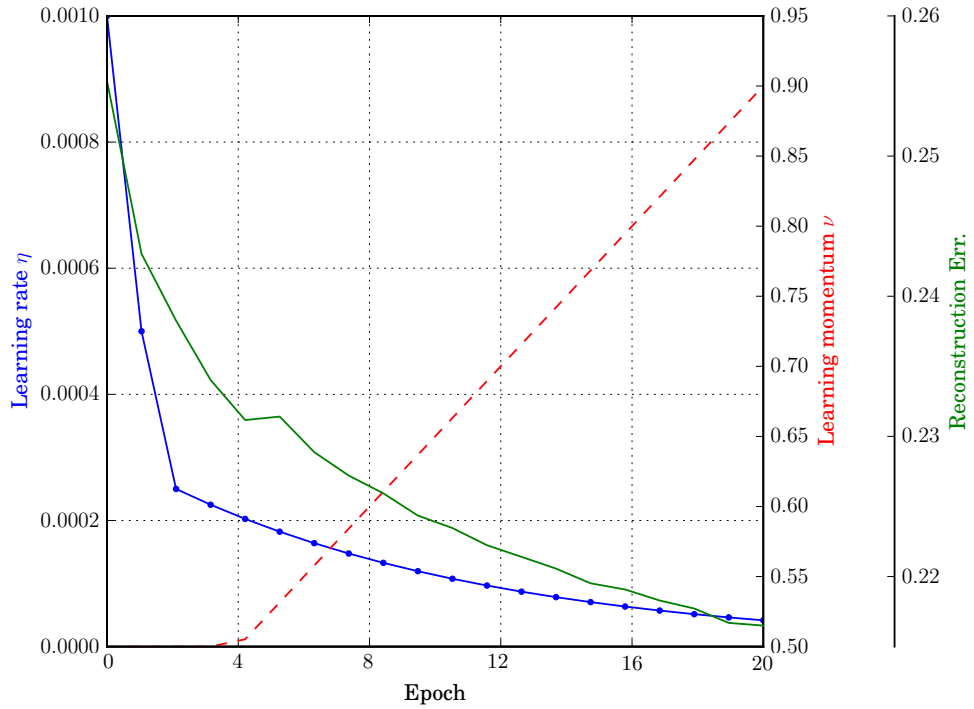
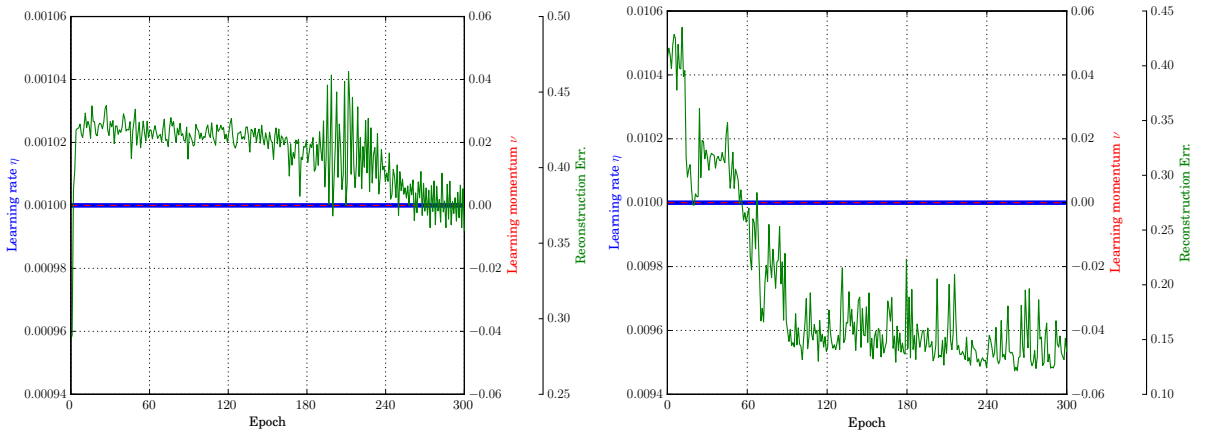


Figure 5.3: Training process of the GBRBM given data of turbine three.

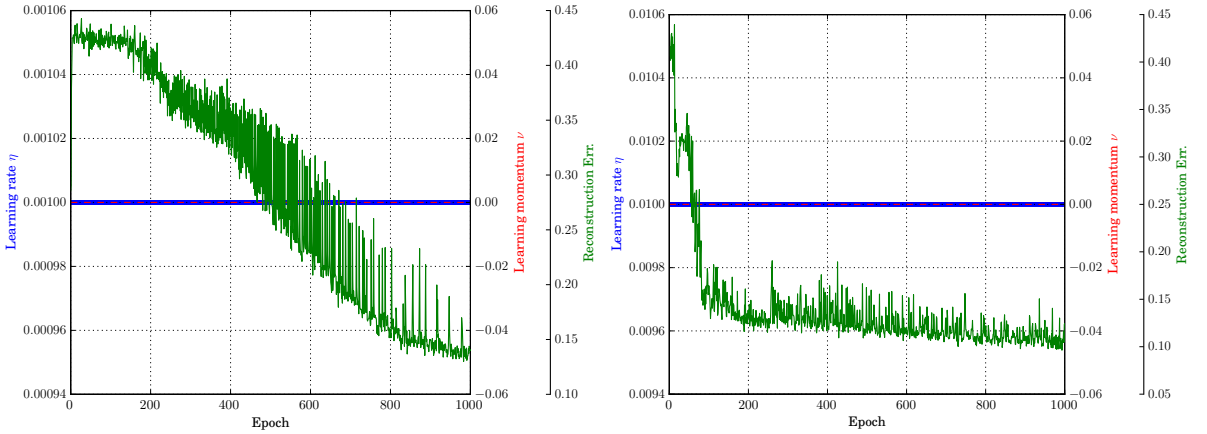
Intriguingly the **conditional** restricted Boltzmann machine shows striking fluctuations of the reconstruction error, depending on the parameter setting and on the choice of regional inter-dependency, that is the number of past data points influencing the current units. Based on the recommendations of Hinton (2010) [26] we started the training experiments considering one past data point, i.e. $n_c = 1$, and using $\eta = 0.001$, 0.01 respectively, $\nu = 0$ and 300 epochs. However, as one can see in figure 5.4 the reduction of the reconstruction error with $\eta = 0.001$ (a) has been slower compared to 0.01 (b).



(a) (b)

Figure 5.4: Training process of the cGBRBM with $\eta = \{0.001, 0.01\}$, $n_c = 1$ and 300 epochs.

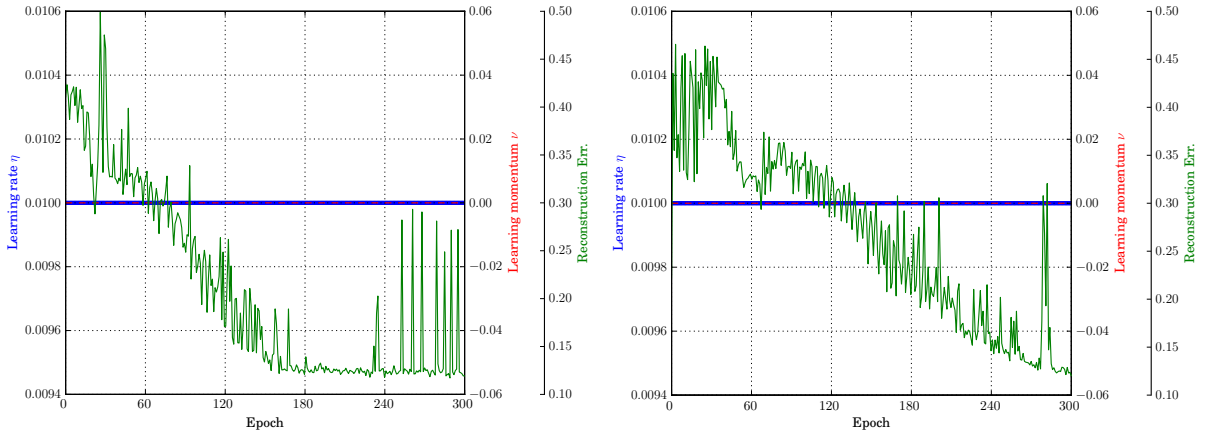
Prolonging the training procedure by setting $E = 1000$ shows that the lower learning rate results in slower convergence to the minimum, which nevertheless is 5% higher (0.15 normalized reconstruction error) in comparison to the training with $\eta = 0.01$ (0.10 normalized reconstruction error). Furthermore the reduction exhibits a stronger fluctuation (see figure 5.5).



(a) (b)

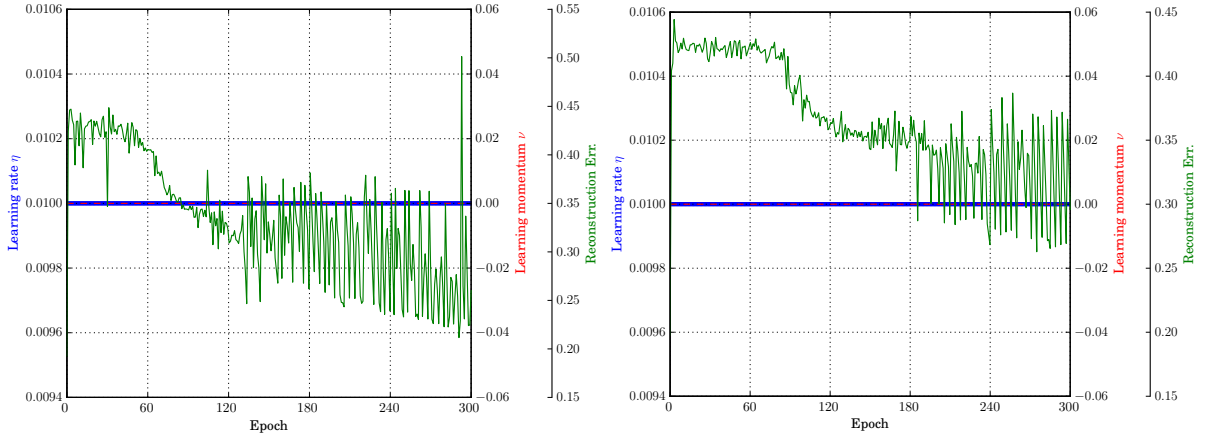
Figure 5.5: Training process of the cGBRBM with $\eta = \{0.001, 0.01\}$, $n_c = 1$ and 1000 epochs.

Since the model performed appropriately with $n_c = 1$, we consequently investigated the training process with > 1 conditional units. Figures 5.6 and 5.7 illustrate the process for 2 and 3, 4 and 5 units respectively.



(a) (b)

Figure 5.6: Training process of the cGBRBM with $\eta = 0.01$, $n_c = \{2, 3\}$ and 300 epochs.



(a) (b)

Figure 5.7: Training process of the cGBRBM with $\eta = 0.01$, $n_c = \{4, 5\}$ and 300 epochs.

One can easily see that convergence speed is reduced with increasing number of considered past units. In other words with growing model complexity the learning of the underlying distribution of the data requires a higher number of learning epochs in order to reach a minimum. Figure 5.8 depicts the process for a cGBRBM with $\eta = 0.01$, $n_c = 3$, $E = 1000$. As already noticeable with $E = 300$ and different values of n_c the error tends to oscillate at convergence. There could be a number of reasons causing this oscillation. A very likely one, however, might be that the model is not able to adjust properly the connecting weights to units, that are representing former time series $(v_{t-1}, v_{t-2}, ..)$. Possibly the influence on the current state in reality (at least in the provided data)[35] cannot be represented with the applied model. However, the conditional units improve the modeling through reducing the reconstruction error by obtaining further features in the form of regional inter-dependencies of the data points.

[35]Recall that the provided time series data consists of already averaged data (10 minutes average values).

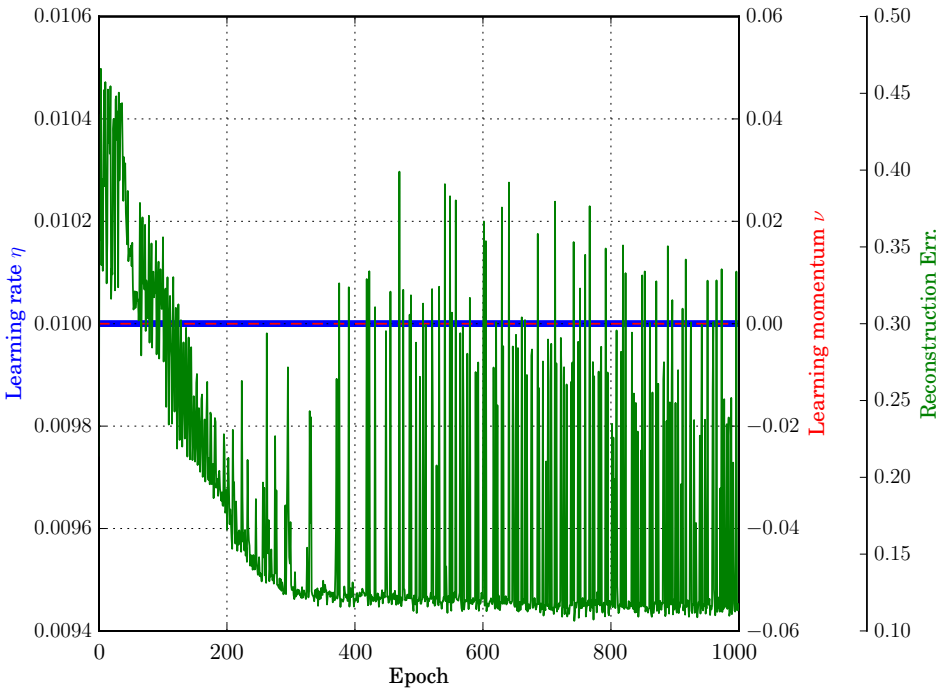


Figure 5.8: Training process of the cGBRBM with $\eta = 0.01$, $n_c = 3$ and 1000 epochs.

5.2 Prediction capability

As already outlined the implemented models shall detect events based on time series data. The scenario is fairly simplified, since the models serve as merely binary classifier, that decide whether something happens given the input or not. That is, one can enlarge the setting of the investigation by e.g. classifying with regard to different types of events or with regard to positive and negative events. However, we set the focus on examining the general prediction capability of the RBM.

The capability to correctly predict future events is measured with the so-called *ROC*, a visualization which besides correct also captures incorrect detections.

5.2.1 ROC

A *receiver operating characteristic* (ROC) is a graphical plot used for visualizing the performance of classifiers. It depicts the tradeoff between the rate of positives correctly classified (*true positives*) and the rate of negatives incorrectly classified ((*false positives*)). For better understanding of the terms see the confusion matrix in figure 5.9.

$$\text{true positives } rate = \frac{\text{positives correctly classified}}{\text{total positives}} \tag{5.1}$$

$$\text{false positives } rate = \frac{\text{negatives incorrectly classified}}{\text{total negatives}} \tag{5.2}$$

$$\text{sensitivity} = \text{recall} \tag{5.3}$$

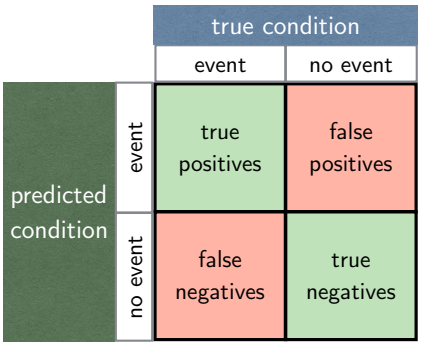


Figure 5.9: The confusion matrix.

$$\text{specificity} = \frac{\text{true negatives}}{\text{false positives} + \text{true negatives}} = 1 - \text{FP } \textit{rate} \tag{5.4}$$

The ROC curve demonstrates the tradeoff between *sensitivity* and *specificity* and the area under the curve (AUC) measures the *accuracy*. The latter is useful, when a single number for performance is needed. A diagonal line from down left to up right represents a classification performance of a random guess.[17][44] The ROC plots for SVM and RBM for different scenarios are illustrated in the following figures.

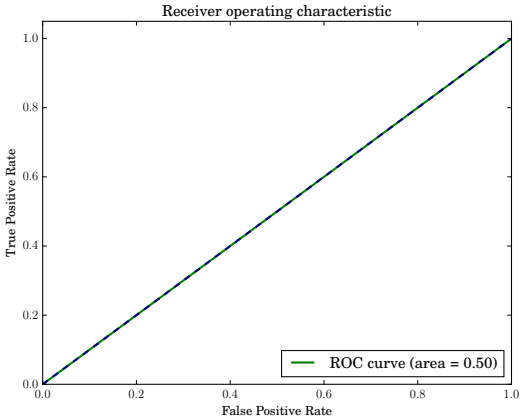


Figure 5.10: The ROC for RBM and turbine one (t=7).

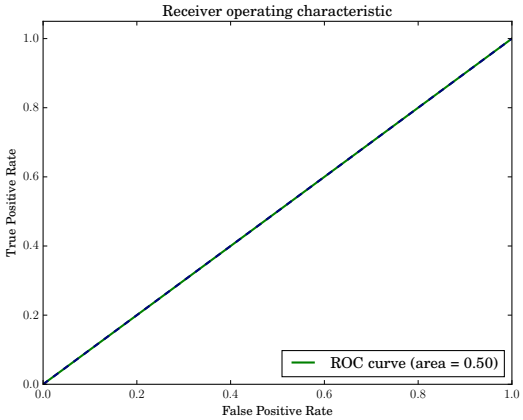


Figure 5.11: The ROC for RBM and turbine two (t=7).

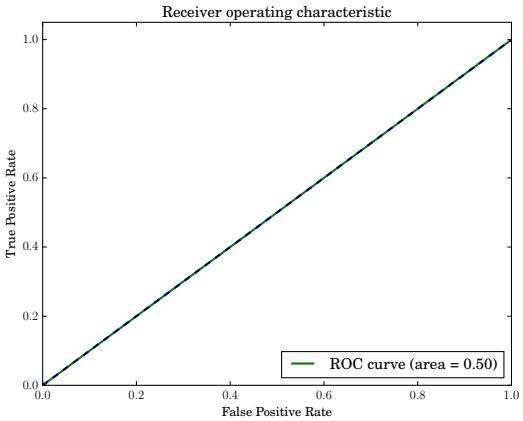


Figure 5.12: The ROC for RBM and turbine three (t=7).

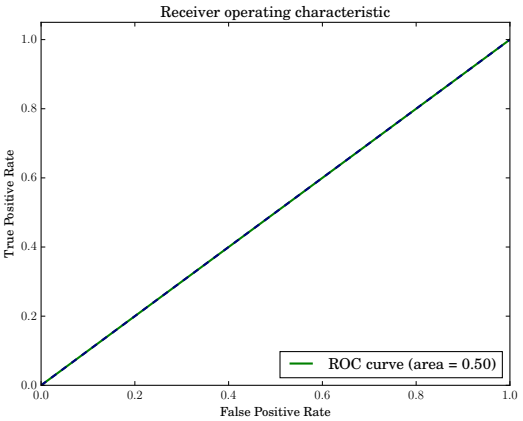


Figure 5.13: The ROC for CRBM and turbine one (t=7, $n_c = 1$).

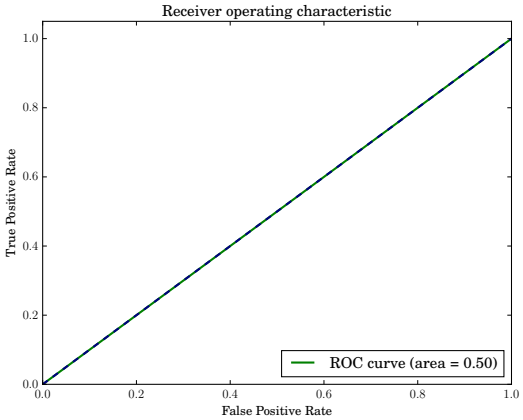


Figure 5.14: The ROC for CRBM and turbine two (t=7, $n_c = 1$).

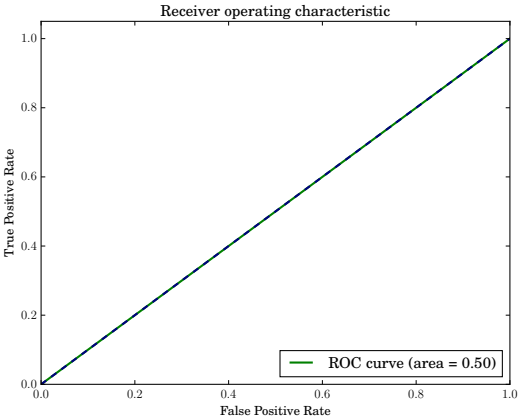


Figure 5.15: The ROC for CRBM and turbine three (t=7, $n_c = 1$).

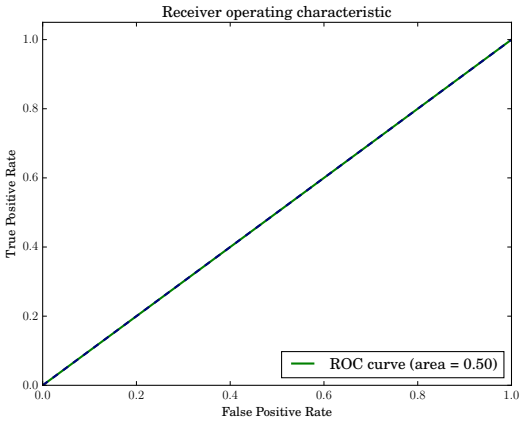


Figure 5.16: The ROC for SVM and turbine one (t=7, *rbf*, C=0.1).

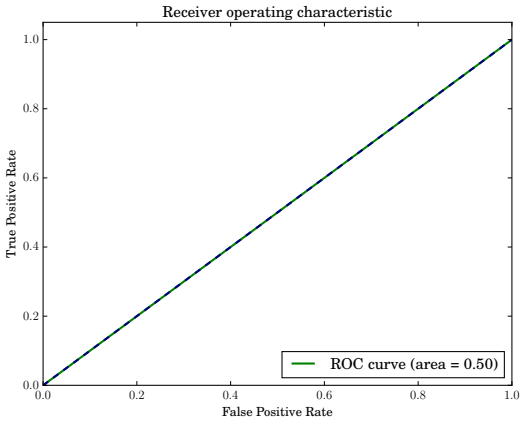


Figure 5.17: The ROC for SVM and turbine two (t=7, *rbf*, C=0.1).

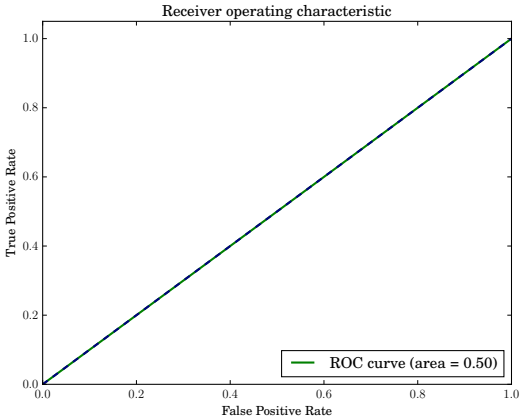


Figure 5.18: The ROC for SVM and turbine three (t=7, *rbf*, C=0.1).

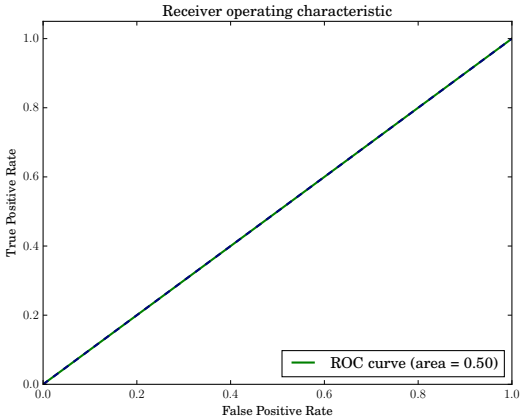


Figure 5.19: The ROC for SVM and turbine one (t=7, *rbf*, C=1.5).

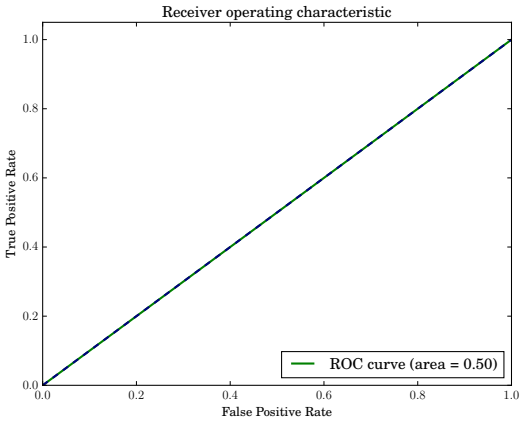


Figure 5.20: The ROC for SVM and turbine two (t=7, *rbf*, C=1.5).

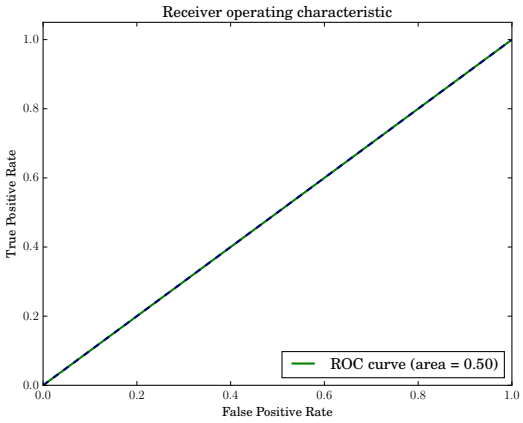


Figure 5.21: The ROC for SVM and turbine three (t=7, *rbf*, C=1.5).

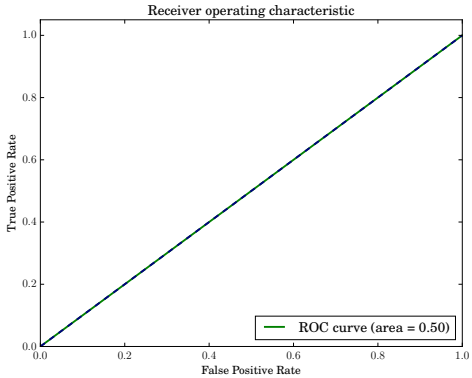


Figure 5.22: The ROC for SVM and turbine one (t=7, *poly*, C=0.1).

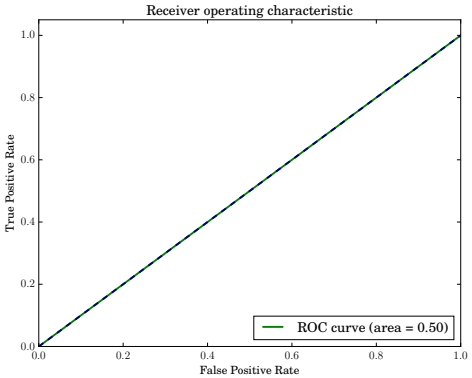


Figure 5.23: The ROC for SVM and turbine two (t=7, *poly*, C=0.1).

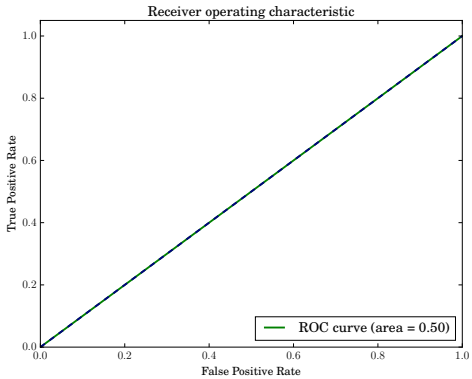


Figure 5.24: The ROC for SVM and turbine three (t=7, *poly*, C=0.1).

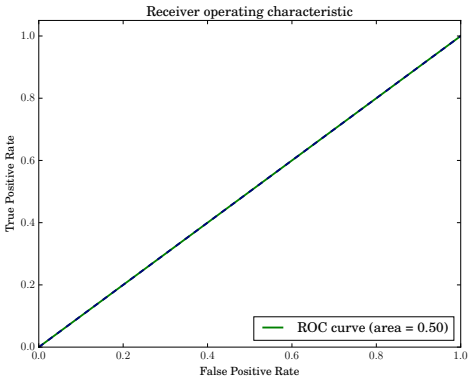


Figure 5.25: The ROC for SVM and turbine two (t=7, *poly*, C=1.5).

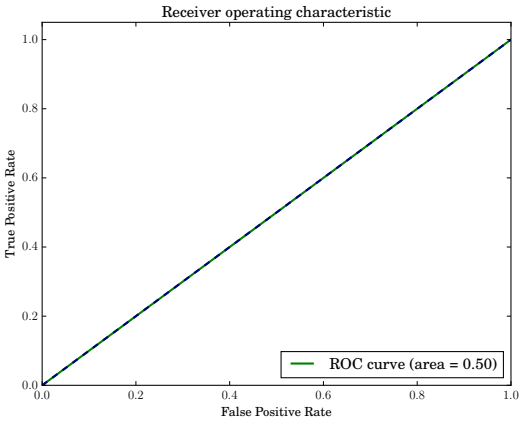


Figure 5.26: The ROC for SVM and turbine three (t=7, *poly*, C=1.5).

5.3 Discussion

As depicted in the previous section, both concepts, the SVM and the (conditional) restricted Boltzmann machine, could not predict events properly. The ROC curves in the illustration - which are in fact straight diagonal lines - cover only an area of 0.5. That is, the capability of both algorithms is not better as *random guess*. Although we have applied different types and various parameterizations, the models are not appropriate for our problem case, our data respectively. The findings show that the data provided do not exhibit predictive features that can be detected by this classifiers, even though we provided statistical moments (see section 3) to the methods. Investigating the details of incorrect classifications show two major issues:

- The GBRBM does very likely classify any time series as anomalous, thus predicting an event. This strongly coheres with the setting of the threshold τ_{th}. As outlined the threshold include a value for each feature and an unacceptable deviation for at least one of these is very likely to occur. Therefore the correct setting of τ_{th} is crucial.
- On the contrary the SVM is not predicting any future event, even when error penalty term C is reduced, which might encourage the method to classify more data points as events. Also the kernel choice does not influence the performance. Obviously the SVM achieves a better cost function when not correctly adjusting the hyperplane, but accepting miss-classifications. The reason might be a *skewed distribution* of the reconstruction errors, which exacerbates the classification task as well as the fact that the data comprises only few events.

Furthermore one must state that the cRBM performed better in training as the standard RBM with a 50% better reconstruction error.

6 Conclusions

The final section shall conclude this thesis by stating and briefly describing accomplishments and gives implications for practical as well as theoretical future work.

6.1 Accomplishments

Although, the results have not been as compelling as expected (see previous section), the derived concept doubtlessly contributes in regards to following issues:

- The relatively young concept of restricted Boltzmann machine with continuous visible units has been applied to time series data of another domain. As the training process depicts, the RBM can learn performance related data of machinery with an appropriate parametrization.
- Omitting the classifying step and and aiming for regression instead, the RBM can be considered as a promising method in predictive analytics.
- The usage of a RBM for classification purposes is heavily depending on the correct setting of a threshold vector, which should be data-driven. Concretely the threshold values should be appropriately coupled to the distribution of the reconstruction error of the respective feature. The simple mean value of the batch, as implemented in this work, does obviously not suffice.
- The conditional Gaussian Bernoulli restricted Boltzmann machine achieves a better modeling of the training data as the standard GBRBM. In fact the reconstruction error is by 50% lower. Crucial for the convergence is the setting of the appropriate regional interdependency, i.e. the amount of past units connected to the current ones. If one choses the amount too large the result is a volatile performance.

6.2 Implications for Future Work

An obvious implication is the further improvement of the methodology in this specific case, such as optimization of the parameters of the SVM based on a thorough investigation of the reconstruction error distributions with e.g. simulated annealing ([41]). Since this investigation dealt with a limited amount of very specific data cRBMs should be applied to different data sets for further investigating applicability.
The concept, however, lacks the capability for online adaption so far, which could be worth investigating. Furthermore the concept of conditional RBM seems not quite evolved at this point. To our best knowledge there have not been any investigation in regards to alternative

ways to connect past observations to the current ones. Finally the work can be seen as a starting point for additional enhancement of predictions with meaningful decision prescriptions, in the sense of *prescriptive analytics*. A link to entrepreneurial circumstances thus could enlarge practical benefits. Concretely in this case the optimization of repair scheduling of a wind turbine given meteorological data or energy price forecasts could decrease maintenance costs.

Bibliography

[1] Rakesh Agrawal, T. Imieliński, and A. Swami. "Mining association rules between sets of items in large databases". In: *Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93*. 1993, p. 207

[2] Nesreen K. Ahmed et al. "An Empirical Comparison of Machine Learning Models for Time Series Forecasting". In: *Econometric Reviews* 29.5 (2010), pp. 594–621

[3] Ana Azevedo and Emanuel Filipe Santos. "KDD, SEMMA and CRISP-DM: a parallel overview". In: *IADIS European Conference on Data Mining*. 2008

[4] Randall Kenyon Bartlett. *A practitioner's guide to business analytics: Using data analysis tools to improve your organization's decision making and strategy*. New York NY: McGraw Hill, 2013. ISBN: 978-0-07-180759-3

[5] Jason Bell. *Machine Learning: Hands-on for developers and technical professionals*. Indianapolis, Indiana: John Wiley & Sons, Inc., 2015. ISBN: 978-118-88906-0

[6] Asa Ben-Hur and Jason Weston. "A user's guide to support vector machines". In: *Methods in molecular biology (Clifton, N.J.)* 609 (2010), pp. 223–239. ISSN: 1064-3745. DOI: 10.1007/978-1-60327-241-4_13

[7] Yoshua Bengio et al. "Greedy Layer-Wise Training of Deep Networks". In: *Advances in neural information processing systems 19*. Ed. by Bernhard Schölkopf, John C. Platt, and Thomas Hofmann. Vol. 19. Advances in neural information processing systems. Cambridge, MA: MIT Press, 2007. ISBN: 1282162004

[8] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. New York, NY: Springer, 2006. ISBN: 9780387310732. URL: http://www.loc.gov/catdir/enhancements/fy0818/2006922522-d.html

[9] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers". In: *COLT '92 Proceedings of the fifth annual workshop on Computational learning theory*. 1992, pp. 144–152

[10] Erik Brynjolfsson, Lorin M. Hitt, and Heekyung Hellen Kim. "Strength in Numbers: How Does Data-Driven Decisionmaking Affect Firm Performance?" In: *SSRN Electronic Journal* (2011). ISSN: 1556-5068. DOI: 10.2139/ssrn.1819486

[11] Bundesministerium für Wirtschaft und Energie, Referat Öffentlichkeitsarbeit, ed. *Informationsportal Erneuerbare Energien: Finanzierung*. 04.10.2016. URL: http://www.erneuerbare-energien.de/EE/Navigation/DE/Technologien/Windenergie-auf-See/Finanzierung/finanzierung.html

[12] Bruno Burger. *Stromerzeugung aus Solar- und Windenergie im Jahr 2015*. Freiburg, 13.01.2016. URL: https://www.ise.fraunhofer.de/de/downloads/pdf-files/aktuelles/folien-stromerzeugung-aus-solar-und-windenergie-im-jahr-2015.pdf, visited on 11/07/2016

[13] Christopher J.C. Burges. "A Tutorial on Support Vector Machines for Pattern Recognition". In: *Data Mining and Knowledge Discovery* 2 (1998), pp. 121–167

[14] H. Chen and A. F. Murray. "Continuous restricted Boltzmann machine with an implementable training algorithm". In: *IEE Proceedings - Vision, Image, and Signal Processing* 150.3 (2003), p. 153. ISSN: 1350245X. DOI: 10.1049/ip-vis:20030362

[15] Kyung Hyun Cho, Tapani Raiko, and Alexander Ilin. "Gaussian-Bernoulli deep Boltzmann machine". In: *International Joint Conference on Neural Networks (IJCNN 2013 - Dallas)*. 2013, pp. 1–7. DOI: 10.1109/IJCNN.2013.6706831

[16] Christopher Clifton. *Encyclopædia Britannica*. URL: https://www.britannica.com/technology/data-mining, visited on 11/06/2016

[17] Tom Fawcett. *ROC Graphs: Notes and Practical Considerations for Data Mining Researchers*. Tech. rep. HP Laboratories Palo Alto, 2003

[18] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. "The KDD process for extracting useful knowledge from volumes of data". In: *Communications of the ACM* 39.11 (1996), pp. 27–34. ISSN: 00010782. DOI: 10.1145/240455.240464. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.27.2315&rep=rep1&type=pdf, visited on 06/06/2016

[19] Steven Finlay. *Predictive Analytics, Data mining and Big Data: Myths, Misconceptions and Methods*. 1. publ. Basingstoke: Palgrave Macmillan, 2014. ISBN: 9781137379290. URL: http://dx.doi.org/10.1057/9781137379283

[20] Asja Fischer and Christian Igel. "Training restricted Boltzmann machines: An introduction". In: *Pattern Recognition* 47.1 (2014), pp. 25–39. ISSN: 00313203. DOI: 10.1016/j.patcog.2013.05.025

[21] Fausto Pedro García Márquez et al. "Condition monitoring of wind turbines: Techniques and methods". In: *Renewable Energy* 46 (2012), pp. 169–178. ISSN: 09601481. DOI: 10.1016/j.renene.2012.03.003

[22] Robert Gasch, Jochen Twele, and P. Bade. *Windkraftanlagen: Grundlagen, Entwurf, Planung und Betrieb*. 4., vollständig überarbeitete und erweiterte Auflage 2005. Wiesbaden: Vieweg+Teubner Verlag, 2005. ISBN: 978-3-322-99446-2. URL: http://dx.doi.org/10.1007/978-3-322-99446-2

[23] Stuart Geman and Donald Geman. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6.6 (1984), pp. 721–741. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1984.4767596

[24] Z. Hameed et al. "Condition monitoring and fault detection of wind turbines and related algorithms: A review". In: *Renewable and Sustainable Energy Reviews* 13.1 (2009), pp. 1–39. ISSN: 13640321. DOI: 10.1016/j.rser.2007.05.008

[25] Erich Hau. *Windkraftanlagen: Grundlagen, Technik, Einsatz, Wirtschaftlichkeit*. 4., vollst. neu bearb. Aufl. Berlin and Heidelberg: Springer, 2008. ISBN: 9783540721505. DOI: 10.1007/978-3-540-72151-2. URL: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10246042

[26] Geoffrey E. Hinton. *A Practical Guide to Training Restricted Boltzmann Machines*. Tech. rep. 6 King's College Rd, Toronto: Departement of Compute Science, University of Toronto, 2010. URL: https://www.cs.toronto.edu/~hinton/absps/guideTR.pdf

[27] Geoffrey E. Hinton. "Training products of experts by minimizing contrastive divergence". In: *Neural computation* 14.8 (2002), pp. 1771–1800. ISSN: 0899-7667. DOI: 10.1162/089976602760128018

[28] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7 (2006), pp. 1527–1554. ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.7.1527

[29] International Energy Agency. *World Energy Outlook*. 2015

[30] A. J. Koning et al. “The M3 competition: statistical tests of the results”. In: *International Journal of Forecasting* 21 (2005), pp. 397–409

[31] Martin Längkvist. “Modelling Time-Series with Deep Networks”. Doctoral Dissertation. Örebro: Örebro University, 2014

[32] Martin Längkvist, Lars Karlsson, and Amy Loutfi. “A review of unsupervised feature learning and deep learning for time-series modeling”. In: *Pattern Recognition Letters* 42 (2014), pp. 11–24

[33] Shih-Wei Lin et al. “Parameter determination of support vector machine and feature selection using simulated annealing approach”. In: *Applied Soft Computing* 8.4 (2008), pp. 1505–1512. ISSN: 15684946. DOI: 10.1016/j.asoc.2007.10.012

[34] Irv Lustig et al. *The Analytics Journey: An IBM view of structured data analysis landscape: descriptive, predictive and prescriptive analytics.* 2010. URL: http://analytics-magazine.org/the-analytics-journey/

[35] Hans-Joachim Meier. *Konzeptstudie Referenzkraftwerk Nordrhein-Westfalen (RKW NRW).* Tech. rep. Essen, 2003

[36] Qian Miao and Viliam Makis. “Condition monitoring and classification of rotating machinery using wavelets and hidden Markov models”. In: *Mechanical Systems and Signal Processing* 21 (2007), pp. 840–855

[37] Tom Michael Mitchell. *Machine learning.* International ed. McGraw-Hill series in computer science. New York, NY: McGraw-Hill, 1997. ISBN: 0071154671

[38] Guy P. Nason. “A test for second-order stationarity and approximate confidence intervals for localized autocovariances for locally stationary time series”. In: *Journal of the Royal Statistical Society* 75 (2013), pp. 879–904

[39] Guy P. Nason. “Stationary and non-stationary time series”. In: *Statistics in Volcanology.* 2006, pp. 129–142

[40] Charles Nyce. *Predictive Analytics White Paper.* Tech. rep. American Institute for CPCU - Insurance Institute of America, 2007

[41] Ping-Feng Pai and Wei-Chiang Hong. “Support vector machines with simulated annealing algorithms in electricity load forecasting”. In: *Energy Conversion and Management* 46.17 (2005), pp. 2669–2688. ISSN: 01968904. DOI: 10.1016/j.enconman.2005.02.004

[42] Athanasios Papoulis and S. Unnikrishna Pillai. *Probability, random variables, and stochastic processes*. 4. ed., internat. ed. Boston, Mass.: McGraw-Hill, 2002. ISBN: 9780071226615

[43] M. B. Priestley and T. Subba Rao. "A Test for Non-Stationarity of Time-Series". In: *Journal of the Royal Statistical Society* 31.1 (1969), pp. 140–149

[44] Foster Provost and Tom Fawcett. *Data Science for Business: What You Need to Know About Data Mining and Data-Analytic Thinking*. Sebastopol, CA: O'Reilly Media, Inc., 2013. ISBN: 978-1-449-36132-7

[45] Peng Qian, Xiandong Ma, and Yifei Wang. "Condition monitoring of wind turbines based on extreme learning machine". In: *21st International Conference on Automation and Computing (ICAC)*, pp. 1–6. DOI: 10.1109/IConAC.2015.7313974

[46] Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 2nd ed. Prentice Hall, 2003. ISBN: 978-0137903955

[47] Bernhard Schölkopf, Alex J. Smola, and K.-R Müller. "Nonlinear component analysis as a kernel eigenvalue problem". In: *Neural computation* 10 (1998), pp. 1299–1319. ISSN: 0899-7667

[48] ScottishPower Renewables (UK) Lt., ed. *East Anglia THREE*. 04.10.2016. URL: http://www.scottishpowerrenewables.com/pages/east_anglia_three.asp

[49] Colin Shearer. "The CRISP-DM model: the new blueprint for data mining". In: *Journal of Data Warehousing* 5.4 (2000), pp. 13–22

[50] Jonathon Shlens. "Notes on Kullback-Leibler Divergence and Likelihood". In: (2014). URL: https://arxiv.org/abs/1404.2000

[51] Paul Smolensky. "Chapter 6: Information Processing in Dynamical Systems: Foundations of Harmony Theory". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Ed. by David E. Rumelhart and James. L. McLelland. MIT Press, 1986, pp. 194–281. ISBN: 0-262-68053-X

[52] Sutton R. S. "Two problems with backpropagation and other steepest-descent learning procedures for networks". In: *Proceedings of the 8th Annual Conference Cognitive Science Society*. 1986

[53] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson New International Edition. Harlow: Pearson, 2014. ISBN: 9781292026152

[54] Graham W. Taylor and Geoffrey E. Hinton. "Factored conditional restricted Boltzmann Machines for modeling motion style". In: *Proceddings of the 26th Annual International Conference on Machine Learning*. Ed. by Andrea Danyluk, Léon Bottou, and Michael Littman. 2009, pp. 1–8. DOI: 10.1145/1553374.1553505

[55] Graham W. Taylor, Geoffrey E. Hinton, and Sam Roweis. "Modeling Human Motion Using Binary Latent Variables". In: *Advances in neural information processing systems 19*. Ed. by Bernhard Schölkopf, John C. Platt, and Thomas Hofmann. Vol. 19. Advances in neural information processing systems. Cambridge, MA: MIT Press, 2007, pp. 1345–1352. ISBN: 1282162004

[56] William Vorhies. *Prescriptive versus Predictive Analytics - A Distinction without a Difference*. 2014. URL: http://www.datasciencecentral.com/profiles/blogs/prescriptive-versus-predictive-analytics-a-distinction-without-a

[57] Christopher A. Walford. *Wind turbine reliability: Understanding and minimizing wind turbine operation and maintenance costs*. Tech. rep. Albuquerque, New Mexico 87185: Sandia National Laboratories, 2006. DOI: 10.2172/882048. URL: http://prod.sandia.gov/techlib/access-control.cgi/2006/061100.pdf, visited on 05/18/2016

[58] Nan Wang, J. Melchior, and L. Wiskott. "An Analysis of Gaussian-Binary Restricted Boltzmann Machines for Natural Images". In: *Proceedings of the 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. Bruges, Belgium, 2012, pp. 287–292

[59] Florian Wilhelm. *Leveraging the Value of Big Data with Automated Decision Making*. 2016. URL: http://blogs.csc.com/2016/04/11/leveraging-the-value-of-big-data-with-automated-decision-making/, visited on 04/11/2016