Electricity Load Modelling using Computational Intelligence

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. J.T. Fokkema,
voorzitter van het College van Promoties,
in het openbaar te verdedigen op
woensdag 14 december 2005 om 10:30 uur
door

Rutger Willem TER BORG

informatica ingenieur
geboren te Delfzijl.

Dit proefschrift is goedgekeurd door de promotor:

Prof. dr. H. Koppelaar

Toegevoegd promotor:

Dr. drs. L.J.M. Rothkrantz

Samenstelling promotiecommissie:

Rector Magnificus, voorzitter
Prof. dr. H. Koppelaar, Technische Universiteit Delft, promotor
Dr. drs. L.J.M. Rothkrantz, Technische Universiteit Delft, toegevoegd promotor
Prof. dr. ir. J. Biemond, Technische Universiteit Delft
Prof. ir. L. van der Sluis, Technische Universiteit Delft
Prof. dr. A. Heertje, Universiteit van Amsterdam
Prof. dr. H.J. van den Herik, Universiteit Maastricht
Dr. J.J. Battjes, Nuon NV

This work was supported by:

Nuon NV
Spaklerweg 20
Amsterdam

Typeset by the author using LaTeX 2$_\varepsilon$.

ISBN 90-8559-118-X.

"I have made this letter longer than usual, because I lack the time to make it short."
Blaise Pascal (1623–1662)

Preface

During my research for doctorate, a number of events occurred with world wide impact, such as the tragedy of the collapse of the twin towers, the introduction of the Euro, terrorist attacks on Madrid and London, and to a lesser extent, the many different Dutch governments during such a short period.

I have spent a tremendous amount of time on my research. At times, it has been difficult for me to keep up the pace necessary to complete this work. Luckily, computers were there to take over some of the tedious tasks such as keeping track of all 705 remarks on my thesis, redrawing figures within seconds, and running numerous tests. I am confident that I have seen LyX more than I will see it ever again in my life.

It was Henk Koppelaar who brought me in contact with Thijs van den Berg of Nuon. I would like to thank them both for giving me the opportunity to conduct research for doctorate and for all their support. Léon Rothkrantz has given me many valuable comments and suggestions, also on my work. Floris Ouwendijk developed a useful tool to test several machine learning techniques. Thanks also go to Coos Battjes, Melissa Brinkman, Jan van den Bor, Sebastiaan Hers, Mirjam Nieman, Willem van Rossum, colleagues of the Monitoring & Forecasting desk, the members of my thesis committee, and everyone else who thinks he should be listed here.

My grandfather Wim spent more time proofreading this thesis than anyone else. This delivered quite a few comments, which he submitted over the Internet with remarkable ease. My parents Bart and Anja were able to distract me from the wonders of science with, for example, a beautiful safari through Kenya, or the occasional Tulpenrallye. And, of course, I am very grateful to my girlfriend Fleur, who kept me motivated, and who had to relinquish a large amount of our time together during this period. Fortunately, we could partly compensate for that with an extra-ordinary trip through Indonesia, and I am sure that that trip was only the beginning of our journeys.

For sure, the last four years were a turbulent period of my life. I have learnt a great deal in scientific and personal areas. Having said all this, let us start with the thesis itself.

Contents

"There is no plea which will justify the use of high-tension and alternating currents, either in a scientific or in a commercial sense."

Thomas Edison (1847–1931)

1 Introduction

The energy markets across Europe have been liberalised as per EU directive 96/92/EC [57], which sparked the energy market sector to become much more turbulent. The main objective of liberalisation was the introduction of competition, which can have multiple benefits for customers. Patel and Samuel [99] mention that these benefits could include pricing efficiency and transparency, product and technical innovation, improvement of quality of service, and responsiveness to changing market conditions.

Since the EU directive on the internal market for electricity, some countries opened their markets rather quickly, while the Netherlands chose to implement the directives more slowly. The Dutch scheme is represented by the Electricity Act [133], by which full liberalisation is in effect as of mid-2004 [111]. After that date all consumers became free to choose their electricity supplier.

The Dutch independent transmission system operator (TSO) Tennet is a key figure in the Dutch electricity market, and has a number of responsibilities. It has to make sure, to prevent black-outs, that the demand for and supply of electricity are in balance at all times. Tennet also has to keep the high-voltage network in a good state of repair. Any participant on the electricity grid that causes an imbalance, will have to settle with Tennet for correcting it.

The Dutch energy company Nuon devised a plan to reduce its imbalance costs, starting with a scientific study on modelling electricity loads. The motive for my research are these imbalance costs, and the goal of my research is to engineer methods, models, and tools that will enable a reduction of the imbalance. This thesis is a report about this scholarly research.

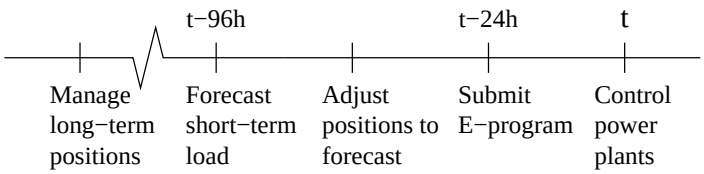

FIGURE 1.1: Time horizon of activities done by a program responsibility partner until the program time unit at time t.

1.1 Imbalance Costs

First, let us review what an imbalance exactly means, how it is determined, how it is settled, and what can be done to reduce it. Roughly said, an imbalance is a deviation between supply and demand. Or, to state it in terms of current regulations, an imbalance is the deviation between an E-program and the allocated load, which will be treated in subsections 1.1.1 and 1.1.2. The settlement procedure and thus the cost associated with an imbalance is highlighted in subsection 1.1.3. This section about imbalance costs is concluded with a number of approaches to the reduction of the imbalance.

1.1.1 E-Programs

To enable Tennet to check the expected demand and supply of electricity, all agreements on the electricity market have to be reported to Tennet in the form of *Electricity-programs* (E-programs). E-programs cover periods of 96 program time units (PTUs), currently set at fifteen minutes, and have to be submitted to Tennet a day in advance at noon. Only program responsibility partners (PRPs) are allowed to make transactions on the electricity market. Two types of PRPs are distinguished: those with full acknowledgement and those with trade acknowledgement. Only a PRP who has full acknowledgement is allowed to exchange electricity physically, so only this kind of PRP can have an imbalance. Program responsibility partners come with a variety of roles on the energy market: they can be suppliers, producers, consumers, traders, or, more commonly, a combination of all of these.

Figure 1.1 shows a time horizon of activities typically performed by PRPs up until the PTU. They will manage their long-term positions, such as long-lasting contracts that have been signed with other PRPs. For long-term and medium-term deals, trading is supported by brokers who have a market-neutral position: their concern is to effect transactions in electricity on which a provision is earned. Alternatively, they trade at Endex, a centralised market for long-term set up by Benelux market participants. Some PRPs own power plants with which they are to meet the demand of their customers in the immediate and further future.

Several days in advance, they start making short-term forecasts of the expected loads during the coming PTUs. On the basis of these forecasts, they will adjust their long-term positions to match these expected short-term loads in a variety of ways. They will buy and sell electricity from elsewhere on the market, for instance from and to producers, or on a day-ahead basis at the Amsterdam power exchange (APX) [127]. A day in advance at noon, E-programs have to be submitted to Tennet.

The last 24 hours before physical delivery, decisions about the production process come into play. Producers will dispatch their power plants in the most cost-efficient way possible, also known as economic dispatching. Broadly said, inflexible power plants are cheap and will run continuously, and flexible power plants are expensive and will run to counterbalance the more resilient uses of electricity. Some power-producing facilities are hard to control, e.g., wind energy installations. Because here the energy produced strongly depends on the actual wind speed at a certain location, the power output cannot be guaranteed at all times. Unexpected variations in their production levels of electricity have to be counterbalanced by flexible and expensive power production facilities.

In the end, roughly 105% of the electricity demand by consumers will be produced. They occupy the opposite side of the electricity market with respect to producers. The additional ∼5% is due to network transportation losses.

1.1.2 Allocation

Tennet does not measure the total load of each program responsibility partner per PTU. This task is delegated to regional grid administrators, who have to submit each PRP's definite load allocation to Tennet within 14 days after expiration of that PTU. Regional grid administrators are positioned between Tennet and the other participants. They have a neutral position: they may deliver transportation services to all producers, consumers, suppliers and traders. Pricing of these services is described in the tariff code [67] and boils down to the following: small customers (such as households) pay a flat-rate fee, larger customers have more sophisticated price structures.

The total load of each regional grid administrator is measured per PTU. For each PTU it measures all telemetered connections, and estimates the remaining part. Regional grid administrators do not measure the load of all their connections per PTU; e.g., most households and small businesses have an electricity consumption meter which is read-out once a year only. To address this problem, regional grid administrators adhere to a number of standard profiles on the basis of categories. Each non-measured connection is assigned one of these standardised profiles. All profiles are normalised over a year, i.e., it adheres to $\sum_t \mathsf{profile}_c(t) = 1$ for t within one calendar year and for each profile category c. A profile together with a customer's yearly volume determine a customer's load per PTU.

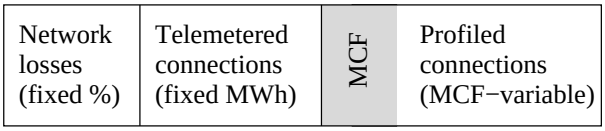

FIGURE 1.2: Determination of the allocated electricity load for each program time unit.

Figure 1.2 illustrates the break down of a regional grid administrator's measured total load. The first part (most left in the figure) are network losses due to the transport of electricity, set at a fixed percentage at all times. A grid administrator is responsible for its network losses, of which the percentage is not necessarily the same for all regional grid administrators. The second part from the left depicts the connections that are measured per PTU by telemeters. The volume in this category is obtained by adding up all telemetered readings. Electricity volume allocated to the profiled (non-measured) connections is the total volume decreased by the network losses and the telemetered connections. A measuring correction factor (MCF) is applied to the aggregate volume of the profiled connections to correct for errors caused by the estimation. The MCF is published by each grid administrator for each PTU.

In other words, each regional grid administrator computes the load for each PRP in the following manner

$$\mathsf{load}_{\mathrm{PRP}}(t) = \mathsf{MCF}(t) \sum_i \mathsf{volume}_{\mathrm{PRP},i} \mathsf{profile}_i(t) + \sum_j \mathsf{measured}_{\mathrm{PRP},j}(t) \qquad (1.1)$$

where $\mathrm{load}_{\mathrm{PRP}}$ is the load of that PRP at time t of the PTU, MCF is the applied measuring correction-factor (MCF), volume_i is the total yearly electricity consumption volume of that PRP's customers in profile group i, $\mathsf{profile}_i$ is one of the determined default profiles, and $\mathsf{measured}_j$ are measured connections of that PRP. Values of all variables in (1.1) are sent to the PRPs, on the basis of which a time series of historic values can be constructed. In case of a power production facility, both profiled connections and measured connections can have negative values. An example of a negative profile allocation is a non-measured power production facility such as some wind-power turbines. They are assigned a negative yearly volume.

1.1.3 Imbalance Settlement

At the time of the PTU, imbalances are corrected by Tennet, who will buy or sell what is needed to correct each (fully acknowledged) PRP's imbalance. The imbalance price is a sum of the resulting price of the bid ladder price and the

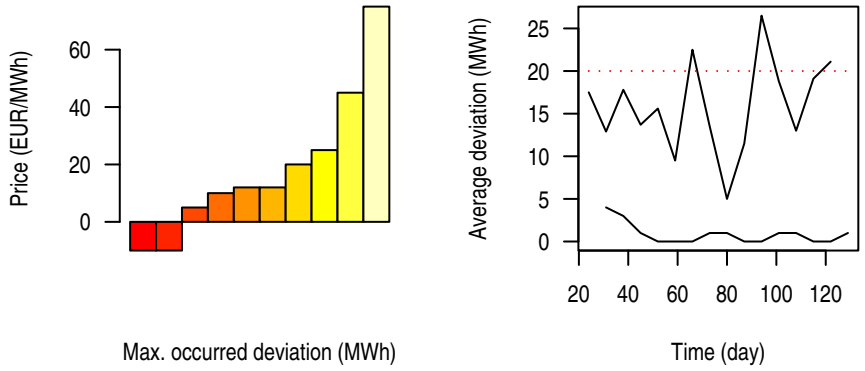

FIGURE 1.3: Price bid ladder (left) and incentive component (right).

incentive component, $\text{price}(t) = \text{bid ladder price}(t) + \text{incentive}(t)$ with t the time of the PTU. These two components are discussed below.

- **Bid ladder price.** Tennet only corrects the overall imbalance, the sum of the imbalances of all individual PRPs. Before the PTU takes place, regulating and reserving power suppliers (RRPSs) offer Tennet the opportunity to adjust their electricity production either upward or downward. Such an offer is presented to Tennet by means of a *price bid ladder*, which is a list of prices per MWh increasing with the magnitude of deviation. Interested RRPSs are permitted to announce their prices until one hour before a PTU. From all submitted price bid ladders, Tennet assembles the cheapest possible overall price bid ladder, of which an example is shown in figure 1.3. So, during the PTU, the cheapest RRPSs are called upon to regulate. If the actual deviation exceeds the maximum value of the price bid ladder chart, the emergency power suppliers are mobilised to fulfil the unexpectedly high demand. The price for this is at least 10% higher than the maximum price bid ladder price. The price of the most expensive regulation applicable to a PTU will be the resulting price for that (entire) PTU. Prices can also be negative; this occurs when PRPs have collectively overestimated their E-programs.
- **Incentive component.** Compared with prices on the price bid ladder, this component is mostly relatively small. Based on a week observed in 5-minute intervals, the incentive component is raised (in theory, that is) if during that period at least one of the following conditions is true: the number of deviations absolutely larger than 300 MW exceeds 40 or the average

deviation is larger than 20 MW. Figure 1.3 (right) illustrates a part of this decision-making process. It shows that if the average deviation (the upper solid line) reaches levels above 20 mega watts (MW, the dotted line), the incentive component price is raised three days after this criterion has been observed (the lower solid line).

The *imbalance settlement* between Tennet and each PRP is calculated by the size of a PRP's deviation in megawatt-hours (MWh) times the imbalance price in € per MWh for that PTU,

$$\mathsf{settlement}(t) = (\mathsf{Eprogram}(t) - \mathsf{allocation}(t)) \times \mathsf{price}(t)$$

with t the time of the PTU, and settlement the sum paid or received by a PRP. A PRP can receive a sum of money when it has a positive imbalance, e.g., in case it produces more that its E-program states, and other PRPs hold a negative imbalance. A PRP with a long position (positive imbalance) will be paid by Tennet for its extra produced electricity. Imbalance settlement costs can be very high, e.g., when the overall correction needed is large; this occurs typically when PRPs collectively hold a long or a short position, or when a PRP has a large deficient position.

1.1.4 Imbalance Reduction Strategies

PRPs with full responsibility face the challenge of keeping their imbalances as small as possible. In order to minimise them, the allocated volumes should match the E-programs as accurately as possible. The two most important factors contributing to deviations between E-programs and allocated volumes are uncertain future demand and uncertain future production. Imbalances can be reduced in several ways. I mention three of them.

- **Improve portfolio effects.** A large portfolio of a variety of customers may show a *pooling effect*: customers complementing each other in their consumption patterns in such way that the uncertainty of their collective demand is smaller than that of one individual customer. A great deal of the efficiency improvement in the electricity market can be accounted for by the knowledge one has been able to gather about its customers [134]. PRPs need a good assessment of how their customers react under different circumstances.

- **Improve short-term demand forecasts.** As submitted E-programs cannot be altered, the coming demands must be predicted as accurately as possible. In other words, a PRP needs a good short-term electricity demand forecast.

- **Improve short-term production forecasts.** Many renewables have the drawback that their future production strongly depends on the weather conditions. Most dominantly present in this category is wind-power production, which already produces considerable amounts of energy in the Netherlands during windy times.

These issues can be more or less leading in how to reduce an imbalance. A shared theme across all approaches is how to make the demands and power-production levels more transparent. With a better insight on these fronts PRPs will be able to take better founded decisions thus improving the efficiency of their operations.

1.2 Research Goals

As announced at the beginning of this chapter, the goal of my PhD research is to engineer methods, models, and tools that could enable reduction of the imbalances of a PRP. To achieve this, the following will be the two key aspects during my research: to obtain a predictive model describing electricity demands and to obtain a predictive model for short-term wind-power production.

- **Electricity-Demand Model.** The target application of the electricity demand model will have to include long-term scenarios as well as short-term forecasting. If used for back-casting, gaps present in time series of electricity load can be filled with model predictions. Historical electricity demand patterns can be normalised, i.e., demands predicted for an average-weather scenario. This information can be used, e.g., for determining prices for clients. Using short-term weather forecasts supplied by a meteorological institute, the model should be able to make short-term forecasts of the aggregated expected load of a PRP. A key challenge will be to cover both long-term and short-term in one model.
- **Wind-Power Production Forecast.** The endeavour to reduce the total amount of carbon dioxide in the atmosphere [50] has been ongoing for a while. The European Union (EU)'s renewables directive aims to raise the share of electricity produced from renewable energy sources (RES) in the EU to 22% by 2010 [51]. The efficiency of wind-energy turbines has been significantly improved during the last decade, and they have become an attractive source of renewable energy. At the moment, wind energy is the fastest growing type of renewable energy in Europe. However, due to the unreliability of wind energy production, it is also increasingly responsible for larger imbalances. These can be reduced with a more accurate short-term forecast of wind-power production.

The key challenge will be to obtain predictive models on the basis of recordings of allocations of electricity demands and of recordings of wind-power production

that have been made available for this research. Whereas classical statisticians typically assume that the form of the correct model is known and the objective is to estimate the model parameters, artificial intelligence researchers devise and use many ways to automate the task of constructing a predictive model from the data. The current state-of-the-art way to construct a model from the data are *kernel machines*, of which the successful *support vector machine* is the most frequently cited type. Kernel machines use either statistical learning theory or hierarchical Bayes to decide whether a particular model is adequate or whether a different model would produce better predictions. They can be regarded to be the theoretically better founded successors to artificial neural networks, which have dominated the machine-learning field since the 1980s. Kernel machines have quite literally taken the machine-learning field by storm, and my research is not an exception.

1.3 Thesis Outline

In chapter 2, a review of existing literature on the subject of modelling electricity load is given. Commonly used approaches and typical solutions and techniques are discussed. Criteria of quality are also discussed in this chapter. Chapter 3 introduces my novel smooth Bayesian kernel machines [13]. In chapter 4, the concepts of kernel machines and electricity demand patterns are merged to a model [14]. A model for short-term wind-power production forecasting is also proposed [15]. Chapter 5 discusses how to implement kernel machines in an efficient way with a freely available software library that I wrote: the Kernel-Machine Library. Chapter 6 discusses experiments with the models that have been implemented and tested. Finally, chapter 7 concludes this thesis.

“Copy from one, it is plagiarism; copy from two, it is research.”

Wilson Mizner (1876–1933)

2 Related Work

A well-studied field that relates closely to modelling electricity demand is the field of electricity load forecasting [59, 72]: in many cases it also involves obtaining a model that describes electricity demands only. Section 2.1 gives a short introduction to this field, and elaborates on common approaches to modelling electricity demands. The area of wind-power production forecasting is younger and smaller than the field of electricity load forecasting, and is treated separately in literature. Section 2.2 discusses typical directions taken, and plans of attack used to obtain a short-term wind-power production forecast. Overlap of the two fields can be found in their used regression techniques, which will be topic of discussion in section 2.3. Section 2.4 concludes the chapter with a discussion of quality criteria.

2.1 Electricity Demand Models

As with many fields of study in applied science, the subject of electricity load forecasting is quite broad. First of all, types of forecasting can be identified to be *spatial* or *non-spatial* [6]. Spatial forecasting [see, e.g., 94] concentrates on the prediction of the electricity consumption patterns of a specific geographical area such as a city, an island, or a country. This information is used by (regional) politics to determine future directions of the energy policy. Non-spatial forecasting deals with the future electricity load of specific consumers without special restrictions on their geographic position, e.g., that of the customer base of a certain electricity supplier, or that of a large world-wide company.

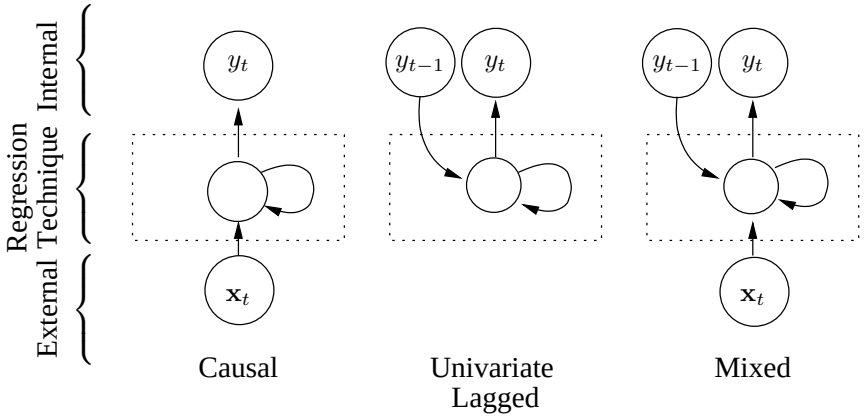

FIGURE 2.1: Commonly encountered structures of predictive models used for electricity load modelling.

Furthermore, differentiating on the basis of forecast time-horizon, *long-term*, *medium-term,* and *short-term* forecasting are used for different purposes [135]. Long-term forecasting is mainly used for system planning and typically spans periods of 10 to 20 years. Key factors of interest in long-term prediction include the type and level of economic activity, population growth, price of alternative sources of energy, factors such as marketing, conservation campaigns and environmental changes. Medium-term forecasting usually covers periods of a few weeks and is used for estimating fuel (storage) requirements and for planning the execution of maintenance programs. Short-term load forecasting has a time span of less than a week, and is employed for day-to-day operation, scheduling of power plants, and setting targets for E-programs.

I distinguish three types of electricity demand models: causal models, univariate lagged models, and mixed models. The separation is made on the basis of what variables are in use by these models, and is illustrated in figure 2.1. Two types of variables are distinguished: external and internal. Internal variables are considered to be the variables for which the predictive model is to be made, e.g., electricity demands or wind-power production levels. External variables are other types of influential variables that are considered, e.g., temperature and wind speeds. The arrows depict the flow of information (or values) in the model. If an arrow points to the same circle (or node), it means that historic information readings are kept in some kind of memory. Regression techniques are discussed in more depth in section 2.3. The causal, univariate lagged and mixed models are discussed next.

2.1.1 Causal Model

In this thesis, I use the term causal model if the input space exclusively contains external variables and propagates to the output as in

$$\tilde{y}(t) = f(\mathbf{x}_i(t - \tau_{i,j})), \tag{2.1}$$

with t discretised time, $\tilde{y}$ the predicted electricity demand in $\mathbb{R}$, f the model's function, $\mathbf{x}_i$ an input variable in $\mathbb{R}^N$, $\tau_{i,j}$ a time lag variable offset, and i and j indices. A time lag is used for variables (such as temperature) that do not have an immediate effect on the electricity demands: e.g., it takes a while before a building cools down or warms up. Historical (or lagged) recordings can be introduced to make these time-delayed influences measurable. Input spaces where $\tau_{i,j} = 0$ for all values of i and j in (2.1) are often referred to as *direct* or *flat* models. This type of model is encountered frequently [71, 79, 90], and is used in combination with a large variety of computational learning methods, ranging from self-organising maps [27, 91, 97] to fuzzy logic controllers [129].

A recurring problem is that predictability of input variables may have an influence on the accuracy and forecast horizon of an electricity demand model. Examples of exactly predictable input variables are time of the day [45], day of the week [97], seasonality with month number [71], and holiday indicators to match holiday information. However, not all required inputs can be predicted accurately over a large time span. The temperature is a good example of this: because the relation between ambient temperature and electricity demands is important, an accurate prediction of future temperatures is needed if one wants to use such an established relation. The load predictor then builds on the results of these temperature forecasts [79]. As the temperature cannot be predicted with an acceptable degree of accuracy, it is sometimes not taken into consideration [33].

2.1.2 Univariate Lagged Models

Otherwise known as *local learning models*, univariate lagged models use only one variable in their input space, namely historical demand values. These univariate lagged models are expressed by

$$\tilde{y}(t) = f(y(t - \tau_k)), \tag{2.2}$$

with $y \in \mathbb{R}$ a historic electricity demand value, and with the remaining symbols identical to those used in (2.1). Univariate lagged models look at a certain window of historic electricity loads as their input space. The field of non-linear time series analysis is largely about determining the step size and length of the window to look at [16, 117].

Several regression techniques have been used, such as artificial neural networks [23, 24], support vector machines [107], fuzzy logic controllers [98], and

nearest neighbours [9]. One step ahead (or iterated) prediction in this class of models suffers from *error propagation:* values $y(t-\tau_i)$ become vulnerable values themselves. A way to address this is by making a separate model for each needed forecast time horizon. One of the biggest drawbacks of this type of models is that it is not able to fully contain a priori knowledge, such as the effect of public holidays.

2.1.3 Mixed Models

Another approach is to use both historical electricity demand measurements and external variables [7, 26, 27, 33]; they combine the self-containing information of signals with related variables and a priori knowledge. They can be formulated as

$$\tilde{y}(t) = f(\mathbf{x}_i(t-\tau_{i,j}), y(t-\tau_k)). \tag{2.3}$$

As with causal variables, it can be a problem if an input attribute is used which cannot be predicted accurately [33]. Also, like univariate lagged models, they suffer from error propagation.

2.2 Wind-power Production Models

An exhaustive literature survey of wind-power prediction is available in a report by Giebel et al. [54]. With respect to short-term forecasting of wind power, two different approaches are identified: a physical approach and a statistical approach. In a physical approach, one makes a description of the dynamics of the underlying system, based on complete knowledge of all its subsystems [16]. The statistical approach is, like the approach to electricity demand forecasting, to construct models from the data.

2.2.1 Physical Approach

Generally, in the physical approach, the underlying system is separated in three different subsystems, which we will discuss next.

- **To scale down.** During this phase, the localised numerical weather forecasts are scaled down to match the turbine hub height (the name can be misleading, because one could just as well scale up the numerical weather forecasts to turbine hub height). To estimate the wind power produced at some site, ideally one wants to have wind speed measurements done at that exact location. Often weather data are obtained by using weather models such as HIRLAM. Unfortunately, actual errors made by a HIRLAM model are not available because the actual wind speeds are not measured.

- **To convert to power.** In this step, it is assumed that the total amount of electricity produced by the turbines depends on wind speed only. Using the so-called power-curve of the turbines, the wind speeds are converted to an expected power production. Power curves are often obtained from the supplier of the wind turbines, or are estimated on the basis of measurements on the wind turbines themselves [80, 81].
- **To scale up.** The estimated power of a subset of the complete farm is scaled up to match the installed total wind-power production capacity, and then the efficiency of the wind farm is used to determine the farm's total output. This specific step has also been addressed by more sophisticated tools such as a fuzzy-neural network [101].

The results obtained are acceptable. However, they require a great deal of data management of all individual farms and changing characteristics thereof.

2.2.2 Statistical Approach

In the statistical approach, the model is inferred from the data. The common approach with this one-stage type of approach is to use neural networks [81] or another kind of regression technique [4]. It usually estimates the wind-power production in one step, by taking the numerical weather predictions and transforming them to the estimated wind-power production. This is the same approach as the one that is taken to electricity demand modelling. This class of models works in a more implicit way: the numeric weather predictions are translated to a wind-power production forecast in one single step. A drawback of this approach is that it is a black box, i.e., its one and only function is to predict the expected wind-power production. For instance, additional information such as the performance of individual wind farms does not become available.

2.3 Used Regression Techniques

Let us consider a data set $\mathcal{D} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ containing N input-output pairs $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, with $\mathcal{X}$ typically containing multidimensional vectors in $\mathbb{R}^M$, and $\mathcal{Y}$ typically representing scalars in $\mathbb{R}$. Then, a *regression problem* is the task to find a relation

$$y(\mathbf{x}) = f(\mathbf{x}) + \varepsilon \tag{2.4}$$

that maps *any* input $\mathbf{x} \in \mathcal{X}$ (including unseen ones) to an output y assuming an additive error ε. Generalisation to unobserved measurements implies the difficulty of computational learning [70, 122]: it is the cornerstone of mainstream computational learning research. One of the ways to obtain the relation in (2.4) is by *supervised learning*, where examples are presented in the shape of input-output pairs. A selection of supervised learning methods, used for both electricity

demand modelling [83] and short-term wind-power prediction [54], is described below.

2.3.1 Nearest Neighbours

The simplest approach to forecasting is by reproduction of the past. In this type of model, one devises some sort of distance metric on the input space $\mathcal{X}$. Predictions for a certain $\mathbf{x}$ are made by querying *all* of the historical data, to find the measurements that have their inputs closest to that $\mathbf{x}$. It then returns the output of that historical reading. Brockmann and Kuthe [21] created a model that looks for the same day of the week of one year ago, and uses that day as a prediction. Although it does not perform disappointingly, it can be difficult and time consuming to devise a distance metric and to configure the number and weighing of neighbours. Most importantly, it is not able to discover unknown relations or to generalise properly.

Self-organising maps such as Kohonen neural networks [73] have found their use mostly in clustering data. The main application of these unsupervised learning methods to electricity load forecasting has been to discover which days of the week have similar electricity load patterns, which allows a reduction of the number of input variables. In the dominant results two clusters are found, one containing weekdays and one containing weekends. Other separations made are found to be days close to the weekend, midweek, holidays and Sundays, and more [8, 21, 91, 97]. Other research claims that every day of the week can have a unique load pattern [42], so every day is treated separately [33, 71]. Besides those that cluster on the basis of day of the week, algorithms that cluster on all input variables are employed as well [26, 27, 65].

2.3.2 Artificial Neural Networks

Artificial neural networks, commonly referred to as *neural nets*, are computational models which consist of a large number of densely connected simple processing units. The simple processing units are often called artificial neurons or nodes. The connections between simple processing units are also often called links. Each link is associated with a weight which is often called synapse strength of the link. Neural networks are typically arranged in a number of layers. In the case of feed forward networks, there are connections only between adjacent layers. Figure 2.2 shows a multi-layer feed-forward network which consists of an input layer, one hidden layer, and an output layer.

The application of artificial neural networks [ANNs; 58, 92, 108, 130] to electricity demand modelling has been around for more than a decade [42]. When neural network solutions are used for function approximation, the multi-layer perceptron (MLP) network seems to be a very attractive choice. This is because it has been theoretically proved that MLP can well approximate continuous func-

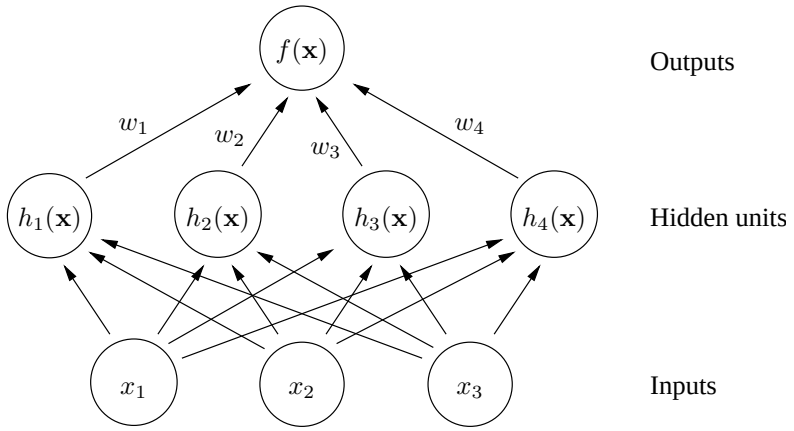

FIGURE 2.2: A feed-forward neural network with inputs in $\mathbb{R}^3$ and 4 hidden units.

tions when enough neurons are used. MLP networks are implemented in almost all commercial neural network simulators and many of the shareware neural network simulators. It is probably due to this wide availability and use that it is one of the most referred to types of ANN in the electricity demand modelling literature [20, 23, 24, 35, 40, 42, 47, 59, 71, 79, 95, 110, 114, 138], and that it is still the most frequently used technique.

Besides the traditional multilayer perceptron, many other types of neural networks have been utilised in this field, e.g., simple recurrent networks [38], grey neural networks [61], functional networks [29, 30], and adaptive logic networks [45]. Radial basis function networks are an advanced variant of artificial neural networks and have excellent nonlinear approximation capabilities. They have been successfully applied to a large diversity of problems, including chaotic time series modelling [28]. Radial basis function (RBF) networks have traditionally been associated with radial basis functions in a single layer network such as shown in figure 2.2. In the input layer, each element of the input vector $\mathbf{x}$ is fully connected to all inputs of the hidden layer neurons. RBF network topology is determined by the number of hidden units. In the hidden layer, the hidden unit activation function $h_i(\mathbf{x})$ is a radial basis function. The output layer combines the outputs of the functions in the hidden layer, to form

$$y(\mathbf{x}) = w_0 + \sum_{i=0}^{N} w_i h_i(\mathbf{x}). \tag{2.5}$$

A radial basis function neural network which has its radial bases at locations of patterns presented in the training data equals a kernel machine using a Gaussian

kernel, discussed in subsection 2.3.5.

Training a Neural Network

Training of a neural network is the procedure of finding the proper weights of the network in such a way that the inputs correspond with the outputs. The error back-propagation learning algorithm is a form of supervised learning used to train mainly feed-forward neural networks. In outline, the algorithm is as follows.

- **Initialisation.** The weights of the network are initialised to small random values. First the weights are initialised, usually with random values often with a dispersal around 0.
- **Forward pass.** Inputs of each training pattern are presented to the network. The outputs are computed using the inputs and the current weights of the network. Certain statistics are kept from this computation, and used in the next phase. The target outputs of each training pattern are compared with the actual activation levels of the output units, the difference between the two is termed the error. Training may be pattern-by-pattern or epoch-by-epoch. With pattern-by-pattern training, the pattern error is provided directly to the backward pass. With epoch-by-epoch training, the pattern errors are summed across all training patterns, and the total error is provided to the backward pass.
- **Backward pass**. In this phase, the weights of the neural network are updated.

Often, a separate learn data set and test data set are used to decide when to stop learning. This is exemplified by the progress of a learn set error and test set error, which is depicted in figure 2.3. By over-learning (or over-specialisation) it can happen that the error decreases on the learning data while it increases on the test data. This procedure is repeated until some stopping criterion is reached. Figure 2.3 illustrates a good moment to stop training, the best point to stop the training process is marked by the dotted line. Often a large amount of iterations (epochs) are needed for each set of patterns. Learning continues until the error is small enough, or even better yet, until the best results are attained using a separate set of test patterns.

There are many variations on the above mentioned back-propagation learning, that sometimes are able to achieve better results. Well-known variations are Quick-prop [46], and RPROP [106].

As for radial basis function networks, parameters are established by minimising a cost function

$$\min \sum C(y_i, y(\mathbf{x}_i))$$

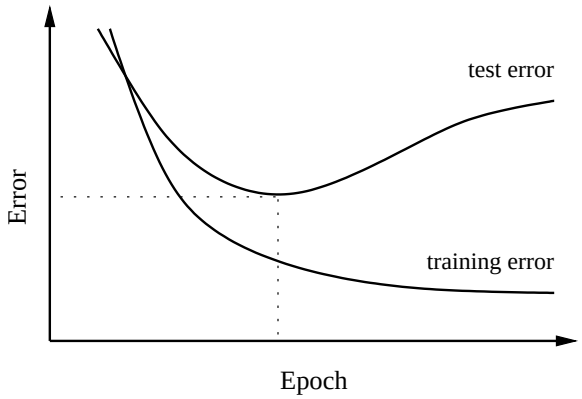

FIGURE 2.3: Typical error developments of back-propagation training [58].

which is typically the sum of the squares of the residuals. As with MLP artificial neural networks, RBF networks can be trained by a variety of supervised learning algorithms. In the initial approaches, all data samples are assigned to the hidden layer to act like a centroid. In later approaches, the number of hidden units is reduced by the use of clustering algorithms such as k-median [17], or by stochastic choice [63]. Other algorithms are orthogonal least squares [115] and gradient descent [68].

One of the drawbacks of neural networks is that they deliver an arcane web of interconnected neurons, and as such are not easily interpretable by humans [5]. This sparked separate research into the inner workings of neural networks applied to electricity demand forecasting [19]. Another problem can be the training times on large data sets. Despite these drawbacks, neural networks can give good all-round performance and have been applied successfully to load forecasting. The mathematical theories necessary to guarantee the performance of applied neural networks are still under development.

2.3.3 Fuzzy Inference Systems

During an international meeting of fuzzy researchers in Tokyo in 1987, Takeshi Yamakawa demonstrated the use of fuzzy control in an *inverted pendulum* experiment, which is a classical control problem in which a vehicle tries to keep a pole, mounted on its top with a hinge, upright by moving back and forth. Observers were impressed by this demonstration, as well as by later experiments by Yamakawa in which he mounted a live mouse on top of the pendulum and the system remained stable. Since then, a wide range of other applications has been investigated or implemented; character and handwriting recognition, opti-

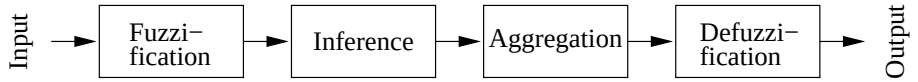

FIGURE 2.4: A schematic view of a fuzzy logic controller.

cal fuzzy systems, voice-controlled unmanned vehicles, control flow of powders in manufacturing film, elevator systems, et cetera.

Zadeh [136] devised fuzzy sets, an extension to Boole's set theory [10], to enable the assignment of an intermediate degree of membership to a subset. In the classical Boolean set theory a subset $B \subset S$ is defined by $\forall x \in S$: $x \mapsto \{0, 1\}$, where $x \mapsto 1$ means $x \in B$ and $x \mapsto 0$ means $x \notin B$. A fuzzy subset $F \subset S$ is represented by $\forall x \in S$: $x \mapsto [0, 1]$ with the meaning of 0 and 1 unchanged, but values in-between representing intermediate degrees of membership. This kind of mapping is usually described by the *membership function* of F, commonly denoted by $\mu_F(x)$, where μ is a *truth value*.

A decade after Zadeh's introduction, Mamdani and Assilian [89] devised the *fuzzy logic controller* (FLC), which differs from a classical proportional-integral-derivative (PID) controller [93] in that it not only can be programmed by linguistic statements, but also in that it performs better on non-differentiable problem domains. Figure 2.4 gives a graphical representation of the processes in a FLC, which I will discuss from left to right. In the fuzzification stage, all inputs are mapped to truth values using membership functions. Given mappings of input variables into membership functions and truth values, a fuzzy logic controller has to determine the output based on a collection of logic rules in the form of `IF-THEN` statements. In this process, inference calculates the outcome of each single rule. If multiple rules fire on the same consequent linguistic variables, the aggregation operator defines the combinatorial result. The final defuzzification stage maps truth values to output values.

With respect to modelling electricity load, the classical fuzzy inference model is applied [82, 98, 129], although the more sophisticated neuro-fuzzy models, which are combinations of a neural network and a fuzzy inference system, are applied as well [1, 21, 62]. Most of these methods are used as elements of hybrid models, since they can not generalise by themselves.

Although the FLCs are appealing because of the linguistic statements, these are also their drawback, e.g., linguistic statements have been inferred recurrently: they become lengthier if the outcome is less crisp. A vast amount of linguistic statements is difficult to be interpreted by humans [11].

2.3.4 Evolutionary Computing

Holland began his work on genetic algorithms in the early 1960s. Followed by

Holland [60]'s publication, the interest in algorithms inspired by Darwin's theory of *natural selection* [39] flourished. As happens in many fields of science, the container term *evolutionary computing* now encompasses many sub fields, such as genetic algorithms and Koza's [76] genetic programming. These algorithms have been successfully applied to a large set of problems. Chen et al. [34] used an evolutionary algorithm to build a combined forecasting method from several other forecasting techniques. Bhattacharya et al. [7] employed a linear genetic programming approach to model electricity-load patterns.

Although an interesting class of methods, it has an important drawback: any evolutionary computing method is only as strong as its underlying model. The computational costs also remain high, often a distributed computing approach is needed to address the scalability issues [12].

2.3.5 Kernel Machines

Vapnik's optimal margin classifier [18], which later became known as the *support vector machine* (SVM), has drawn a considerable amount of attention [22, 37, 112]. SVMs belong to the family of *kernel machines*. They are closely related to artificial neural networks, in fact, using a kernel machine with a sigmoid kernel function is equivalent to a two-layer, feed-forward neural network. An arbitrary kernel function can be used, for polynomial, radial basis function and multi-layer perceptron functions in which the weights of the network are mostly found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, unconstrained minimisation problem as in standard neural network training.

Kernel machines have in common that they combine statistical learning theory to optimise generalisation [123–125], mathematical programming to find solutions efficiently, and the *kernel trick* to handle nonlinearity [3]. In case of regression they use the fact that observational data can, under certain conditions, be represented by a linear combination of *kernel functions* k [41, 128]

$$f(\mathbf{x}) = w_0 + \sum_{n=1}^{N} w_n k(\mathbf{x}_n, \mathbf{x}). \tag{2.6}$$

One can, but often does not have to, deliberately design the similarity of points in the state space by altering this kernel function. When considering Gaussian (which is a radial basis function) kernel functions

$$k(\mathbf{u}, \mathbf{v}) = \exp\left(-\tfrac{1}{2}\sigma^{-2} \left\|\mathbf{u} - \mathbf{v}\right\|_2^2\right), \tag{2.7}$$

kernel machines can be regarded as a topology adaptive approach to radial basis function networks (as discussed in subsection 2.3.2), with the locations of the radial bases restricted to the set of inputs.

Also derivative Bayesian methods have emerged, which include the relevance vector machine [119, 120], Figueiredo's method [48], and the kernelised Lasso [109]. These methods tend to produce more accurate and concise results than the support vector machine. However, they are computationally more costly than the support vector machine.

Perhaps because kernel machines are relatively new, they are rarely found in the load forecasting literature. Support vector machines were used in two entries of the Eunite competition including the winning one [33, 107].

2.4 Quality Criteria

Quality criteria are an important topic, because one can go as far as saying that any model can be better than any other model by merely using an other quality criterion. I consider three properties, successively accuracy, consistency, and robustness.

Accuracy of a model is measured by the size of the average error resulting from the model. This average can be determined in many different ways, and is mostly obtained by a transformation on one of the vector p-norms of the difference between the predicted outputs $\tilde{\mathbf{y}}$ and the actual outputs $\mathbf{y}$, given by $\|\mathbf{y}-\tilde{\mathbf{y}}\|_p = (\sum_i |y_i - \tilde{y}_i|^p)^{1/p}$. In this category, the vector p-norms most commonly used are the root-mean-square of the error

$$\mathrm{RMSE}(\mathbf{y},\tilde{\mathbf{y}}) = \sqrt{N^{-1}\,\|\mathbf{y}-\tilde{\mathbf{y}}\|_2^2},$$

the mean absolute error $\mathrm{MAE}(\mathbf{y},\tilde{\mathbf{y}}) = N^{-1}\,\|\mathbf{y}-\tilde{\mathbf{y}}\|_1$, or the maximum absolute error $\mathrm{MAX}(\mathbf{y},\tilde{\mathbf{y}}) = \|\mathbf{y}-\tilde{\mathbf{y}}\|_\infty$. In the field of electricity-load forecasting, the mean absolute percentage error

$$\mathrm{MAPE}(\mathbf{y},\tilde{\mathbf{y}}) = 100\,N^{-1}\sum_{i=1}^{N} |y_i - \tilde{y}_i|/y_i \tag{2.8}$$

is a frequently used benchmark. The MAPE error benchmark has been used by the great building energy predictor shootout [56] and the Eunite competition. The European Network of Excellence on Intelligent Technologies for Smart Adaptive Systems organised an electricity-load forecasting competition in 2001, where the MAX and MAPE error benchmarks were used.

Consistency is about the higher order moments of the error distribution. If one estimates a probability density function of the error, ideally it should have its mode located at the mean, and the distribution should tend towards a normal distribution.

Robustness of a model addresses the performance in uncharted waters. This can be measured by using cross-validation. In this approach, data are (repeatedly) split up in a *training set* and *test set*. The computational learning method

trains on the training set and consequently predicts for samples in the test set. Error measures and distributions of the test set should be similar to that of the training set.

Common benchmarks in determining the quality of the short-term wind-power prediction are the persistence model and the mean-production model. When using persistence, one takes the previous measured value(s) as the prediction for the next value(s). The model to beat is the persistence model [54].

3

Smooth Bayesian Kernel Machines

Sparked by the introduction of the support vector machine [18], kernel machines have become a popular tool to model measured data. As a result of the combination of several disciplines, these machines have in common that they combine the kernel trick [3] and the principle of parsimony [112, 124]. The latter is brought forth by obtaining a sparse set that utilises a small subset of the data to represent a function.

A great deal of research has been done to find the optimum balance between quality of fit and the size of the support vector set. A common approach is to put a constraint or penalty on the model parameters, as is done by, e.g., the 0-norm support vector machine [131], 1-norm support vector machine [137], or 2-norm support vector machine [116]. The relevance vector machine [120] uses automatic relevance determination [86] to find that balance. Other types of traditional regression methods, such as the Lasso [118], have also been applied successfully in combination with kernels [109].

Quite some evidence can be found for the proposition that sparseness, in some sense, equals smoothness [see, e.g., 49, subsection 2.2]. However, imposing constraints on the magnitude of the weights alone does not necessarily result in representations which are smooth in the output space (see figure 3.3). As yet, no special attention is paid to the smoothness of the function, i.e., that it should have continuous derivatives up to some order[1].

Ramsay and Silverman [105] promote smoothness by a roughness penalty: a

[1] The smooth support vector machine [78] entails a formulation of the quadratic program of the support vector machine.

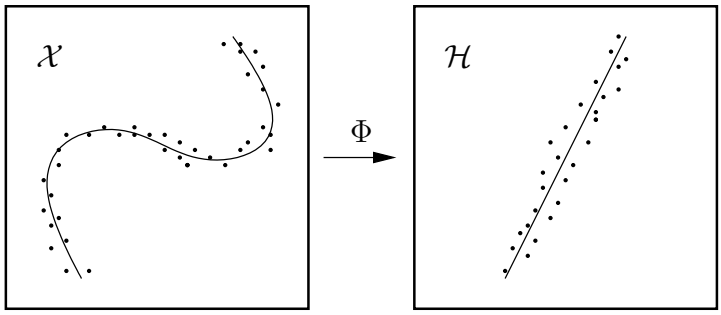

FIGURE 3.1: A mapping Φ from input space $\mathcal{X}$ to feature space $\mathcal{H}$.

penalty on the magnitude of one or more derivatives of the function. Advertised advantages of this penalty are its flexibility, i.e., the freedom to design the penalty to fit the problem at hand more adequately, and that it results in high quality models.

The remainder of this chapter is organised as follows. In section 3.1, the concept of smooth functional representations is laid down from the view of kernel machines. I introduce derivatives of kernel machines, derivatives of several kernel functions, and kernel roughness penalties. It is noted that through penalised regularisation, roughness penalties do lead to smoothness, but do not lead to sparseness. To find a balance between smoothness and sparseness, I have devised the smooth relevance vector machine [13], discussed in section 3.2. A brief introduction to Bayesian model inference is given, followed by a novel prior for sparseness and smoothness, its related model, and its update equations. Section 3.3 shows comparative experimental results of the smooth relevance vector machine on synthetic data.

3.1 Smooth Functional Representations

Kernel machines exploit the *kernel trick*, the idea of mapping data to a high-dimensional feature space where some linear algorithm is applied that works exclusively with inner products. Assume we have some mapping Φ from an input space $\mathcal{X}$ to a feature space $\mathcal{H}$, then a *kernel function* (or kernel)

$$k(\mathbf{u}, \mathbf{v}) = \langle \Phi(\mathbf{u}), \Phi(\mathbf{v}) \rangle \tag{3.1}$$

is used, under certain conditions, to define the inner product in feature space $\mathcal{H}$. Figure 3.1 illustrates the basic idea of the kernel trick: to linearise in feature space $\mathcal{H}$ is equal to a nonlinear estimate in input space $\mathcal{X}$. Some commonly used kernel functions [53] are written down in table 3.1.

TABLE 3.1: Some common kernel functions and their properties.

Type	$k(\mathbf{u}, \mathbf{v})$	$\dim \mathcal{H}$
Linear	$\langle \mathbf{u}, \mathbf{v} \rangle$	1
Polynomial	$(\gamma \langle \mathbf{u}, \mathbf{v} \rangle + b)^d$	d
Exponential	$\exp(-\gamma \|\mathbf{u} - \mathbf{v}\|)$	∞
Gaussian	$\exp\left(-\frac{1}{2}\sigma^{-2} \|\mathbf{u} - \mathbf{v}\|^2\right)$	∞
Sigmoid	$\tanh(\gamma \langle \mathbf{u}, \mathbf{v} \rangle + b)$	∞

In supervised learning, we consider a data set $\mathcal{D} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ containing N input-output pairs $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, with $\mathcal{X}$ typically containing multidimensional vectors in $\mathbb{R}^M$, and $\mathcal{Y}$ representing either classes in case of classification, or scalars in $\mathbb{R}$ in case of regression. I will consider the case of regression. Wahba [128] shows that we can represent our data $\mathcal{D}$ using a linear model of the form

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{N} w_i k(\mathbf{x}, \mathbf{x}_i) \tag{3.2}$$

with bias w_0, parameters $w_1, \ldots, w_N$, and a kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ as defined in (3.1).

In the field of functional data analysis, smoothness is favoured explicitly by applying a roughness penalty [55], a penalty on the degree of curvature of one or more derivatives of $y(\mathbf{x})$. Fortunately, as $y(\mathbf{x})$ in (3.2) is linear in terms of kernels $k(\mathbf{x}, \mathbf{x}_i)$, its derivatives are defined by

$$D^n y(\mathbf{x}) = \sum_{i=1}^{N} w_i D^n k(\mathbf{x}, \mathbf{x}_i), \tag{3.3}$$

where D^n is a derivative operator, which will be further developed below, together with derivatives of the Gaussian and polynomial kernel functions.

3.1.1 Derivative Kernels

To establish derivative kernels, let us rely on some basic vector calculus. For a fixed $\mathbf{x}_i$, a kernel function $k(\mathbf{x}, \mathbf{x}_i)$ maps $\mathbf{x}$ to a scalar field, i.e., it is a function such that $k : \mathbb{R}^n \mapsto \mathbb{R}$. If the kernel function $k(\mathbf{x}, \mathbf{x}_i)$ is differentiable, the partial derivative of a scalar field is defined in an unambiguous way. This results in a vector field called the gradient, denoted by $\nabla : \mathbb{R}^n \mapsto \mathbb{R}^n$, and obtained by $\nabla = \partial / \partial \mathbf{x}$. To determine the derivative of that resulting vector field, we will use the divergence operator, which involves an inner product and results in a scalar field again, $\nabla^{\mathrm{T}} \nabla : \mathbb{R}^n \mapsto \mathbb{R}$. In physical terms, the divergence of a vector field is

the extent to which the vector-field flow behaves like a source or a sink at a given point. So, in short, we use the gradient and divergence alternately, following a recurrent definition

$$D^n = \begin{cases} \nabla D^{n-1} & \text{for } n \text{ odd}, \\ \nabla^{\mathrm{T}} D^{n-1} & \text{for } n \text{ even}. \end{cases}$$

Let $D^0 k(\mathbf{x}, \mathbf{x}_i) = k(\mathbf{x}, \mathbf{x}_i)$, then the dimension of the n-th order derivative of a kernel $D^n k(\mathbf{x}, \mathbf{x}_i)$ is either 1 for n is even, or M for n being an odd number, where M is the dimension of underlying vectors $\mathbf{x}$ and $\mathbf{x}_i$. Because a kernel maps to a scalar field, derivative kernels exist for n even or for $M = 1$ (or both).

Furthermore, if we use the following definition for the power operator for vectors

$$\mathbf{x}^n = \begin{cases} (\mathbf{x}^{\mathrm{T}}\mathbf{x})^{n/2} & \text{for } n \text{ even}, \\ \mathbf{x}(\mathbf{x}^{\mathrm{T}}\mathbf{x})^{(n-1)/2} & \text{for } n \text{ odd}, \end{cases} \tag{3.4}$$

we can simply substitute scalars for vectors in derivative calculus, with the product operator assumed to be the inner product. The derivatives of the commonly used Gaussian kernel and derivatives of the polynomial kernel are highlighted next.

Gaussian Derivative Kernel

Derivatives of the Gaussian kernel $k(\mathbf{u}, \mathbf{v}) = \exp(-\frac{1}{2}\sigma^{-2} \left\|\mathbf{u} - \mathbf{v}\right\|_2^2)$ are identified by Rodrigues' formula for Hermite polynomials

$$H_n(x) = (-1)^n \exp(x^2) D^n \exp(-x^2). \tag{3.5}$$

By letting the power operator for vectors be defined by (3.4), substituting the term $(\sqrt{2}\sigma)^{-1}(\mathbf{u} - \mathbf{v})$ for x in (3.5), and keeping track of additional terms of $(\sqrt{2}\sigma)^{-1}$, we arrive at the convenient compact form of the derivatives of Gaussian kernels

$$D^n k(\mathbf{u}, \mathbf{v}) = (-\sqrt{2}\sigma)^{-n} H_n((\sqrt{2}\sigma)^{-1}(\mathbf{u} - \mathbf{v})) \exp(-\tfrac{1}{2}\sigma^{-2} \left\|\mathbf{u} - \mathbf{v}\right\|_2^2).$$

Figure 3.2 shows a Gaussian kernel and its first five order of derivatives. They are located at 0 and shown for the domain $[-3, 3]$.

Polynomial Derivative Kernel

Derivatives of the polynomial kernel $k(\mathbf{u}, \mathbf{v}) = \left(\gamma \mathbf{u}^{\mathrm{T}}\mathbf{v} + \lambda\right)^d$ are found to be

$$D^n k(\mathbf{u}, \mathbf{v}) = \frac{d!}{(d-n)!} \gamma^n \mathbf{v}_i^n \left(\gamma \mathbf{u}^{\mathrm{T}}\mathbf{v} + \lambda\right)^{d-n}$$

which is valid in this instantiation as long as $d \geq n$.

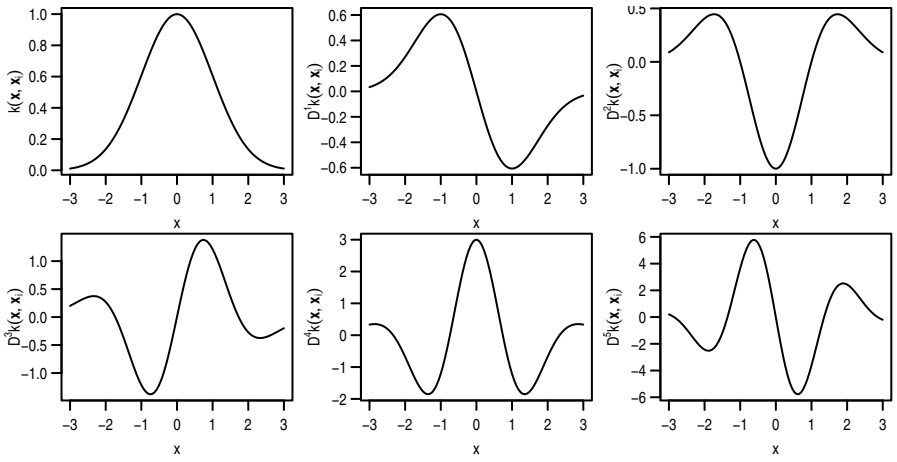

FIGURE 3.2: Gaussian kernel (parametrised by $\sigma = 1$, top left), and its first five orders of derivatives.

3.1.2 Kernel Roughness Penalties

An applied roughness penalty can be related to the assumption that the regression surface is made of some kind of elastic sheet. It is observed in nature that tensions in any elastic sheet tend to be distributed to an overall minimum value. The regression surface can be assumed to be such an elastic sheet. In this case, the tensions in this surface will have to be kept at an overall minimum, or limited to a maximum.

Let a kernel matrix $\mathbf{K}$ be a matrix with entries $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ for inputs $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \mathcal{X}$, and let a design matrix $\mathbf{H} = [\mathbf{1}\,\mathbf{K}]$ be a matrix that consists of the combination of $[1, 1, \ldots, 1]^{\mathrm{T}}$ and a kernel matrix,

$$\mathbf{H} = \begin{bmatrix} 1 & k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \ldots & k(\mathbf{x}_1, \mathbf{x}_N) \\ 1 & k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \ldots & k(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & k(\mathbf{x}_N, \mathbf{x}_1) & k(\mathbf{x}_N, \mathbf{x}_2) & \ldots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}.$$

For example, in a regularisation setting, applying a roughness penalty can be accomplished by penalising the summed curvature of the second order derivatives

$$\min \|\mathbf{y} - \mathbf{H}\mathbf{w}\|_2^2 + \lambda \left\|D^2\mathbf{H}\mathbf{w}\right\|_2^2 \tag{3.6}$$

with $D^2\mathbf{H} = [\mathbf{0}\, D^2\mathbf{K}]$ being a design matrix of a second order derivative kernel with $D^2 K_{ij} = D^2 k(\mathbf{x}_i, \mathbf{x}_j)$. For $\lambda = 0$, this imposes an ordinary least squares, and for $\lambda \to \infty$, the result will approximate a straight line. Besides a penalty on the

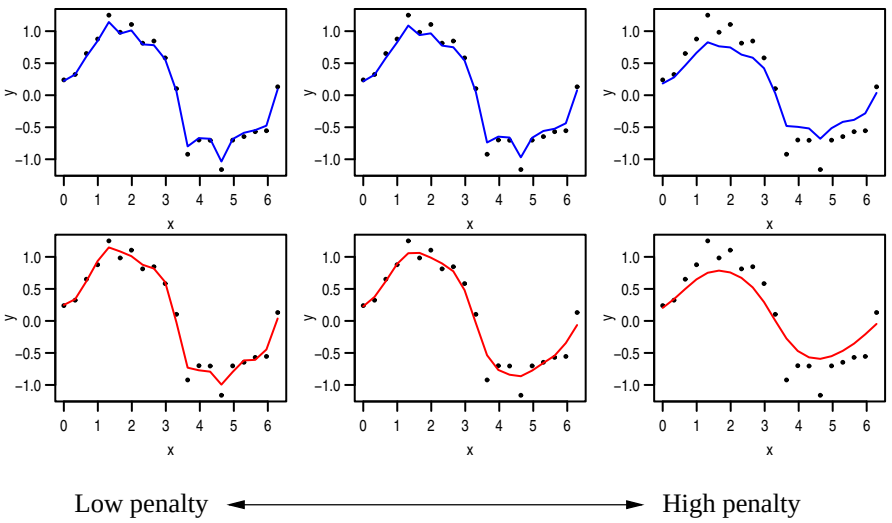

FIGURE 3.3: Ridge penalised regression (top row) and roughness penalised regression (bottom row).

second order derivative as shown in (3.6), more generally, a linear differential operator

$$Ly(\mathbf{x}) = \sum_n c_n D^n y(\mathbf{x}) \tag{3.7}$$

can be defined [104], and used to form a weighted sum of penalties on $D^n y(\mathbf{x})$ from (3.3). If applied in penalised regularisation, L takes the shape

$$\min \|\mathbf{y} - \mathbf{H}\mathbf{w}\|_2^2 + \lambda \|\mathbf{L}\mathbf{w}\|_2^2 \tag{3.8}$$

with $\mathbf{L}$ a roughness penalty matrix, formed by the sum of several derivative design matrices $\mathbf{L} = \sum_n c_n D^n \mathbf{H}$. The optimisation problem in (3.8) is solved by $\mathbf{w} = (\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda \mathbf{L}^{\mathrm{T}}\mathbf{L})^{-1}\mathbf{y}^{\mathrm{T}}\mathbf{H}$.

Figure 3.3 illustrates the difference in results between ridge penalised regression, which can be obtained by solving for

$$\min \|\mathbf{y} - \mathbf{H}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 , \tag{3.9}$$

and roughness penalised regression, both on a noisy sine function $\mathcal{N}(\sin(x), 0.2)$ on the domain $[0, 2\pi]$. The amount of penalty (controlled by changing parameters λ in (3.8) and (3.9)) is increased from left to right. In this case, a Gaussian kernel has been used with $\sigma = 0.2$, and the 4th order Gaussian derivative kernel has been penalised for roughness. The roughness penalty structure deviates

from the usual kernel machines in the sense that it uses a projection of the feature space on a lower-dimensional (but often still high-dimensional) space to determine the amount of penalty placed on the model parameters.

Despite reasonable conditions of matrices $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ and $\mathbf{L}^{\mathrm{T}}\mathbf{L}$, this system of equations is singular in case of duplicate inputs. In the field of functional data analysis, this is addressed by pre-processing, such as starting with the dominant frequencies [105]. Also, the choice of parameters λ is done manually by methods such as cross-validation.

Although the resulting functions are smooth, regularisation as in (3.8) does not promote sparseness. My aim is to have a representation that is both smooth and sparse, and in addition to have an automatic selection of the value of parameter λ.

3.2 Smooth Relevance Vector Machine

In essence, the Bayes approach provides a probabilistic rule explaining how you should change your existing beliefs in the light of new evidence. In other words, it allows us to combine new data with our existing (prior) knowledge or expertise.

Assume we formulate a parametrised model for the data. If these model parameters are subject to existing beliefs also, we speak of a *hierarchical* Bayesian model [25]. We could state beliefs of these beliefs and beliefs thereof et cetera, but the hierarchy must stop at some point: we will use the observed data to estimate the final stage parameters. MacKay [85, 86, 87, 88] showed that a two-stage hierarchical Bayesian model can embody *Occam's razor*: certain prior ideas on the model parameters enforce the model to be fitted in a concise way. Stated differently, with this we can find the least complex explanation for the evidence, the observed data.

Perhaps even more important in practice is that non-hierarchical models are usually inappropriate: with a few parameters, they cannot fit large datasets accurately, whereas with many parameters, they tend to over-fit such data. This over-fitting takes place in the sense that the producing models fit well to the existing data but lead to inferior predictions for unseen data [52]. To put it differently, they lack generalisation.

In a Bayes approach we have to specify a model for the observed data $\mathcal{D} = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$ given an unknown parameter vector $\mathbf{w}$; this is specified in (3.2). Let us further assume that parameter vector $\mathbf{w}$ is a random quantity as well, having a *prior* distribution $p(\mathbf{w} \mid \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a vector of *hyper-parameters* (or second-stage parameters). Normal Bayesian *inference* concerning $\mathbf{w}$ is then based on its *posterior* distribution

$$p(\mathbf{w} \mid \mathbf{y}, \boldsymbol{\theta}) = \frac{p(\mathbf{w}, \mathbf{y}, \boldsymbol{\theta})}{p(\mathbf{y}, \boldsymbol{\theta})} = \frac{p(\mathbf{y}, \mathbf{w} \mid \boldsymbol{\theta})}{p(\mathbf{y} \mid \boldsymbol{\theta})} \tag{3.10}$$

where $p(\mathbf{y} \mid \boldsymbol{\theta})$ is the *marginal distribution* of $\mathbf{y}$

$$p(\mathbf{y} \mid \boldsymbol{\theta}) = \int p(\mathbf{y} \mid \mathbf{w}) p(\mathbf{w} \mid \boldsymbol{\theta}) \, d\mathbf{w}.$$

If the hyper-parameter vector $\boldsymbol{\theta}$ is unknown, the *fully* Bayesian approach would adopt a second-stage prior distribution, a *hyper-prior* distribution $p(\boldsymbol{\theta})$ which allows inference concerning $\boldsymbol{\theta}$ to be based on

$$p(\boldsymbol{\theta} \mid \mathbf{y}) = \frac{\int p(\mathbf{y} \mid \mathbf{w}) p(\mathbf{w} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) \, d\boldsymbol{\theta}}{\iint p(\mathbf{y} \mid \boldsymbol{\eta}) p(\boldsymbol{\eta} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) \, d\boldsymbol{\eta} \, d\boldsymbol{\theta}} = \int p(\mathbf{w} \mid \mathbf{y}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathbf{y}) \, d\boldsymbol{\theta}. \tag{3.11}$$

If the hyper-prior distribution $p(\boldsymbol{\theta})$ cannot be specified, it is possible to follow an *empirical* Bayes approach. The name empirical Bayes stems from the fact that we are using the data to estimate hyper-parameter vector $\boldsymbol{\theta}$ by $\hat{\boldsymbol{\theta}}$. The empirical Bayes approach essentially replaces the integration in the right-hand side of (3.11) by a maximisation, also known as the marginal maximum likelihood estimation. In that case, inference is based on the estimated posterior distribution $p(\mathbf{w} \mid \mathbf{y}, \hat{\boldsymbol{\theta}})$.

3.2.1 A Novel Prior for Smoothness and Sparseness

To fit (3.2) in a Bayesian framework, we will assume that our observational data $\mathcal{D} = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$ are corrupted by independently and identically distributed (iid) noise generated by a Gaussian probability function (denoted by $\mathcal{N}$)

$$y_i \overset{\text{iid}}{\sim} \mathcal{N}\left(y(\mathbf{x}_i), \sigma^2\right).$$

Although this can be a dangerous assumption, it is often a good approximation due to a surprising result known as the central limit theorem. This theorem states that the mean of any set of random variables with any distribution having a finite mean and variance tends to the Gaussian distribution.

I combine the idea of regularisation by a roughness penalty with a Bayesian sparseness inducing prior. As described above, I presume that the outputs are corrupted by Gaussian noise,

$$p(\mathbf{y} \mid \mathbf{w}, \sigma^2) = \mathcal{N}(\mathbf{y} \mid \mathbf{H}\mathbf{w}, \sigma^2 \mathbf{I}). \tag{3.12}$$

To promote both smoothness and sparseness, I propose a zero-mean Gaussian prior over the weights, and a covariance matrix consisting of two terms

$$p(\mathbf{w} \mid \boldsymbol{\alpha}, \lambda) = \mathcal{N}(\mathbf{w} \mid \mathbf{0}, (\lambda \mathbf{L}^{\mathrm{T}} \mathbf{L} + \mathbf{A})^{-1}). \tag{3.13}$$

The first term $\lambda \mathbf{L}^{\mathrm{T}} \mathbf{L}$ is the smoothness promoting part, formed by a roughness penalty matrix defined underneath (3.8). The second term is a diagonal matrix

TABLE 3.2: The prior used by the RVM and the SRVM.

Method	Prior
RVM [120]	$p(\mathbf{w} \mid \boldsymbol{\alpha}) = \mathcal{N}(\mathbf{w} \mid \mathbf{0}, \mathbf{A}^{-1})$
SRVM [this thesis]	$p(\mathbf{w} \mid \boldsymbol{\alpha}, \lambda) = \mathcal{N}(\mathbf{w} \mid \mathbf{0}, (\lambda \mathbf{L}^{\mathrm{T}}\mathbf{L} + \mathbf{A})^{-1})$

$\mathbf{A} = \mathrm{diag}(\alpha_0, \ldots, \alpha_{\mathrm{N}})$ containing one hyper-parameter α_i per weight w_i, as in automatic relevance determination (ARD) [88], as applied in the relevance vector machine (RVM) [120]. Table 3.2 shows the difference between the priors of the RVM and my SRVM.

3.2.2 Posterior Distributions

To arrive at the posterior distributions, we could integrate out the weights,

$$\int p(\mathbf{y} \mid \mathbf{w}, \sigma)\, p(\mathbf{w} \mid \boldsymbol{\alpha}, \lambda)\, d\mathbf{w}$$

but there is another way to get there, that is, by exploiting Bayes' rule

$$p(\mathbf{y} \mid \mathbf{w}, \sigma^2)\, p(\mathbf{w} \mid \boldsymbol{\alpha}, \lambda) = p(\mathbf{y} \mid \boldsymbol{\alpha}, \sigma^2)\, p(\mathbf{w} \mid \boldsymbol{\alpha}, \sigma^2). \tag{3.14}$$

Substitution of (3.12) and (3.13) into the left-hand side of (3.14) results in a product of two Gaussians

$$\mathcal{N}(\mathbf{y} \mid \mathbf{H}\mathbf{w}, \sigma^2\mathbf{I})\, \mathcal{N}(\mathbf{w} \mid \mathbf{0}, (\lambda \mathbf{L}^{\mathrm{T}}\mathbf{L} + \mathbf{A})^{-1}). \tag{3.15}$$

Let us consider the terms in the exponent of the product of the Gaussians in (3.15),

$$\sigma^{-2}\mathbf{y}^{\mathrm{T}}\mathbf{y} - 2\mathbf{w}^{\mathrm{T}}\sigma^{-2}\mathbf{H}^{T}\mathbf{y} + \mathbf{w}^{\mathrm{T}}\left(\sigma^{-2}\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda\mathbf{L}^{\mathrm{T}}\mathbf{L} + \mathbf{A}\right)\mathbf{w}.$$

This can be rewritten using a matrix multiplication identity that is given by $(\mathbf{a} - \mathbf{B}\mathbf{c})^{\mathrm{T}}\mathbf{B}^{-1}(\mathbf{a} - \mathbf{B}\mathbf{c}) = \mathbf{a}^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{a} - 2\mathbf{a}^{\mathrm{T}}\mathbf{c} + \mathbf{c}^{\mathrm{T}}\mathbf{B}\mathbf{c}$, by defining

$$\begin{aligned} \boldsymbol{\Sigma} &= \left(\sigma^{-2}\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda\mathbf{L}^{\mathrm{T}}\mathbf{L} + \mathbf{A}\right)^{-1}, \text{ and} \\ \boldsymbol{\mu} &= \boldsymbol{\Sigma}\sigma^{-2}\mathbf{H}^{\mathrm{T}}\mathbf{y}, \end{aligned}$$

substituting, rewriting, and grouping, to end up with

$$\begin{gathered} (\mathbf{w} - \boldsymbol{\mu})^{\mathrm{T}}\, \boldsymbol{\Sigma}^{-1}\, (\mathbf{w} - \boldsymbol{\mu}) + \\ \mathbf{y}^{\mathrm{T}}(\sigma^{-2}\mathbf{I} - \sigma^{-2}\mathbf{H}(\sigma^{-2}\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda\mathbf{L}^{\mathrm{T}}\mathbf{L} + \mathbf{A})^{-1}\mathbf{H}^{\mathrm{T}}\sigma^{-2})\mathbf{y}) \end{gathered} .$$

The latter part can be rewritten using the matrix inversion lemma (also known as the Sherman-Morrison-Woodbury identity) given by

$$(\mathbf{A}+\mathbf{X}\mathbf{B}\mathbf{X}^{\mathrm{T}})^{-1}=\mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{X}(\mathbf{B}^{-1}+\mathbf{X}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{A}^{-1}$$

to obtain

$$(\mathbf{w}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu})+\mathbf{y}^{\mathrm{T}}\left(\sigma^2\mathbf{I}+\mathbf{H}\left(\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}+\mathbf{A}\right)^{-1}\mathbf{H}^{\mathrm{T}}\right)^{-1}\mathbf{y} \qquad (3.16)$$

as the terms in the exponent. We can re-express (3.16) as a product of two Gaussians

$$\mathcal{N}(\mathbf{w}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma})\,\mathcal{N}(\mathbf{y}\,|\,0,\sigma^2\mathbf{I}+\mathbf{H}(\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}+\mathbf{A})^{-1}\mathbf{H}^{\mathrm{T}})$$

which is the right-hand side of (3.14) that we are looking for. The posterior distributions are given by

$$\begin{aligned} p(\mathbf{w}|\mathbf{y},\boldsymbol{\alpha},\lambda,\sigma^2) &= \mathcal{N}(\mathbf{w}|\boldsymbol{\mu},\boldsymbol{\Sigma}),\\ p(\mathbf{y}|\boldsymbol{\alpha},\lambda,\sigma^2) &= \mathcal{N}(\mathbf{y}|\mathbf{0},\sigma^2\mathbf{I}+\mathbf{H}\mathbf{S}\mathbf{H}^{\mathrm{T}}). \end{aligned}$$

with $\boldsymbol{\Sigma}=(\sigma^{-2}\mathbf{H}^{\mathrm{T}}\mathbf{H}+\mathbf{S}^{-1})^{-1}$, $\mathbf{S}=(\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}+\mathbf{A})^{-1}$ and $\boldsymbol{\mu}=\sigma^{-2}\boldsymbol{\Sigma}\mathbf{H}^{\mathrm{T}}\mathbf{y}$.

3.2.3 Obtaining Posterior Modes

To obtain (locally) optimum values for $\boldsymbol{\alpha}$, λ and σ^2, we will follow a type-II maximum likelihood approach similar to that of the RVM [120]. Of course, the normal procedure for finding optima of functions, is to differentiate the function and to find the roots.

We take a Gamma prior over the parameters $\boldsymbol{\alpha}$, σ^2 and λ. Therefore, we maximise

$$\log p(y|\log\boldsymbol{\alpha},\log\beta,\log\lambda)+\sum\log p(\log\alpha_i)+\log p(\log\beta)+\log p(\log\lambda)$$

which gives the following objective function (as in ref. Tipping).

$$\mathcal{L}=-\tfrac{1}{2}\log|\sigma^2\mathbf{I}+\mathbf{H}(\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}+\mathbf{A})^{-1}\mathbf{H}^{\mathrm{T}}|+\mathbf{y}^{\mathrm{T}}(\sigma^2\mathbf{I}+\mathbf{H}(\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}+\mathbf{A})^{-1}\mathbf{H}^{\mathrm{T}})\mathbf{y} \quad (3.17)$$

the first term is rewritten using the matrix inversion lemma for determinants $|\mathbf{A}+\mathbf{X}\mathbf{B}\mathbf{X}^{\mathrm{T}}|=|\mathbf{B}||\mathbf{A}||\mathbf{B}^{-1}+\mathbf{X}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{X}|$ to

$$|\sigma^2\mathbf{I}+\mathbf{H}(\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}+\mathbf{A})^{-1}\mathbf{H}^{\mathrm{T}}|=|(\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}+\mathbf{A})^{-1}||\sigma^2\mathbf{I}||\boldsymbol{\Sigma}^{-1}|$$

When taking the log, use the identity $\log a^{-1}b^{-1}=-\log a-\log b$ to obtain

$$\log|\sigma^2\mathbf{I}+\mathbf{H}(\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}+\mathbf{A})^{-1}\mathbf{H}^{\mathrm{T}}|=-\log|\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}+\mathbf{A}|+N\log\sigma^2-\log|\Sigma|$$

and the data dependent term is rewritten as

$$=\sigma^2\mathbf{y}^{\mathrm{T}}\mathbf{y}-\sigma^2\mathbf{y}^{\mathrm{T}}\mathbf{H}\boldsymbol{\Sigma}\mathbf{H}^{\mathrm{T}}\mathbf{y}\sigma^2$$

$$=\sigma^2\mathbf{y}^{\mathrm{T}}(\mathbf{y}-\mathbf{H}\boldsymbol{\mu})$$

$$=\sigma^2\left\|\mathbf{y}-\mathbf{H}\boldsymbol{\mu}\right\|^2+\boldsymbol{\mu}(\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}+\mathbf{A})\boldsymbol{\mu}.$$

Hyper-parameters

In order to obtain an update rule for the hyper-parameters α_i, an identity of the derivatives of the determinant is used, $\partial \log |X(z)|/\partial z = \mathrm{Tr}(X^{-1}\partial X/\partial z)$. Here, Tr is the trace operator, defined to be the sum of the values on the diagonal of a matrix. Derivatives of (3.17) with respect to $\log \alpha_i$ are

$$\frac{\partial \mathcal{L}}{\partial \log \alpha_i} = -\tfrac{1}{2}(-\alpha_i S_{ii} + \alpha_i \Sigma_{ii} + \alpha_i \mu_i^2) + a - b\alpha_i, \tag{3.18}$$

here the fact is used that values on the diagonal are computable when multiplying with a diagonal matrix. Solving for α_i in (3.18), thereby following MacKay [85], the update rules become

$$\alpha_i^{t+1} = \frac{\alpha_i^t(S_{ii} - \Sigma_{ii})}{\mu_i^2}. \tag{3.19}$$

Note that in the case of $\lambda = 0$, then $S_{ii} = (\alpha_i^{-1})^t$, which makes (3.19) identical to the update equation for the hyper-parameters used by the original RVM [120].

Noise variance

With respect to $\log \sigma^2$ the derivatives of (3.17) are

$$\frac{\partial \mathcal{L}}{\partial \log \sigma^2} = -\tfrac{1}{2}(N - \mathrm{Tr}(\Sigma\sigma^{-2}\mathbf{H}^{\mathrm{T}}\mathbf{H})) - \sigma^2 \left\|\mathbf{y} - \mathbf{H}\mu\right\|^2.$$

This can be written as

$$(\sigma^2)^{t+1} = \frac{\left\|\mathbf{y} - \mathbf{H}\boldsymbol{\mu}\right\|^2}{N - \mathrm{Tr}(\boldsymbol{\Sigma}(\sigma^{-2})^t\mathbf{H}^{\mathrm{T}}\mathbf{H})}. \tag{3.20}$$

which re-estimates the noise variance σ^2.

Smoothness parameter

For the smoothness parameter λ, the derivative with respect to $\log \lambda$ is given by

$$\frac{\partial \mathcal{L}}{\partial \log \lambda} = -\tfrac{1}{2}(-\mathrm{Tr}(\mathbf{S}\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}) + \mathrm{Tr}(\boldsymbol{\Sigma}\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L})) + \lambda\boldsymbol{\mu}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{L}\boldsymbol{\mu}$$

The roughness-penalty parameter λ is updated by

$$\lambda^{t+1} = \frac{(\mathrm{Tr}(\mathbf{S}\lambda^t\mathbf{L}^{\mathrm{T}}\mathbf{L}) - \mathrm{Tr}(\boldsymbol{\Sigma}\lambda^t\mathbf{L}^{\mathrm{T}}\mathbf{L}))}{\boldsymbol{\mu}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{L}\boldsymbol{\mu}}. \tag{3.21}$$

In practice, we obtain traces of $\mathbf{SA}$ and $\boldsymbol{\Sigma}\mathbf{A}$ by evaluating (3.19). We compute $\mathrm{Tr}(\boldsymbol{\Sigma}\lambda^t\mathbf{L}^{\mathrm{T}}\mathbf{L})$, and obtain the other trace in (3.21) by $\mathrm{Tr}(\mathbf{S}\lambda^t\mathbf{L}^{\mathrm{T}}\mathbf{L}) = N - \mathrm{Tr}(\mathbf{SA})$, and the trace present in (3.20) by using $\mathrm{Tr}(\boldsymbol{\Sigma}(\sigma^{-2})^t\mathbf{H}^{\mathrm{T}}\mathbf{H}) = N - \mathrm{Tr}(\boldsymbol{\Sigma}\mathbf{A}) - \mathrm{Tr}(\boldsymbol{\Sigma}\lambda^t\mathbf{L}^{\mathrm{T}}\mathbf{L})$.

TABLE 3.3: Comparative performance.

Method	SVs	D^0	D^2	D^4
RVM (old) [120]	6.5 ± 1.0	0.047 ± 0.010	0.054 ± 0.014	0.106 ± 0.030
RVM (new) [121]	6.0 ± 1.1	0.049 ± 0.010	0.056 ± 0.013	0.108 ± 0.026
Figueiredo [49]	6.3 ± 1.3	0.053 ± 0.012	0.067 ± 0.022	0.136 ± 0.051
KRLS [44]	17.0 ± 0.0	0.054 ± 0.012	0.077 ± 0.017	0.242 ± 0.066
SOG-SVR [43]	17.0 ± 0.0	0.062 ± 0.010	0.050 ± 0.008	0.062 ± 0.012
AO-SVR [84]	20.7 ± 3.2	0.057 ± 0.011	0.112 ± 0.029	0.374 ± 0.115
SRVM [this thesis]	16.3 ± 3.4	0.039 ± 0.009	0.030 ± 0.009	0.033 ± 0.013

3.3 Experimental Results

To measure the effectiveness of the proposed Bayesian model, we have performed a series of experiments. Because we need availability of the derivatives of the underlying function in order to be able to measure and compare performances, I have chosen the well-known sinc benchmark.

The data for one experiment consist of 50 points, with x_i uniformly distributed over $[-10, 10]$, and $y_i \sim \mathcal{N}(\mathrm{sinc}(x_i), 0.1)$. On these data, I trained a series of kernel machines, as shown in the first column of table 3.3. Besides the smooth relevance vector machine (SRVM) that I have proposed in this chapter, six other kernel machines have also been implemented with the kernel-machine library (described in chapter 5): three types of Bayesian kernel machines, and three types of deterministic kernel machines.

Two algorithms implementing the relevance vector machine (RVM) have been used; the algorithm presented in the original paper [120], and the more recent and faster algorithm [121]. Next is Figureido's method, which is also a Bayesian type of kernel machine using a Jeffrey's non-informative hyper-prior in order to promote sparse solutions. Figueiredo showed that his method is computationally faster than the original RVM, and that its solutions had the same quality as those of the RVM.

The kernel recursive-least-squares (KRLS, [44]) algorithm is a greedy online method that uses an approximate linear dependency test to select the active set of the support vectors. This way of determining the support vectors is also applied in sparse on-line greedy support-vector regression (SOG-SVR), which is also an algorithm by Engel et al. [43]. These two methods have in common that they are greedy, i.e., they do not find an exact solution, but rather trade-off accuracy against computational costs. This is not the case for the accurate online support-vector regression algorithm (AO-SVR, [84]). This algorithm applies changes to the parameters while maintaining the Karush-Kuhn-Tucker (KKT, [69, 77]) conditions. These conditions guarantee that a global optimum will have

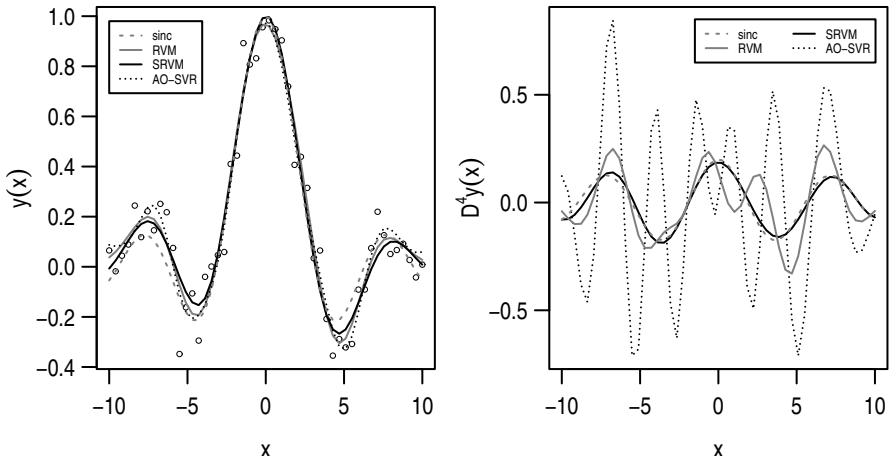

FIGURE 3.4: Input data to a single experiment, the underlying $\mathrm{sinc}(x)$ function, and the resulting RVM, SRVM, and AO-SVR (left). Right, the 4th order derivative of $\mathrm{sinc}(x)$, RVM, SRVM, and AO-SVR.

been found. Training points are presented to the algorithm one at a time, thereby enabling a truly incremental way to find the solution. It delivers an exact solution to the mathematical programming formulation of support vector regression [41].

For the smooth relevance vector machine (SRVM), I have chosen to penalise the 10th order derivative only, which should effectuate smoothness up to the 8th order [105]. A Gaussian kernel is used by all methods, parametrised by $\sigma = 1.6$. I repeated this experiment 1000 times. Table 3.3 shows the mean and standard deviation of the results, subsequently of the number of basis vectors, and of the root-mean-square of errors of zeroth, second, and fourth order derivatives.

As shown in table 3.3, smoothness requires notably more support vectors (SVs) than common sparse Bayesian models. In contrast, the support vectors seem to be spent wisely, because the error on all derivative orders is consistently lower than that of the compared methods. Figure 3.4 illustrates a function that is smooth in output space. Although at first sight the fit of the function itself is not drastically different from others (left), the gain in quality of fit is clearly visible in a higher order derivative (right): the SRVM shows a good match with the underlying sinc function.

"Engineering problems are under-defined, there are many solutions, good, bad and indifferent. The art is to arrive at a good solution. This is a creative activity, involving imagination, intuition and deliberate choice."

Ove Arup (1895–1988)

4 Modelling Electricity Load

This chapter deals with the merger between the domain specific knowledge of electricity-load patterns and kernel machines. In section 4.1, I will describe the generic architecture that is used for both the electricity-demand model and the wind-power production model. The modelling step of this architecture is discussed in section 4.2. For the electricity-demand model, the layout of a multi-component setup, and representations needed to embed the domain specific knowledge are developed in section 4.3 [14]. Section 4.4 blends wind-power production patterns with a kernel machine [15].

4.1 System Architecture

Because the electricity-demand model will have to satisfy different applications, the model type is chosen to be the causal type (see subsection 2.1.1). It can deal with missing values; it does not need a contiguous data set. The overall system architecture is shown in figure 4.1. It illustrates the three different stages of dealing with a load model: the modelling, calibrating and predicting stages (shown with a light-grey background colour). These three stages are discussed below.

- **Modelling.** Modelling involves creating and obtaining the model itself. Data analysis has to be performed, input parameters have to be selected, pre-processing steps have to be suited to the problem at hand, and also the needed historical data are selected. A more detailed description of

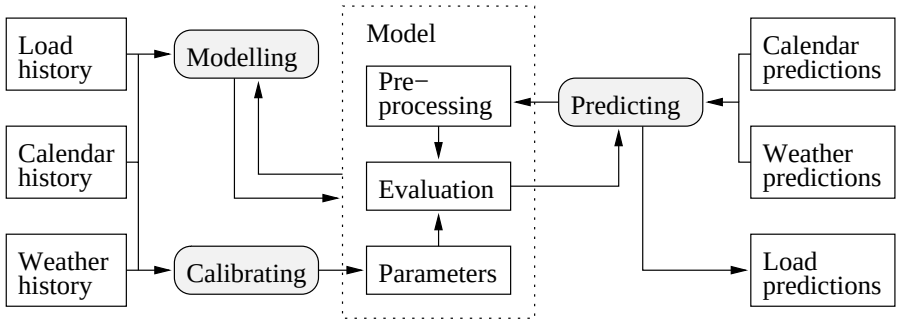

FIGURE 4.1: Architecture of the electricity-load prediction system.

the procedure used to obtain a model is described in section 4.2. Once the modelling is done, the model structure can be used for a number of different applications. The same model can be used to (re-)calibrate the its parameters to suit different electricity-load patterns. E.g., one model could be used for the electricity-load patterns of a series of customers. The modelling phase includes the use of historic data of load patterns, calendar events and weather patterns, but this phase also defines the structure of the patterns to be used for predictions. The calendar database may be implemented by a lookup table or scripts generating a series of dates and their associated day-types, but, it could also be that the calendar history is not needed at all.

- **Calibrating.** The need for recalibration of the model depends on the purpose the predictive model will be used for, the aim of calibrating a model is to (re-)fit the model to historic data. When modelling is done, the calibration boils down to finding the right parameters on the basis of historic weather measurements, historic calendar events, and of course, the historic dependant variables, the historic electricity-load patterns. The historic weather patterns should match the same period of the historic load patterns. E.g., when newer or other history has become available, one can consider to (re-)calibrate the model. If the calibration engine is enhanced (e.g., by the introduction of a newer type kernel machine), the predictive quality of all models can be enhanced.

- **Predicting.** When predicting, the model constructs an electricity-load prediction, aided by predicted calendar-events and numeric weather-forecasts fed to it. The model will return a load prediction, which is presented to an end-user by some graphical user-interface. For a day-to-day use of a short-term forecasting system, one could expect that the prediction stage is the most frequently consulted one. And for the other extreme, for customer

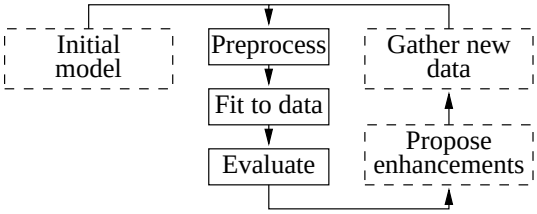

FIGURE 4.2: Modelling process.

profile analysis, it could be that the prediction step is performed only once.

The presented structure implies that for the model to be able to predict, it is adamant that predicted weather data are available from the same weather stations as were used during the modelling process. E.g., atmospheric data for the purpose of wind-power generation prediction should be available from all selected weather stations.

4.2 Modelling Process

The modelling process has been set up in the way depicted in figure 4.2. The process begins with an initially proposed model, depicted on the left of the figure. Next, the model will be fitted to the data after which the results will be evaluated. Hereafter, enhancements to the system will be installed. If necessary, new data will be collected and the processing steps repeated. A more detailed treatment of these steps is given below.

- **To pre-process.** Before a non-linear regression is executed by the kernel machine, I will apply a series of preprocessing steps to the input variables. For all kernel machines, I will use a Gaussian radial basis function (RBF) kernel [53], expressed by

  $$k(\mathbf{u}, \mathbf{v}) = \exp\left(-\tfrac{1}{2}\sigma^{-2}\|\mathbf{u} - \mathbf{v}\|_2^2\right)$$

  since it has shown good all-round results, also on time series predictions [96]. As a radial basis function, it has the foundation to formalise representations in a flexible way. However, inputs will have to be represented in a way that suits well to the Gaussian kernel. Although a kernel can be interpreted as an inner product in feature space, it can also be regarded as a measure of similarity. Because of this, I consider pairwise distances of patterns when designing preprocessing transformations.

- **To fit to the data.** Fitting of the model will be done by a kernel machine on the pre-processed data. Testing the quality of the model will be done by

visual inspection, and by measuring generalisation by means of a split of the data in a training set and a test set. A maximum generalising power is desired, i.e., a low train error alone is clearly not desirable.

- **To evaluate.** The evaluation is more an evaluation of the input space than an evaluation of the fit on the data itself; that has been taken care of in the previous step. Once a new variable is added, the correlation of the error with all existing input variables is checked. It is best if the input variable set is as linearly independent as possible with respect to the already created input space. The maximum errors are located and investigated for similarity of occurrence.
- **To propose enhancements.** Given the results from the previous steps, I can propose enhancements to improve the model. Information about the output variable that is inherent in the input space, or feature space, is of utmost importance [74]. While, in a theoretical sense, having more features should only give us more discriminating power, this is often not the case because of the added redundancy or noise. Supervised learning methods perform best if the training data contain as little irrelevant and redundant features as possible. As with linear regression problems, I will do a straightforward incremental selection of input variables.
- **To gather new data.** For a number of variables, new data will have to be gathered to be able to represent the needed variables. What variables have been gathered is described in the data analysis sections below.

4.3 Electricity Demand Model

A common start when creating models is to identify the variables with which a historically verifiable link can be established. In subsection 4.3.1, I seek for, and discuss, phenomena present in electricity demand patterns. The multi-component setup of the electricity demand model is motivated and elicited in subsection 4.3.2. Novel representations of several electricity demand specific concepts designed for a radial basis such as the Gaussian kernel are discussed in the next two subsections.

4.3.1 Data Analysis

I am able to use a data set measured at the province North Holland in the Netherlands, consisting of historical hourly electricity demand measurements made during 1996, 1997, and 1998. Also for the same period I have collected the hourly temperature, wind speed, and radiation measurements from a weather station located at Amsterdam Airport Schiphol as an indication for weather related circumstances.

The data set has been extended with the following calendar related attributes: day of the year, day of the week, local time of the day, and a daylight saving indicator. To describe events with a larger time horizon (such as seasonality), an accurate and complete collection of features describing Earth's position in the ecliptic (the geometric plane that describes the orbit of the Earth) has been gathered. Moshier's ephemeris calculator[1] is used to generate the longitude (angle), latitude (tilt), and the radius (distance) for all measurements. The Sun's altitude and the Sun's azimuth have been computed for the Netherlands. The Sun's altitude is the angle of the Sun with the horizon, while the azimuth expresses the angle between the Sun and the geographic north. Attributes describing all Dutch national, school, and construction holidays have been gathered and included.

Because data play a central role in electricity-demand modelling, I will discuss below the variables that I consider to have a relationship with electricity demand to a certain extent.

Sunlight

The Earth's axial rotation and orientation towards the Sun cause yearly and daily cycles in almost all its dynamic systems, and electricity consumption is not an exception. Sunlight intensity is an important natural factor that directly and indirectly influences electricity load. The most obvious direct influence is that lights will be switched on whenever the illumination of the surroundings drops below a certain limit.

As the Netherlands are not situated close to the equator, the daylight duration changes throughout the year. Figure 4.3 (left) illustrates this by showing the yearly electricity load pattern of the North Holland data on Mondays at 8 PM. It clearly shows the seasonality of summer (low loads) and winter (high loads) in the electricity load. The Sun's altitude (figure 4.3, right) indicates lower loads with higher altitudes, i.e., the higher the Sun rises above the horizon, the more light there is, the less electricity will be used.

Cloud cover may have such a dampening effect on the sunlight that is passing through that people switch on the lights as well. Also, in order to save energy, the clock is set forward and backward one hour in spring and autumn, respectively. The effect of this daylight saving time can best be noticed at times of the day when the light intensity changes considerably (i.e. at twilight).

Temperature

A part of the electricity consumption depends on temperature-sensitive processes or controllers. The electricity requirements of such processes are mostly negatively related to the temperature, for example electrical heating-systems and an aluminium factory require more energy when the ambient temperature is lower

[1] The software used for these calculations can be found at `http://www.moshier.net/`.

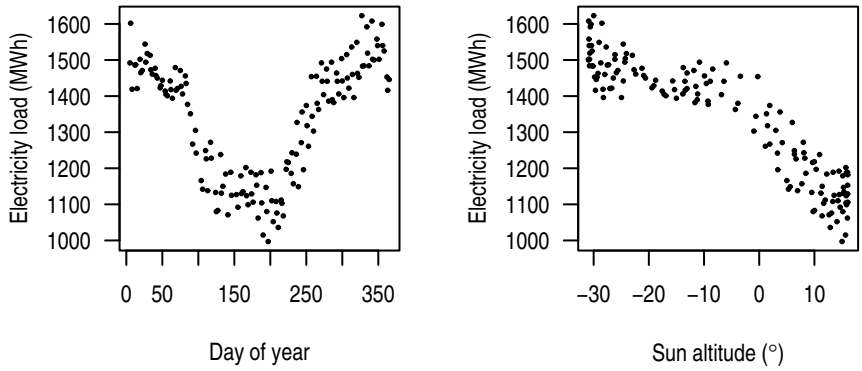

FIGURE 4.3: Yearly patterns of the North Holland data set of Mondays at 8 PM seen from different perspectives.

and vice versa. However, the opposite can also be true, e.g., for refrigerators and air-conditioning systems. The utilisation factor of these systems is one of the factors that determine the overall correlation between temperature and electricity use. Figure 4.4 (left) relates ambient temperature and electricity load measurements of the North Holland data set at 5 AM. Near-linear relationships show for the working days (the high density points at the top), and Saturday and Sunday (which are at the middle and bottom of the figure). Earlier work assumed linearity [e.g., 30, 75], but for the data under consideration, that hypothesis is falsified by inspection of figure 4.4 (right). This plot of the North Holland data at 5.00 hrs demonstrates non-linearity between ambient temperature and electricity load. Brierley and Batty [20] noted that people tend to react slowly to quickly changing weather conditions. If not taken into account, this could lead to wrongly estimated electricity demands after relatively large changes in temperature.

Wind

Wind is a weather variable of which the direct relation with the electricity demand is hard to establish. However, Brierley and Batty [20] found that electricity loads on windy days tend to be systematically underestimated if the wind chill factor is not taken (sufficiently) into account. The same is true for buildings that are sensitive to draught. They also reported that thunderstorms can greatly influence electricity requirements.

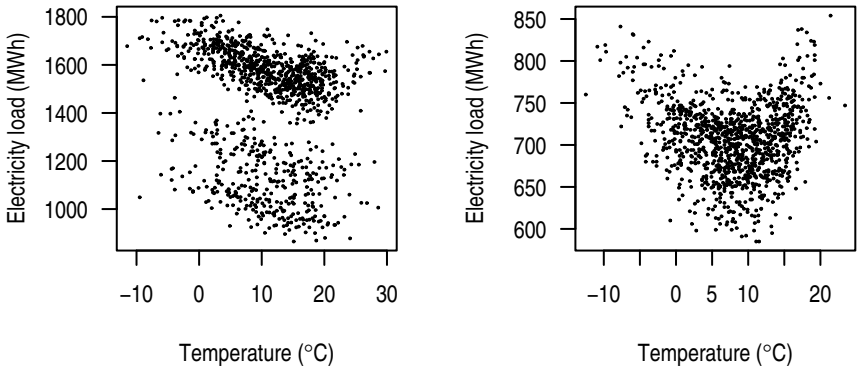

FIGURE 4.4: Temperature versus load at 1 PM (left) and at 5 AM (right).

Working Hours

Despite the continuing move towards a 24-hour economy in the Netherlands, people tend to sleep at night, and most jobs are still from 9 to 5. During working hours, more electricity-consuming devices (such as heavy machinery) are used than outside working hours. In the same context, a typical weekly load pattern can be identified because the days from Monday through Friday are typical working days.

Holidays occur rather irregularly while influencing large groups of consumers, and therefore they are one of the more interesting phenomena. This is exemplified in figure 4.5 by a period surrounding the Queen's Birthday and Ascension day of 1996 of the North Holland data. The first days of interest in figure 4.5 are the Queen's Birthday (point 2) and Ascension day (point 4), showing an irregular electricity usage pattern because most people do not work on public holidays. In terms of electricity load, these days look like Sundays. The subsequent Thursday and Friday following the Queen's Birthday (point 2) show no drastic differences from normal. However, the Friday (point 5) after Ascension day clearly shows that many people take the day off between two non-working days [75]. These days are called bridge days. Other atypical days can also be identified, e.g., in the south of the Netherlands, electricity use during carnival shows typical public holiday patterns. Also, Good Friday, being actually a working day two days before Easter, shows similarities to the pattern of a bridge day. For similar reasons Christmas Eve and New Year's eve (may) show the same pattern.

School holiday periods and the summer holidays show recurring yearly pat-

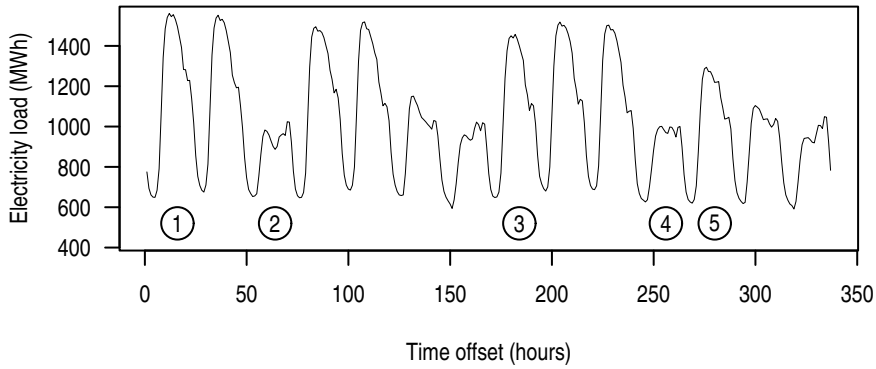

FIGURE 4.5: A period of two weeks in 1996 starting on Monday (point 1), with two public holidays (points 2 and 4), a school holiday (exemplified by point 3) and a bridge day (point 5).

terns of electricity demand. The Netherlands stagger the timing of several school holidays over three regions so as to prevent traffic jams and over-occupied recreation areas. Non-summer school holidays start early in the North region, then in the Central region, and the South region vacations start last [2]. Summer holidays fall in different periods, using the same staggering method as the school holidays, but with a less straightforward starting scheme. The starting times are rotated over a period of six years, in which a region will start early twice, then middle once, late twice, and middle once again. An interesting issue also in this category is the construction holidays (the Dutch *Bouwvak*), because a large part of the heavy-duty machinery is not operating during that time. Construction holidays are divided over three regions like the school summer holidays and the first construction holiday starts at the same time as the first school summer holidays. The impact of a school holiday is visible when comparing the Monday at point 3 in figure 4.5 with the Monday a week before, at point 1. The impact of the various holidays may vary considerably; e.g. the Christmas holiday has a greater impact on the electricity load than any other holiday.

Extreme Events

Happenings such as strikes and large media events can change the electricity use pattern to a large extent. For instance, football matches may have such a big influence because a large part of the population will be glued to its TV set. Electricity use is lower than usual, but during half-time people will take a beer,

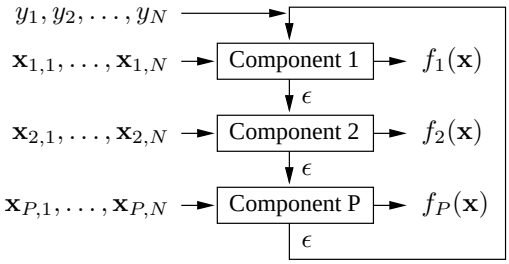

FIGURE 4.6: Multi-component architecture and its iterative fitting procedure.

turn on lights etc., and a peak in demand results. These changes in demand are called *TV pick ups*[2]. Only a few particular football matches seem to have this influence; otherwise the effect is negligible.

Indirect Influences

Sunlight and half-time of a football match have an immediate effect on the electricity load, which other variables, such as temperature, have not: e.g., it takes a while before a building cools down or warms up. Historical (or lagged) recordings can be introduced to make these time-delayed influences measurable.

4.3.2 Multi-component Setup

Let us assume that the relative influence of temperature, wind, and other weather related variables is the same for all day types. To increase the orthogonality of these (and other) input variables, we do not want to mix the input space in which the day types reside with that of the temperature. Quite a common approach is to combine the predictions in such way, that the electricity load is assumed to be a (weighted) sum of a number of components such as the base load, trend, season, temperature and irregular components [100]. An increase in orthogonality can be achieved by a weighted sum of different kernel machines from (2.6), given by

$$y(\mathbf{x}) = \sum_j f_j(\mathbf{x}) = \sum_j \left(w_{j,0} + \sum_{n=1}^{N_j} w_{j,n} k(\mathbf{x}_{j,n}, \mathbf{x}) \right). \tag{4.1}$$

The fitting procedure is illustrated by figure 4.6. We start by fitting the first kernel machine f_1, or component 1, to the data. The next step will be to fit component

[2]The National Grid website at `http://www.nationalgrid.com/uk/activities/mn_pickup.html` shows several TV pick ups.

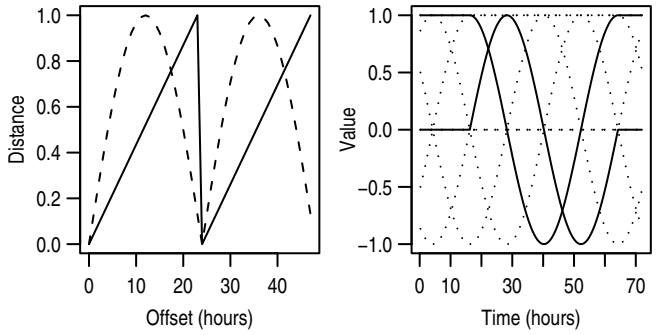

FIGURE 4.7: Cyclic time-of-day representation distances (left), and cyclic day-type representation (right).

2 to the residual of step 1. This procedure is repeated for all P components until the residuals of all components have converged.

4.3.3 A Novel Day-type Representation

One of the main characteristics of electricity load data are periodic events such as days, weeks and years. Although this set of periodic events sounds quite straightforward, it is one of the hardest representation problems when modelling electricity load.

Time of the day has been represented in various ways: in a binary form, a linear form and a trigonometric form. The linear form is represented by something like $x_i = c \cdot t_d$, which can be a numbered hour or half hour [e.g. 1]. The time-of-day can also be represented by binary encoding [e.g. 71]. The disadvantages of the linear forms are the jumps in value if the day changes (i.e. a sawtooth function over time). A possible solution to this is the other form, which is a sine-cosine quadrature [45, 47, 95], which expresses the normalised time of day t_d using $x_i(t) = \sin(2\pi t_d)$ and $x_j(t) = \cos(2\pi t_d)$. From a distance point-of-view, the polar variant is the correct one in terms of day-to-day transitions [45]. Figure 4.7 (left) illustrates this by showing the computing distance from a fixed time of day to the future.

Although weeks are regular periodic events, many different week patterns may be present in the data because of public holidays. Because a day is a nominal variable, every day is assigned an associated *day type*. A great deal of research has been performed to figure out how many different day-types could be defined in electricity-load forecasting by using self-organising maps (see subsection 2.3.1). Without loss of generality, I will assume that every day of the week has a unique demand pattern [42].

Nearly all researchers use a 24-hour day-type representation for each calen-

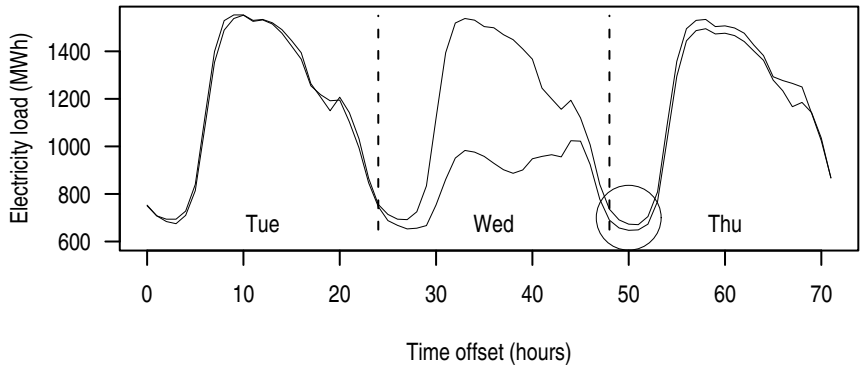

FIGURE 4.8: The error made by *all* systems using 24-hour day-type representations is illustrated in the circle near hour 50.

dar day starting at midnight. However, this representation does not force smooth day-to-day transitions on the system: it does not consider the case that a day type may differ when the day is followed by, preceded by, or by chance is a public holiday, which will cause a jump on the transition point. Figure 4.8 shows the adjacent days to the Wednesday at point 2 of figure 4.5, and of the Wednesday a week before that, when no public holiday occurred. The circle near hour 50 in figure 4.8 illustrates that if the day-type representation of Thursday does not take into account that the preceding day has been a public holiday, the modelled electricity demands will most probably show a jump at the day transition point and also most likely cause a considerable error.

Therefore, I propose to extend the scope of a day type to more than 24 hours so as to ensure smooth transitions between day types. To keep the number of possible states in terms of number of active day types to a minimum, the length of day types is increased to 48 hours. This ensures that at any time two day types are active. These 48-hour periods will start and end at the time of day at which the variance of the loads is smallest on average (e.g., 4 AM). To represent the day types, I propose a combination of a binary day-type representation and a polar representation: all day types will have their own circle of 48 hours. Figure 4.7 (right) illustrates this representation.

4.3.4 Emphasising Twilight

To represent time of year (or seasonality), I propose several alternative representations of which we select one on the basis of experiments described in chapter 6.

The first alternative will be a polar indicator for time of year, with the position on the circle determined by the position of the Earth in the ecliptic. The second proposed representation will be the Sun's altitude: the angle of the Sun with the horizon.

Sun light is often considered to be incorporated in a time of the year indicator; few researchers considered representations for direct daylight [45]. At civil twilight, i.e. when the altitude of the Sun reaches below -6°, artificial illumination is needed to be able to read. We will adopt the sigmoid function from classical artificial neural networks to accentuate the twilight period of the Sun altitude values,

$$x_i(t) = \text{sigmoid}(\lambda \, \text{altitude}(t)).$$

and scale parameter λ in such a way that the function is $\sim$1 near 6° and $\sim$-1 near -6°. This twilight indicator will be resilient during the application of the daylight savings times, but only during twilight, and it will be dormant at other times of day.

4.4 Wind-power Production Model

This section consists of three subsections, starting with an analysis of the data at hand, followed by a proposed method to increase the wind speed resolution, and a way to represent the wind direction.

4.4.1 Data Analysis

The data available for the case of short-term wind-power production forecasting are the production data of wind farms of an electricity company in the Netherlands. They are the aggregate of a large number of telemetered connections. Although it seems straightforward to acquire these data, it is a tremendous undertaking to collect them all and to verify that they contain all the elements needed for appraisal of the performance of wind power plants.

Figure 4.9 shows the produced wind power and wind speeds for one week. It illustrates energy production that is typical of wind-power production, i.e., on the first day, almost no energy production occurred, while on the fifth day, the production is often well above 150 MW. When wind speeds are not sufficiently high, no production takes place at all. The variation in correlation between wind speeds and the wind-power production is significant. This is exemplified by point 1 and point 2 in figure 4.9. At point 1, wind-power production is significantly lower than at point 2, while the wind speeds at point 1 are marginally higher than those at point 2.

We do not have any wind speeds measured at the parks themselves, but rather measured at weather stations operated by the Dutch meteorological institute.

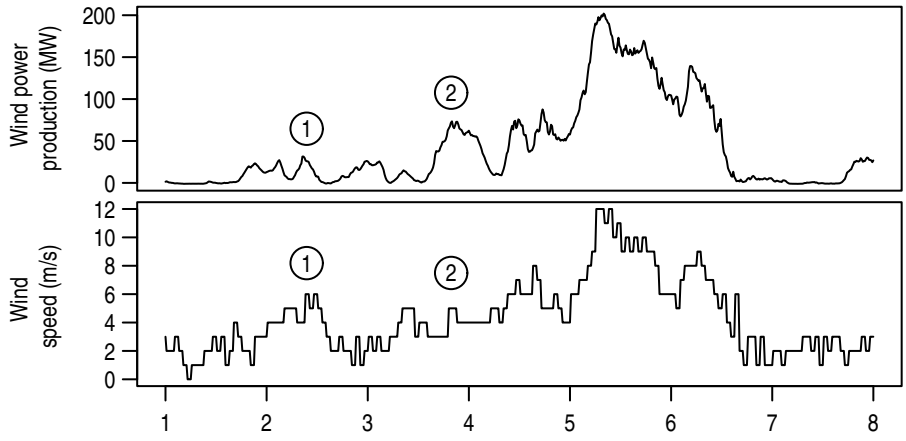

FIGURE 4.9: Seven days of wind energy production (top), and their corresponding measured wind speeds at the same time (bottom).

TABLE 4.1: Correlation matrix of available weather stations and the aggregate wind-power production.

	Amsterdam	De Kooy	Stavoren	Valkenburg	Production
Amsterdam	1.00	0.851	0.810	0.875	0.864
De Kooy		1.000	0.821	0.799	0.881
Stavoren			1.000	0.791	0.857
Valkenburg				1.000	0.831
Production					1.000

Traditionally wind speeds were measured in knots[3]. The potential wind speed is computed from the measured wind speed. The correction factor is usually between 0.9 and 1.2. The distribution of the potential wind speed therefore will be clustered around the original value in knot. The potential wind speed is reported in 0.1 meter per second, but the resolution of the records from which it was computed is approximately 0.5 meter per second. From July 1996 onwards wind speeds are measured in integer values of meter per second. So, since then the resolution is even less. We have available wind speeds measured from the four stations mentioned in table 4.1. This table also displays the correlations between the wind speeds measured at each station, and the recorded wind-energy production.

[3]Source: website KNMI `http://www.knmi.nl`.

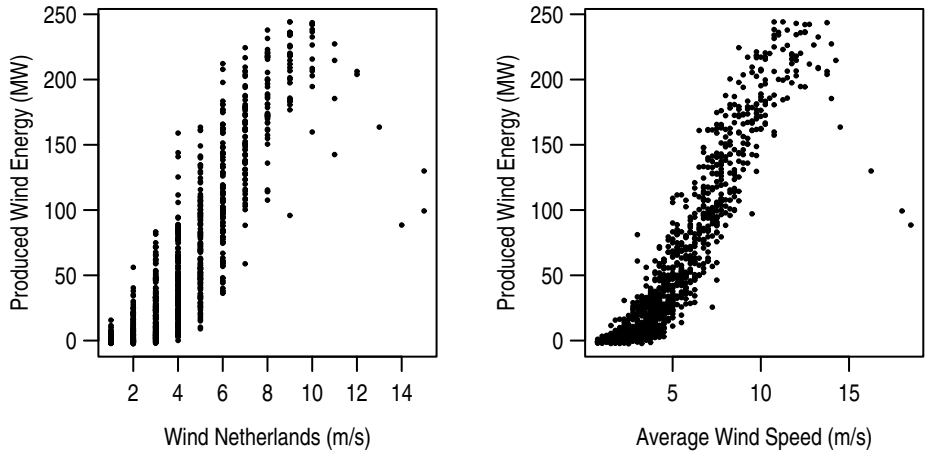

FIGURE 4.10: A wind-power curve using the discretised wind speeds (left), and using the computed average wind speeds (right).

4.4.2 Increasing Wind-speed Resolution

Because of the time window of 14 days between submitting E-programs and their definite allocations, a *causal model* as introduced in subsection 2.1.1 will be used. Therefore, I have to resort to the numerical weather predictions (i.e., I cannot use a time-series model). A lack of resolution in wind speed records as mentioned in subsection 4.4.1 is one of the first issues that one encounters on the way to a wind forecasting system. If this resolution is not improved, the predictions will suffer from the same discretion level. Figure 4.10 (left) illustrates this by the power curve of the data using the wind speed of the Netherlands. We propose to overcome the problem of coarse resolution by taking into account multiple weather stations. Figure 4.10 (right) shows the wind-power curve of the data in combination with averaged wind speed of the four weather stations presented in table 4.1.

4.4.3 Representing Wind Direction

By taking into account the direction of the wind, I assume that a wind-power production farm is surrounded by a varying landscape. For instance, assuming identical wind speeds, if a forest is to the East, and open sea to the West, one could expect that if the wind blows from the East, a lower production would occur, due to causes such as turbulence. Additionally, in most cases meteorological institutes measure and predict weather on a different altitude than where the wind-power turbines are located.

The direction of the wind is recorded in degrees to North. In order to optimise

this entity to work in combination with a radial basis function, we will map the polar representation to an Euclidean representation during a pre-processing step. The exact setup of the feature space for the wind energy prediction system will be done by experiments and in an incremental way as described in section 4.2.

"Science is what we understand well enough to explain to a computer. Art is everything else we do."

Donald Knuth (1938–)

5

The Kernel-Machine Library

Since the introduction of the support vector machine (SVM), about a dozen software packages have been introduced that implement support vector machines. SVM^{light} [66] contains a number of extensions e.g. a preference ranking problem, and extensions to deal with structures such as trees. LibSVM [32] is distinct in that it gives a user-friendly introduction to support vector machines. SVMTorch [36] was primarily made to deal with large data-sets, and is now part of the more generic Torch machine-learning software package. All the previously mentioned packages have two things in common: they are geared towards the SVM, and are developed as end-user tools. Creation, improving, or embedding of other types of kernel machines are not their primary objective.

I initiated the development of the kernel-machine library (KML) to fill that void: it should be an open-source[1] library to promote the use and progress of kernel machines, both for academic use and for developing real-world applications. This is the objective of the kernel-machine library: to enable researchers to develop other types of kernel machines, to improve upon the existing algorithms, and to be able to integrate kernel machines into all kinds of applications. The library is available at `http://www.terborg.net`. A concurrent-versioning system is set up to improve joint development over the Internet. As such, it is subject to constant change, with new features and methods being added continuously.

[1] It is released under the GNU General Public License of the Free Software Foundation.

5.1 Requirements

Here I formalise my initial requirements for the kernel-machine library.

- **It should be efficient.** Because kernel machines are all about numerical computations, one of the most important topics will be to achieve efficiency. Efficiency can be expressed in the theoretical performance of a central processing unit (CPU), often expressed in floating-point operations per second (FLOPS).

- **It should have a clean application programming interface (API).** Because it is a development library, one of the more difficult things of the development of a software library compared with the development of an end-user software tool is the quality of code: co-developers have to be able to read the source code. Good documentation, good use of comments, and sensible names of parameters can help, but foremost the design should be clear and simple.

- **It should be cross-platform.** With a dominance of more than 90% of the desktop market, Microsoft Windows systems do contain many potential users and should be well supported. It should be tested on different operating systems.

- **It should be flexible.** It should be able to accommodate all kinds of kernel machines, and therefore be able to process data set formats used by other packages.

- **It should be extensible.** In case an unforeseen situation comes into play, it should be possible to adjust the library in line with new ideas with minimal effort without altering the validity of its contents.

5.2 Design

The exact design of software depends on the capabilities of the language it is written in. The C++ language is a *multi-paradigm* language, in other words, the language supports procedural, object-oriented, and even functional program structures. Recently, one of C++'s features achieved much attention: its template typing system is extremely powerful, as it allows for Turing-complete programming during compile-time: the time when the program is being compiled by a compiler. Run-time is when the program is under execution, i.e., running. This will be an important issue in the design of the KML: selecting the actions to be taken in compile-time, and what actions will be executed in run-time. The following may exemplify this distinction,

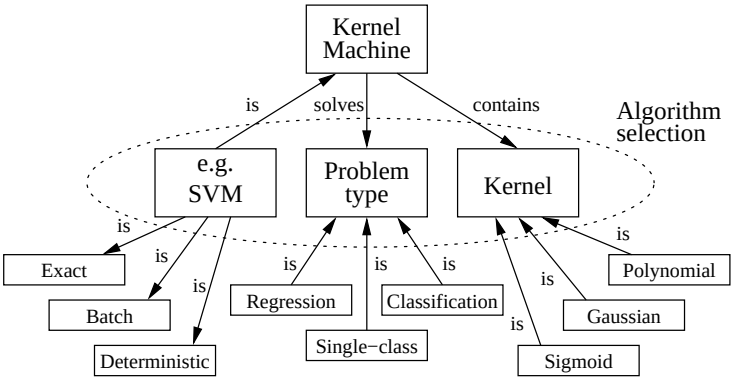

FIGURE 5.1: Entity-relation diagram used in the kernel-machine library.

```
template<int N>
struct factorial {
  static const int value = N * factorial<N-1>::value;
};
template<>
struct factorial<0> {
  static const int value = 1;
};
```

and its run-time counterpart would look like as follows

```
int factorial_rt( int n ) {
   return ( n==0 ? 1 : n * factorial_rt(n-1) );
}
```

Now, if a factorial needs to be computed from a known number N, this can be done by the compiler at compile-time, reducing the number of computations needed at run-time to zero.

Figure 5.1 shows an entity-relation diagram used in the library. A kernel machine is a term for a combination of several concepts: a problem type, a kernel and an algorithm. The problem type is the type of problem to be solved, e.g. classification, regression, or single class. Given a problem type, a kernel-machine states how the problem is solved and by what mechanisms. For instance, for a regression problem, an SVM has a mathematical optimisation problem different from that for a classification problem. A kernel machine *contains* a kernel; it *is* not a kernel. However, algorithms have been devised that are specialised towards a kernel machine in combination with a specific kernel. Therefore, a kernel can be responsible for the selection of the algorithm used by the kernel machine.

Kernel machines sometimes solve identical problems in different ways: here, trade-offs and guarantees of pre-learning and post-learning come into play.

- Batch machine or on-line machine. A batch machine will only fit to a complete data set. It cannot modify its solution efficiently if the data set changes. An on-line machine is capable of adjusting its found solution if the data set changes.
- Exact machine or greedy machine. An exact machine solves the mathematical optimisation problem with machine precision; a greedy machine approximates the solution found by an exact machine in shorter time.
- Deterministic machine or probabilistic machine. A deterministic machine can predict the mean value of a sample, whereas a probabilistic machine can predict both the mean value and variance (or more moments).

These properties have been used deliberately as traits: only a few formulations have available all variants (on-line, greedy, etc.), and some kernel machines exist that have their own name, and have certain traits, but are not designed to execute tasks outside these traits.

5.3 Implementation

The kernel-machine library consists completely of C++ header files. Users of the library can include appropriate headers to their source code. A significant reduction of development time has been achieved by using a number of libraries from the Boost Libraries, as discussed in subsection 5.3.1. For efficient numeric computing, one usually resorts to a basic linear algebra system (BLAS) of which ATLAS is a fast variant, discussed in subsection 5.3.2. Kernel machines share types of operations on matrices and vectors, which are included in the kernel-machine library and discussed in subsection 5.3.3.

5.3.1 The Boost Libraries

As stated on `http://www.boost.org`, the Boost web site provides free *peer-reviewed* portable C++ source libraries. The emphasis is on libraries which work well with the C++ Standard Library. The libraries are intended to be widely useful, and are in regular use by thousands of programmers across a broad spectrum of applications. A further goal is to establish "existing practice" and provide reference implementations so that Boost libraries are suitable for eventual standardisation. Ten Boost libraries will be included in the C++ Standards Committee's upcoming C++ Standard Library Technical Report as a step towards becoming part of a future C++ Standard.

Libraries from the Boost Libraries that are in use by the kernel-machine library are the following.

- Boost.Meta-Programming Library. This can be regarded as the compile-time variant of the ISO C++ Standard Library.
- Boost.UBLAS Library. This library contains class structures dealing with matrices, vectors, sparse matrices and sparse vectors. It supports BLAS operations (treated in more detail below).
- Boost.Utilities Library. Several components from the Boost Utilities library are used, such as Enable_if, and ... Type-Traits Library.
- Boost.Serialization Library.
- Boost.Range Library.

5.3.2 BLAS: ATLAS

During the 1970s, an enormous effort has taken place to formalise the numeric computations needed by scientists. This effort resulted in numeric packages such as LINPACK and LAPACK (short for Linear Algebra PACKage). LINPACK's successor LAPACK used a new system for the operations (besides that an other type of computer memory organisation was utilised), called the Basic Linear Algebra System (BLAS). This divides the linear algebra operations into the following

- BLAS level 1: scalar-vector and vector-vector operations,
- BLAS level 2: vector-matrix operations,
- BLAS level 3: matrix-matrix operations.

LAPACK contains algorithms for operations such as Cholesky decompositions and QR factorisations for solving positive definite systems of equations. I have rewritten the smooth relevance vector machine using BLAS and LAPACK algorithms as shown in algorithm 1.

Because the operations are clearly defined, one can speed up all LAPACK operations by using a better BLAS engine. The automatically tuned linear algebra software (ATLAS) project has been ongoing for a while [132]. It has a full implementation of BLAS as well as a few important routines from LAPACK. ATLAS "tunes" towards speed of memory and size of memory assuming a memory structure of modern CPUs, as illustrated in figure 5.2. Several line and grid searches are deployed to find the optimum size of these blocks (a sub-matrix of a matrix) for all different BLAS operations. Several specific user-contributed algorithms are available that fully utilise extensions such as AMD's 3DNow!, IBM's AltiVec and Intel's SSE. The ATLAS library is a C library. Fortunately, the Boost.Numeric.Bindings library provides the glue between ATLAS and Boost's uBLAS data types.

H ← design matrix(x, k)
L ← design matrix(x, D^p k)
HtH ← gemm($\mathbf{H}^{\mathrm{T}}$, $\mathbf{H}$)
LtL ← gemm($\mathbf{L}^{\mathrm{T}}$, $\mathbf{L}$)
Hty ← gemv($\mathbf{H}^{\mathrm{T}}$, $\mathbf{y}$)
while $\max(\Delta \log \alpha_i) > 1 \cdot 10^{-3}$ **do**
  $S \leftarrow$ posv($(\lambda LtL + A)$, I)
  $\Sigma \leftarrow$ posv($(\sigma^{-2} HtH + S^{-1})$,I)
  $\mu \leftarrow$ symv(σ^{-2}, Σ, Hty)
  for i in N **do**
    $\alpha_i \leftarrow \alpha_i(S_{ii} - \Sigma_{ii})/\mu_i^2$
  end for
  $\lambda \leftarrow (N - \mathrm{Tr}(SA) - \mathrm{Tr}(\Sigma LtL))/\mu LtL\mu$
  $\sigma^2 \leftarrow \|\mathbf{H}\mu - \mathbf{y}\|/(N + \mathrm{Tr}(LtL) + \mathrm{Tr}(\Sigma A))$
end while

ALGORITHM 1: Algorithm for the Smooth Relevance Vector Machine expressed in BLAS calls.

5.3.3 Utilities

View matrix

The view matrix class has been implemented to cover recurring operations found in a large number of active-set type of algorithms. In algorithms of kernel machines, the design matrix or kernel matrix is often extended when a new support vector is added, or is shrunk when a support vector is deleted from the active set. The standard matrix resize operation as implemented in the uBLAS library supports a resize, but will reallocate an entire matrix if requested. Figure 5.2 (right) illustrates the memory structure used by a view matrix: the larger, static matrix will be reallocated infrequently, whereas the view matrix may be resized in constant time $\mathcal{O}(1)$.

Input and Output

To be able to read and write data sets in formats used by other software packages, an IO module is written that is able to read and write from the file data format structures as used in the SVM^{light}, LibSVM and SVMTorch. This can be very convenient if one wants to compare the output of the different packages.

Statistics

For assessment of the quality of fit, or other types of statistics, a number of functions have been implemented that are able to operate on any type of range.

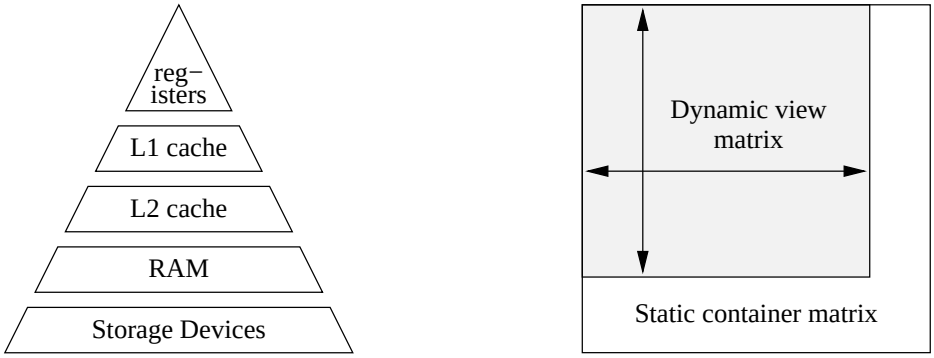

FIGURE 5.2: The pyramid of a modern CPU's memory structure (left), memory structure of a view matrix (right).

Root-mean-square, variation, standard-deviation, mean, sum, minimum, maximum, etc., have all been implemented. In case of a range of a scalar type, i.e, the ordinary case, the output is also a scalar type. On the other hand, in case of a range of a vector type, the output is chosen to be a vector type: the statistic is computed separately for each attribute. For example, the mean of two sample vectors

```
[8](1,1,2,3,5,8,13,21)
[8](1,2,3,5,7,11,13,17)
```

will be evaluated to `[8](1,1.5,2.5,4,6,9.5,13,19)`. Also for these cases, compile-time mechanisms select algorithms for either scalar types or for vector types.

Scaling

To accommodate a starting point for feature scaling and representation processes, two types of scaling functions have been implemented. The first function will perform a linear transformation such that the mean and standard deviation comply to a given number. The second type of scaling function will force all data to be within a set minimum and maximum. To exemplify, suppose one has a data set of which the input vectors have characteristics

```
min  [4](-1,-1,-1,-1)
mean [4](0.0597222,0.355556,0.449383,-0.295388)
sd   [4](0.379544,0.936391,0.633393,0.337012)
max  [4](1,1,1,1)
```

then this could be transformed to

```
min  [4](-2.79209,-1.44764,-2.28828,-2.09077)
mean [4](0,0,0,0)
sd   [4](1,1,1,1)
max  [4](2.47739,0.688222,0.869313,3.84375)
```

with a single function call.

FANN bindings

As stated on its website (`http://leenissen.dk/fann/`), the fast artificial neural network (FANN) library implements multilayer artificial neural networks in C with support for both fully connected and sparsely connected networks. Cross-platform execution in both fixed and floating point is supported. It includes a framework for easy handling of training data sets. Although a C++ API is available, I have written new bindings to achieve a more consistent API throughout the kernel-machine library. In other words, the implementation of a neural network is similar to the implementation of a kernel machine.

5.4 Development Tools

Primary requirements for the KML are a C++ compiler and a linker. I will test the library on two kinds: Microsoft's (MS's) freely available Visual C++ Toolkit and GNU's Compiler Collection (GCC). Although the KML is contained in header-files only, cross-platform construction of several example and test programs is desired. I have used the Scons build system to achieve all of these goals, and more, which is discussed in subsection 5.4.1. Documentation is generated on the basis of comments in the source code by the Doxygen system treated in subsection 5.4.2.

5.4.1 The SCons Build System

As described on `http://www.scons.org`, SCons is an open-source software construction tool, that is, a next-generation build tool. SCons is an improved, cross-platform substitute for the classic Make utility with integrated functionality similar to autoconf, automake and compiler caches such as ccache. In short, SCons is an easier, more reliable and faster way to build software.

Examples included in the kernel-machine library are built with the use of SCons. On a Microsoft Windows system using Microsoft's compiler, the output of building an example application to do regression from the KML looks like the following.

```
scons: Reading SConscript files ...
scons: done reading SConscript files.
scons: Building targets ...
```

```
cl /nologo /Wp64 /GX /Zc:forScope /O2 /G7 /arch:SSE2 /I. /I\
/I\atlas /c example\svm_regression.cpp /Foexample\svm_regres
sion.obj svm_regression.cpp
link /nologo /OUT:example\svm_regression.exe /LIBPATH:lib cb
las.lib example\svm_regression.obj
scons: done building targets.
```

On a POSIX system (e.g. a GNU/Linux OS) using GCC, the output of the compile process is shown below.

```
scons: Reading SConscript files ...
scons: done reading SConscript files.
scons: Building targets ...
g++ -Wall -ansi -pedantic -O3 -ffast-math -fomit-frame-point
er -DNDEBUG -DNO_DEBUG -march=pentium4 -mfpmath=sse -msse -m
sse2 -mmmx -I. -I/usr/include -c -o example/svm_regression.o
example/svm_regression.cpp
g++ -o example/svm_regression example/svm_regression.o -lcbl
as -latlas
scons: done building targets.
```

I have facilitated the build scripts with automatic cross-platform CPU detection, which will pass the optimal settings to the compiler for the platform at hand. This can be observed in the output snippets above for an Intel Pentium 4 platform where the SSE2 extensions are enabled automatically. Because the installation of ATLAS under Windows is a somewhat extensive procedure, I have included a SCons construction script in the KML to convert ATLAS constructed libraries to MS Windows native loadable libraries. These can then be easily deployed on Microsoft Windows based platforms.

5.4.2 The Doxygen Documentation System

As advertised on its website (located at `http://www.doxygen.org`), Doxygen is a documentation system for C++, C, Java, Objective-C, Python, IDL and to some extent PHP, C#, and D. It can generate an on-line documentation browser (in HTML) and/or an off-line reference manual (in) from a set of documented source files. There is also support for generating output in RTF, PostScript, hyperlinked PDF, compressed HTML, and Unix man pages. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code.

In a comment just above the implementation of a function, a few lines describing the functionality of the function can be provided.

```
\*! \brief Compute the factorial of a given number n, i.e., n!
```

```
    \param n an integer n */
int factorial( int n ) { ... }
```

This will be translated and aggregated to browseable HTML pages for convenient lookup. For classes, template classes, member functions, and so on, hierarchy in the documentations will be added automatically.

5.5 Using the Library

I have published the initial version of the KML (at `http://www.terborg.net`) with the following algorithms.

- Support Vector Machines. Two variants of the support vector machine are included: Accurate On-Line Support Vector Regression [84], and Sparse On-line Greedy Support Vector Regression [43].
- Relevance Vector Machine [119]. Both the classical learning algorithm [120] and newer fast learning algorithm [121] for the RVM have been included. The fast algorithm is not available elsewhere.
- Kernel Recursive Least Squares [44].
- Adaptive Sparseness using Jeffreys Prior [49].
- Smooth Relevance Vector Machine [13].

To illustrate the expressiveness of the kernel-machine library, the following source code creates a noise-corrupted sinc on $[-10, 10]$, and will train an on-line SVM to it, as is done in section 3.3.

```
boost::mt19937 randomness;
boost::normal_distribution<double> norm_dist( 0.0, 0.1 );
boost::variate_generator<boost::mt19937,
      boost::normal_distribution<double> > noise( randomness,
                                                  norm_dist );
ublas::vector<double> y(50);
std::vector< ublas::vector<double> > x(50);
for( int i=0; i<N; ++i ) {
  x[i].resize( 1 );
  x[i](0) = (double(i)/double(N-1))*20.0-10.0;
  y[i] = boost::math::sinc_pi(x[i](0)) + noise();
}
typedef kml::regression< ublas::vector<double>, double >
        my_problem;
kml::online_svm< my_problem, kml::gaussian > my_machine( 1.6,
```

```
                                                    0.1, 10.0 );
my_machine.learn( x, y );
ublas::vector<double> test(N);
std::transform( x.begin(),x.end(),test.begin(),my_machine );
std::cout << "AOSVR predictions:" << test << std::endl;
```

Compiling and running this example program will give the following output, which is also plotted in figure 3.4.

```
AOSVR predictions:
[50](0.0874446,0.081959,0.08579,0.105701,0.144144,0.19371,0.
235582,0.245479,0.205863,0.117594,0.00262615,-0.105663,-0.17
8993,-0.206344,-0.193398,-0.150872,-0.0822103,0.0187777,0.15
8834,0.333346,0.522492,0.698752,0.839093,0.931865,0.974475,0
.966872,0.908814,0.803572,0.662969,0.506631,0.353948,0.21483
6,0.0873798,-0.03512,-0.15312,-0.254647,-0.318991,-0.327899,
-0.276819,-0.179247,-0.0620642,0.0449141,0.119503,0.153157,0
.150841,0.126204,0.0950485,0.0699378,0.0575196,0.0586123)
```

5.6 Testing the Library

This section tests the kernel-machine library in action. In the first subsection, experiments are reproduced that have been presented in the paper describing the fast relevance vector machine. The second subsection deals with the currently implemented algorithm for the support vector machine (SVM) on a regression task.

5.6.1 The Fast RVM

At the time of writing, the kernel-machine library is the only publicly available software package that has an implementation of the fast relevance vector machine (Fast RVM, Tipping and Faul [121]). The main difference between the algorithms that cause a computational speed-up is that the old algorithm is a decremental-set algorithm, and the new algorithm is an incremental-set algorithm. Here, decremental means that the initial solution holds all data points, and by numerical updates non-relevance vectors will be removed. The newer algorithm on the other hand starts with an empty solution, adds candidate relevance vectors to the working set, and removes non-relevance vectors.

To measure the effectiveness of the fast RVM algorithm, I have conducted the experiment described in the paper where the fast RVM was introduced [121]. To elicit the gained computational efficiency of the newer algorithm, a series of experiments has been conducted. In this case, an increasing number of samples is drawn from a noisy sinc function, identical to that described in section 3.3. These

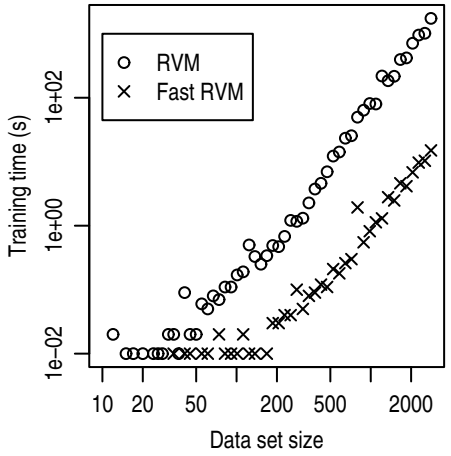

FIGURE 5.3: Comparative performance of the old RVM versus the new RVM algorithm.

experiments have a strong focus on data set size. For the p-th experiment, I have used a data set size of $10 \cdot 1.1^p$. In total, 59 experiments have been conducted by both the old and the new algorithms. Figure 5.3 shows a log-log plot of the data sets used in the experiments versus the needed fitting time of both algorithms. On average, the newer algorithm was about 100 times faster, which is very significant. To illustrate, the largest data set size tested, containing 2812 points, took the old algorithm about 29 minutes to complete, whereas it took the accelerated algorithm about 15 seconds.

5.6.2 On-line SVM versus Batch SVM

In this experiment, I tested the accurate-online support vector regression (AO-SVR, [84]) algorithm available in the kernel-machine library for correctness and computational costs. AO-SVR is a modification of an algorithm by Cauwenberghs and Poggio [31] of the support vector machine (SVM), which implements a classifying accurate-online support vector machine. The same experiment will be executed using different implementations capable of support vector machine regression. The implementations chosen are also mentioned in the introduction of this chapter: LibSVM, SVM^{light} and SVM*Torch*. I conducted experiments with the implementations on two regression problems.

The first data set is one made available through the SVM*Torch* package to test the SVM in regression mode. This is a data set containing 5000 patterns of 12 attributes. Table 5.1 shows the running times and several statistics of the found solutions of the different SVM implementations.

TABLE 5.1: Comparative performance of online SVM regression versus batch implementations (5000 data points).

Tool	Algorithm	Fit time(s)	SVs	SVs at C	SVs < C
LibSVM	Batch [32]	2.0	597	585	12
SVM^{light}	Batch [66]	2.6	597	585	12
SVM*Torch*	Batch [36]	0.4	596	585	11
KML	Online [84]	2.9	597	585	13

TABLE 5.2: Comparative computation times of SVM regression on the cpusmall data set (8192 data points).

Parameters			LibSVM [32]		SVM^{light} [66]		KML [84]	
C	ϵ	σ	Time (s)	SVs	Time (s)	SVs	Time (s)	SVs
10.0	1	2.5	11.1	5851	40.3	5849	65.5	5849
20.0	1	2.5	8.2	4197	28.4	4196	44.6	4198
10.0	2	3.5	8.4	4052	26.4	4051	33.4	4049
10.0	3	3.5	6.5	2903	19.6	2902	24.6	2900

The second regression task is the *cpusmall* data set availble from the authors of LibSVM, which is a regression problem of 8192 patterns of 12 attributes. The results of the second experiment are shown in table 5.2. The results show that SVM*Torch* and LibSVM are the fastest implementations of the support vector machine, and that on-line implementation of the kernel-machine library is slowest.

It should be noted that the direct comparison of computing times as shown in tables 5.1 and 5.2 is questionable. Batch algorithms have the complete data set as an input, whereas an on-line algorithm performs *as many fits as there are data points*. Indeed, the results should not change considerably, but I could have compared the training times of all batch algorithms on several thousands of data sets. For instance, when performing a cross-over validation test, in which a series of data sets have to be trained, the result will be in favour of the on-line algorithm with respect to the computing costs. The on-line algorithm is particularly slow if a large part of the data consists of support vectors. One of the steps in the algorithm is (for each new point) to evaluate the support vector machine for that given point: i.e., when adding the last few points, a lot of kernel evaluations take place. This takes a large amount of time.

At the time of writing, the sequential minimisation optimisation (SMO, [102]) batch algorithm is being contributed to the kernel-machine library.

"In theory, there is no difference between theory and practice. But, in practice, there is."

Jan van de Snepscheut (1953 - 1994)

6 Experimental Results

Experiments need data sets and a kernel machine to run. An overview of the gathered data used for the electricity demand model is presented in subsection 4.3.1, those used for the short-term wind-power production model in subsection 4.4.1. To create the optimally scaled models, I will use cross-validation techniques and visual inspection. Correlation analyses between features, the electricity load patterns, and the kernel machine will assist in feature selection, as discussed in section 4.2. A kernel machine from the kernel-machine library (as introduced in chapter 5) will be used to perform experiments with.

At the time of writing, the electricity-demand model presented in this thesis has been in daily use for over a year. In section 6.1 I will present real-life performance benchmarks of the different electricity-demand models during one month. Due to the competition on the energy market, certain information concerning these experiments will not be disclosed. The experiments that have been performed to obtain my electricity demand model are discussed in section 6.2. Section 6.3 deals with the short-term wind-power production forecasting.

6.1 Electricity-demand Forecasting in Practice

In practice, predictions can be presented to end users by several electricity demand forecasting systems. Guided by expertise and experience the end user will select and combine elements of these predictions to form the final forecast. These final forecasts will be incorporated in the E-program, that will be submitted to Tennet (see subsection 1.1.1).

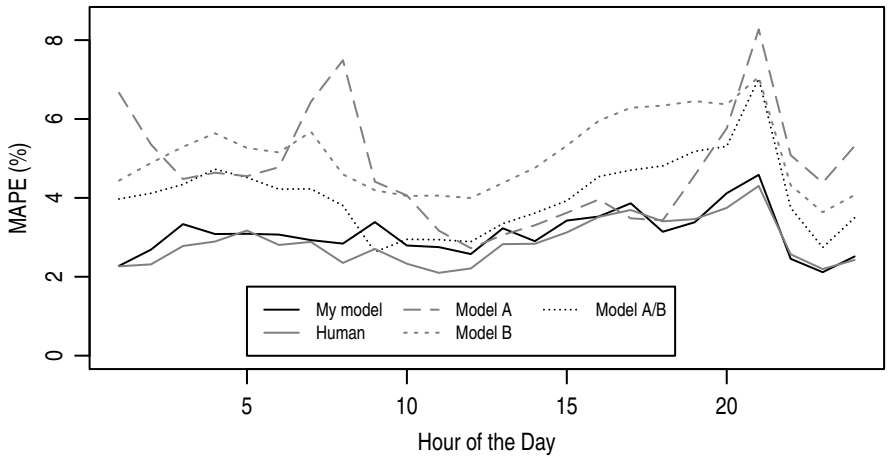

FIGURE 6.1: Comparative average performance per hour of the day during one month.

The data used for the comparative benchmarks presented here were gathered during a full month. I will compare the predictions made by four models and the human end-user. Two commercial electricity-demand models have been used, which I will call model A and model B. These two commercial models have been combined by a certain method to obtain the third model A/B. My model is also measured.

Figure 6.1 shows the mean absolute percentage error (MAPE, see section 2.4) of all predictive models and the human end-user for each different hour of the day. The figure shows that model A/B successfully combines model A and model B, its performance is better for the larger part of the day. Next to the human user, my model clearly has the lowest MAPE throughout the day, except for 9 AM. During a large part of the day, my model is at least 50% more accurate than both model A and B. During the month of measuring, models show most difficulty with the prediction of the electricity demands near 21 hours.

Table 6.1 shows the average MAPE of the whole period that is measured. The first column shows the model name, followed by the MAPE error and its and the standard deviation, and the second column shows the relative MAPE expressed in terms of my model. Model B has the highest MAPE, and of all models, my model has the lowest MAPE. Expressed in relative terms, my model was 30% better than the combination of models A and B. The combination of all models and manual adjustments by the human end-user has a competitive edge of 7% over my model.

The standard deviations of the MAPE shown in table 6.1 also appear to differ from model to model. To illustrate the difference in consistency or robustness

TABLE 6.1: Comparative performance of different models of the real-life data.

Model	MAPE	Relative MAPE
Model A	4.7±4.5	152%
Model B	5.1±4.1	165%
Model A/B	4.1±3.8	131%
Human	2.9±2.3	93%
My model	3.1±2.4	100%

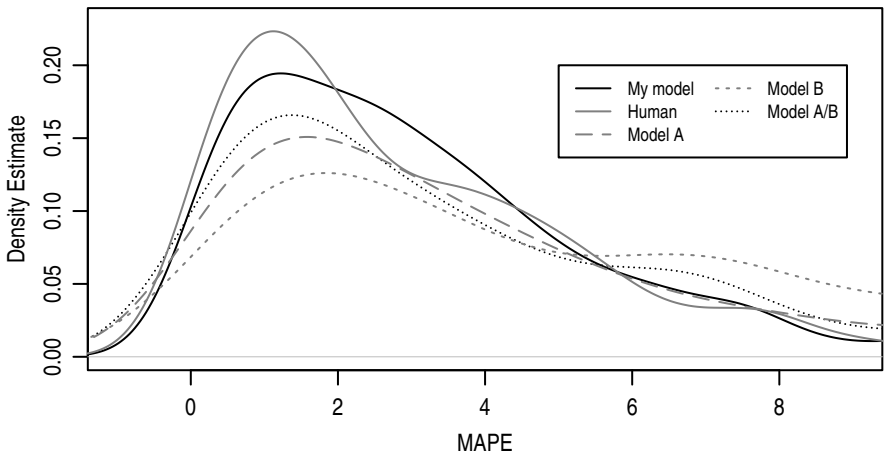

FIGURE 6.2: Estimated probability densities of the error distribution of several electricity demand prediction models on real-life data.

of the different models, a kernel density estimate (which has nothing to do with kernel machines) is depicted in figure 6.2. The total area of my model in the domain [0,8] shown is largest together with that of the human. It implies that the estimated probability density will practically disappear outside the graph, this in contrast to the other models shown.

6.2 Electricity-demand Model Experiments

For the experiments I will use a data set measured at the province North Holland in the Netherlands, consisting of historical hourly electricity demand measurements done during 1996, 1997 and 1998. Also for the same period, weather and holiday related measurements are included as discussed in subsection 4.3.1.

TABLE 6.2: Calendar component compositions.

Time of day	Day type	Train	Test
linear	binary	7.01	6.97
polar	binary	5.81	5.86
linear	polar	6.04	6.13
polar	polar	5.75	5.81

I have used a randomly chosen 60-40 train-test set split, which corresponds with 15770 samples for the training set, and 10514 samples for the test set. For the test data points, we will have to use the actual weather measurements as if they were known in advance. For a real-life case numerical weather predictions will have to be provided. We will use the mean absolute percentage error (MAPE, see (2.8) on page 20). To minimise the MAPE error, I have taken the log of the electricity load values.

I have implemented the iterative system shown in figure 4.6 using a combination of a script written in R [64], and a kernel machine from the kernel-machine library. The following sections discuss the input variables in an incremental manner, i.e., subsection 6.2.1 starts with a system containing one component only, and subsection 6.2.4 discusses the full model containing all components.

6.2.1 Calendar Component

Time of day and day type express the calendar related variables, therefore one component containing both is used.

Time of Day and Day Type

In this experiment we try different types to represent the time of day and day type: for time of day we investigate the linear and polar representations, and for day type, we will test the binary and polar representations. Public holidays are set to Sunday. Table 6.2 shows that the polar time of day representation and the polar day type decrease the error in all comparative cases.

Time of Year

In this experiment, we compare the different ways of representing the time of year part in the calendar component. A linear type (day of the year), the Sun's altitude and a polar yearly data type have been tested. Table 6.3 shows the train and test errors of the different representations. In all cases, the twilight indicator lowered all errors. Although the polar without twilight indicator has the lowest

TABLE 6.3: Comparative performance of time-of-year representations.

Time of year	Train	Test
without time-of-year	5.75	5.81
linear	3.64	3.76
linear and twilight	3.51	3.65
altitude	3.42	3.51
altitude and twilight	3.28	3.39
polar	3.16	3.46
polar and twilight	3.20	3.30

TABLE 6.4: Added trend component.

	Train	Test
without trend	3.20	3.30
with trend	2.64	2.75

train error, its test error is not lowest. The polar time of year representation in combination with the twilight indicator performs best.

6.2.2 Trend Component

The residuals show a structural slide of a prediction that is too high to a prediction that is too low. On further inspection, the North Holland data exhibit a long-term growth pattern. This long-term pattern can be modelled with a separate trend component. To keep the complexity of this component to a minimum, let us assume a steady exponential growth pattern. Table 6.4 shows the train and the test error after applying three iterations of fitting.

6.2.3 Weather Components

We created a separate component for each weather related variable, i.e., a temperature, a radiation and a wind component. To compare the different input spaces for these components, the MAPE errors have been recorded after performing three iterations.

Temperature

In order to achieve accurate results, all components are reset every time a new iteration begins. Also, the exponential moving average (EMA, [103]) of the tem-

TABLE 6.5: Temperature component configurations.

Temperature component	train	test
temperature only	2.25	2.35
time of day	2.21	2.31
EMA	2.26	2.35
EMA and time of day	2.00	2.09

perature is tested as an alternative in the input space. Table 6.5 shows the results of different input space compositions for the temperature component.

Radiation

The radiation component is designed to reflect the distance correctness as well. I.e. during night, the radiation is always equal to zero. Hence, during the night, this component should not add any distance. Therefore, the theoretical altitude of the Sun is determined on which measurements of the radiation above zero may be expected, which is set at -8 degrees. A binary decision forces the points of the radiation component to exactly the same points in feature space.

Wind

The wind component is very straightforward. We use the measured wind speed only to model the changes in electricity demand patterns because of changes in wind speeds.

The top row in table 6.6 shows the resulting model quality when using all weather components after three iterations. Figure 6.3 shows the component predictions for a period of two weeks of the North Holland data.

6.2.4 Additional Information

This experiment serves as an example of adding information that is found on the basis of model performance quality in previous sections. In this case, we have added school holiday information and information about the construction holidays to the calendar component. Table 6.6 shows the impact of adding such information.

Further analysis of the maximum errors made by the model reveals that December 30th shows unusual large errors in all years. December 5th, 1996, my prediction goes astray. It seems that Sinterklaas has influence after all. September 3rd, 1998, another prediction error was made that was larger than usual.

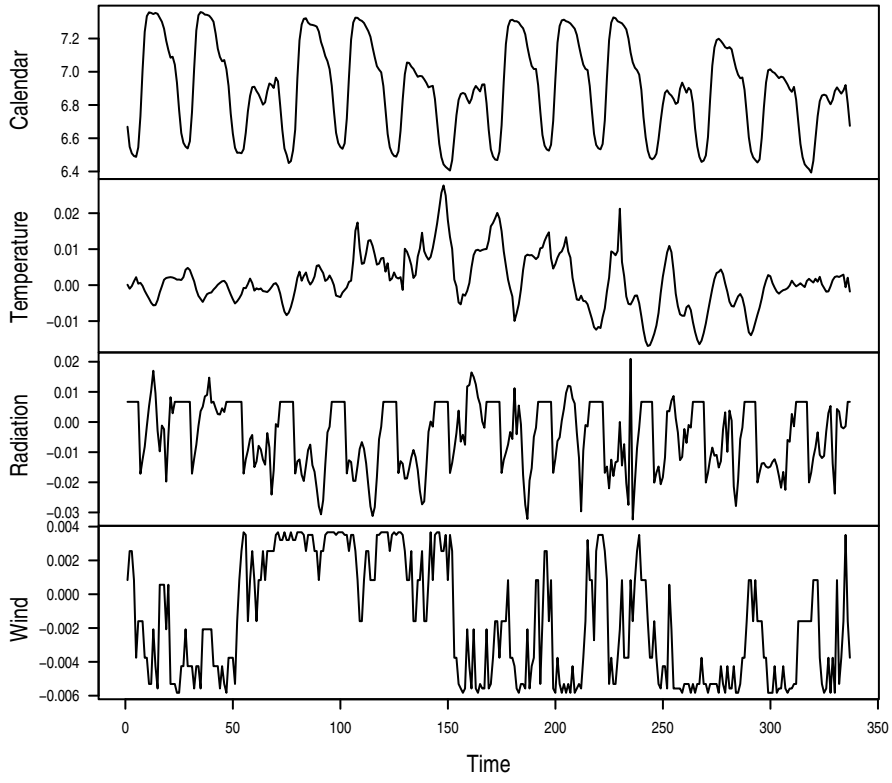

FIGURE 6.3: A display of the different component predictions for a period of two weeks.

TABLE 6.6: Holiday information.

	train	test
without holidays	1.95	2.04
with holidays	1.74	1.84

TABLE 6.7: Error measured over fitted wind-power production models, no wind direction.

added station	RMSE	MAE	MAX	Cor
Stavoren	30.78	22.68	174.57	0.886
De Kooy	21.83	16.08	137.41	0.945
Amsterdam	19.39	14.37	109.07	0.957
Valkenburg	18.96	14.11	100.39	0.959

TABLE 6.8: Error measured over fitted models, including wind directions.

added station	RMSE	MAE	MAX	Cor
Stavoren	28.07	20.55	191.78	0.907
De Kooy	21.33	15.74	137.53	0.947
Amsterdam	18.97	14.08	98.65	0.959
Valkenburg	**18.64**	**13.86**	**93.97**	**0.960**

The weather news archive[1] reveals that September 1998 was very wet: a record was set on September 3rd, 1998, when the second heaviest cloudburst of the 20th century occurred at Boskoop, pouring 92 mm in only 2 hours.

6.3 Short-term Wind-power Production Forecasting

To select the features to be used by the wind-energy prediction system, I will conduct and evaluate several experiments with the wind-power production data. I have initialised a kernel machine in combination with a Gaussian kernel with parameter $\sigma = 1$. The wind speeds are scaled with a factor of 0.05. The diameter of the wind direction circle is set to 0.5. I have tested with cross-validation that these settings were giving acceptable results. Table 6.7 shows the results of fitting with different feature spaces, without taking the wind direction into account. Table 6.8 shows the results of the same feature space, but with the wind direction being taken into account. The error measures used were root-mean-square of the error, the mean absolute error, the maximum absolute error and the statistical correlation.

Common benchmarks in determining the quality of the wind-power prediction are the persistence model and the mean-production model. When using persistence, one takes the latest available measured value(s) as the prediction for the next value(s). The persistence is commonly the model to beat [54].

[1] See `http://www.knmi.nl/voorl/nader/september1998warmenzeernat.htm`.

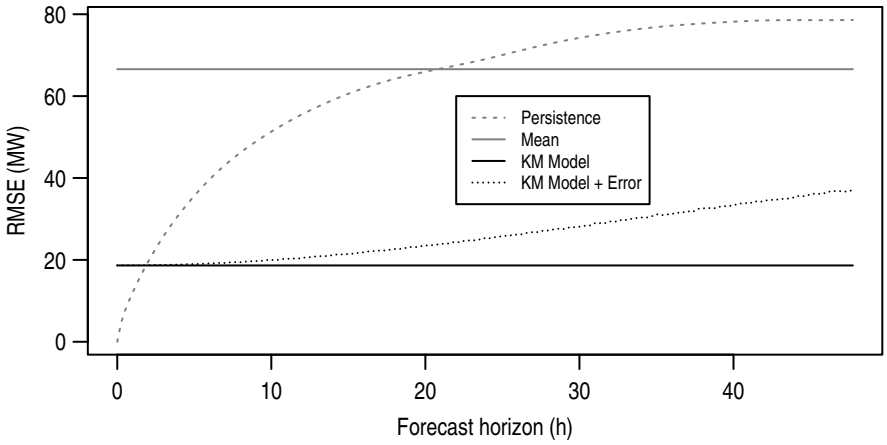

FIGURE 6.4: Root-mean-square of the error of different models.

The mean-production model simply reproduces the mean of the production at all times. I have used the kernel-machine model using an input space of four weather stations and the wind directions, of which the fitting errors are shown in bold on the bottom row of table 6.8.

Because we did not have available the historically predicted wind speeds and directions, we simulated an error development in the numerical weather forecasts. To do so, we have corrupted the wind speed measurements for each forecast horizon step with Gaussian multiplicative noise,

$$w_c(t+i) = w_m(t+i)\mathcal{N}(1, c \cdot i)$$

with w_m being the measured wind speed at time t, i being the number of hours ahead, and with c a constant indicating the severity of error development. During experiments we have set c to 1/120.

Figure 6.4 shows the root-mean-square of the errors of four different models: the persistence model, the mean of the production measurements, the kernel-machine model, and a kernel-machine model with corrupted wind measurement data. It illustrates that the forecast time horizon does not affect the mean-load and kernel machine model. Persistence is the best model for approximately the first two hours, after which the kernel-machine model has the lowest error. The error caused by the multiplicative noise on the wind speeds does not cause dramatic increases in error for the first 24 hours, but ends up nearly doubled at the end of the forecast horizon of 48 hours.

TABLE 6.9: Performance of a 3-layer neural network with a variable number of neurons. Training is stopped when the change in the log of the RMSE of the *test* error is smaller than 1e-4. This table reports the error over *all* data.

Neurons in layer 1	Epochs	RMSE	MAE	Max	Cor
3	3470	20.4	15.1	95.5	0.952
4	2915	20.4	15.2	95.7	0.952
5	2725	20.4	15.2	95.9	0.952
6	2400	20.4	15.0	96.1	0.952
7	2080	20.4	15.0	96.4	0.952
8	1930	20.6	15.2	95.6	0.951
9	1795	20.6	15.3	96.0	0.951
10	1570	20.6	15.2	96.4	0.951
15	1295	20.7	15.3	97.0	0.951
20	1010	20.7	15.2	95.5	0.951
100	375	20.9	15.4	100.9	0.950

6.3.1 Neural Networks for Wind-Power Prediction

To test the differences between applying an artificial neural network and a kernel machine, some experiments have been performed on the wind-power production data. Besides the mapping of the wind directions on a circle, no special kind of transformation specific to the Gaussian kernel has been applied. I have assumed it is reasonable to test an ANN on the same data. I have used the FANN bindings available in the kernel-machine library, described in subsection 5.3.3.

So, I started with a basic feed-forward neural network using the R-Prop training algorithm (an algorithm more advanced than the back-propagation algorithm). I have implemented an automatic stopping criterion. The data are split in a 50% training set and a 50% test set. The neural network is trained for 5 epochs. If the test error stabilises, training is terminated. The total error over all the data is measured and reported. Table 6.9 shows the results of a line search for the optimum number of neurons. During this line search, the following observations have been made.

- The neural network gives different answers each time: all solutions found are *local* solutions. Several kernel machines exist that deliver identical answers each time, i.e., global solutions (e.g. the support vector machine has this property, as illustrated by tables 5.1 and 5.2). Neural networks are often trained multiple times, and the best neural network is stored. The neural network finds a different solution every time it is trained. In some cases the network completely failed to converge to an acceptable point.

TABLE 6.10: Performance of the neural network with several different layer configurations.

Network config	Epochs	RMSE	MAE	Max	Cor
6-5-4-1	1170	20.7	15.7	94.4	0.950
6-6-5-1	950	20.6	15.6	96.4	0.951
6-7-6-1	920	20.7	15.6	97.3	0.951
6-21-21-1	800	20.9	15.6	99.8	0.950
6-5-4-3-1	600	20.7	15.8	94.6	0.951
6-12-24-48-1	875	20.7	15.8	98.4	0.951
6-33-22-11-1	350	20.7	15.8	102.7	0.951
6-5-4-3-2-1	**825**	**20.1**	**15.0**	**99.1**	**0.953**
6-11-9-7-5-1	1755	20.3	15.5	95.5	0.952

TABLE 6.11: Best performing ANN versus the best performing kernel machine.

Regression Technique	RMSE	MAE	MAX	Cor
Kernel Machine	18.6	13.9	94.0	0.960
Neural Network	20.1	15.0	99.1	0.953

This could be because neural networks are initialised with random weights.

- In all cases, an explicit split has to be made between train and test sets to determine when to stop training.
- All experiments have been performed within acceptable times; i.e. less than 10 minutes for all experiments.

After the line-search to a 3-layer neural network, an exploratory search has been done to find a better neural network architecture. Table 6.10 shows the result of different multilayer architectures. It can be noted that neural networks can deliver a wind-power prediction system.

Table 6.11 shows the comparative performance of the best kernel machine and the best artificial neural network (as shown in bold in tables 6.8 and 6.10). However, in not one single case, the neural network has been able to achieve the same quality of predictive model as the kernel machine. Besides this, the amount of human effort and expertise that has to be spent to obtain consistent qualitative models is much higher than that required for the kernel machine.

"All truths are easy to understand once they are discovered; the point is to discover them."

Galileo Galilei (1564–1642)

7 Conclusions

Kernel machines provide the current state-of-the-art methods to obtain predictive models on the basis of measurements. In this work they are used to obtain a model describing electricity demands, and also to obtain a model describing wind-power production.

To this end, I have introduced a concept from functional data analysis to kernel machines to promote smoothness in output space in three steps. First, I have established derivatives of kernel machines by introducing derivative kernels. Second, I have introduced kernel roughness penalties to promote smoothness in output space. Third, I have developed the smooth relevance vector machine (SRVM) which successfully combines kernel roughness penalties with automatic relevance determination by using a novel Bayesian prior. Experiments in section 3.3 elicited that smoothness in output space increases the quality of fit, which holds even more for higher derivative orders. The quality of the approximation to the underlying function is empirically better.

I designed and developed the kernel-machine library, which enables flexible and efficient cross-platform development of kernel machines. It is open for further extensions, but it can be said now already that it implements several state-of-the-art algorithms. An experiment has been reproduced to affirm that the fast relevance vector machine is magnitudes faster than the classic algorithm for the relevance vector machine. Another experiment showed that the on-line algorithm for the support-vector machine is not faster than batch algorithms, especially in case of many support vectors.

The flexibility of kernel machines allows design of input-space representa-

tions specific to the knowledge domain of electricity-demand modelling. In contrast to the in this domain commonly used 24-hour day-type representations, I propose that a day type should last 48 hours. A novel representation is introduced to represent day types so as to ensure smooth day-to-day curve-connections in terms of electricity loads. A twilight indicator is introduced to keep the system aware of lights being switched on or off. Experiments showed that the quality of the electricity-demand model has improved significantly by using these specially designed input variables.

The use of several kernel machines to increase orthogonality of the electricity demand model has shown to perform well, and what perhaps is even more important, using multiple components increases the transparency of the electricity-demand model. As a result, electricity demands can be predicted with a high degree of accuracy and transparency. Detailed patterns produced by the components of the electricity demand model lead to explicit knowledge about the behavioural aspects of electricity demands. A variety of circumstances hidden in the data appears to be correctly mined, e.g. long-term growth, Sinterklaas, and a cloudburst. For an end user, explicit explanations by the model are very valuable information. These explanations can also be used to improve the electricity demand model by adding more specific event knowledge. The electricity demand model can back-cast historic electricity demand patterns for other hypothetical weather scenarios.

I have successfully combined discretised wind-speed predictions obtained from several weather stations to form an accurate high-resolution curve for wind power. A large improvement is obtained by using more than one weather station. The added wind direction successfully discriminates against wind from different directions, as it lowers the error made by the wind-power production model. Although the model performs well, the quality of the numerical wind forecasts remains to have a large influence on the quality of the wind-power predictions. The excellent non-linear approximation capabilities of kernel machines provide good mechanisms to create a short-term wind-power forecasting system. Such a system can also be created with an artificial neural network. However, a great deal of time is then consumed by the training process alone, which makes the design of such a system much less focused on the real results.

Decrease in model prediction errors, if used in Nuon's practice, could result in a reduction of its imbalance, and explicitness of the knowledge would ensure durability of this reduction.

Summarising, based on the research reported in this thesis, I have concluded the following.

- The introduction of derivative kernels, roughness penalties, and a novel Bayesian prior successfully combines sparseness and smoothness. Experiments showed that it can improve model quality.

- The kernel-machine library provides a flexible and efficient base for imple-

menting kernel machines. It is open-source, and has a growing user base.

- The predictive error of the electricity-demand model has been reduced by introduction of domain-specific representations. A 48-hour day-type representation is better than a 24-hour day-type representation. A twilight indicator successfully emphasises switching on and off of lights.
- My electricity-demand model shows competitive performance compared with a number of commercial offerings.
- Kernel machines successfully combine weather data from several sources to deliver an accurate wind-power production forecast. When compared with neural networks, kernel machines are easier to use, and result in stable, more accurate solutions.

7.1 Future Work

Future research should include possibilities for other kernel methods to utilise derivative kernels or roughness penalties. An interesting open theoretical issue is the maximum derivative order on which a signal may be expected, given a data set.

Electricity-demand models could be improved by using more region-specific data and a setup for enhanced handling of the measuring-correction factor. Also, using more weather components such as a precipitation component could structurally reduce the error. Future work in the area of wind-power prediction should include involvement of more weather stations and even more weather related variables such as air pressure, humidity, and wind direction at each weather station. Higher moments such as variance in the numeric weather forecasts should be taken into account. A probabilistic type of kernel machine could give the advantage of estimated confidence intervals of the forecasted wind-power production.

References

[1] Ajith Abraham and Baikunth Nath. A neuro-fuzzy approach for modelling electricity demand in Victoria. *Applied Soft Computing Journal*, 1(2):127–138, August 2001. ISSN 1568-4946. URL http://www-mugc.cc.monash.edu.au/~abrahamp/asc.pdf.

[2] Karin Adelmund. Regeling spreiding zomervakanties 2003–2005. *Gele katern*, 7(15), June 2001. URL http://www.minocw.nl/onderwijs/vakanties/regsprzomervak.pdf.

[3] Mark Aizerman, Emmanuil Braverman, and Lev Rozonoèr. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[4] HM Al-Hamadi and SA Soliman. Short-term electric load forecasting based on kalman filtering algorithm with moving window weather and load model. *Electric Power Systems Research*, 68:47–59, 2004. ISSN 0378-7796.

[5] Robert Andrews, Joachim Diederich, and Alan Tickle. A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge Based Systems*, 8:373–389, 1995. ISSN 0950-7051. URL http://sky.fit.qut.edu.au/~andrewsr/papers/KBSSurvey.ps.

[6] Jie Bao. Short-term load forecasting based on neural network and moving average. Technical report, Iowa State University, 2002. URL http://www.public.iastate.edu/~baojie/pub/2002-05-08_stlf.pdf.

[7] Maumita Bhattacharya, Ajith Abraham, and Baikunth Nath. A linear genetic programming approach for modeling electricity demand prediction in Victoria. In Ajith Abraham and Mario Koppen, editors, *Hybrid Information Systems*, Advances in Soft Computing, pages 379–394, Berlin, Germany, November 2001. Springer-Verlag. ISBN 3-7908-1480-6. URL http://www-mugc.cc.monash.edu.au/~abrahamp/172.pdf.

[8] Zvi Boger. Electricity load forecasting using artificial neural networks clustering. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL http://neuron.tuke.sk/competition/reports/zviboger.pdf.

[9] Gianluca Bontempi. EUNITE world-wide competition: Research report. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL `http://neuron.tuke.sk/competition/reports/gianlucabontempi.pdf`.

[10] George Boole. *An Investigation of The Laws of Thought*. Walton and Maberley, London, 1854. ISBN 0-486-60028-9. Reprinted by Dover Publications, New York, 1951.

[11] Rutger ter Borg. Extracting fuzzy rules using genetic programming. Master's thesis, Delft University of Technology, Delft, the Netherlands, August 2000. URL `ftp://ftp.kbs.twi.tudelft.nl/pub/docs/MSc/all/Borg_Rutger_ter/thesis.ps.gz`.

[12] Rutger ter Borg and Léon Rothkrantz. Graphed evolutionary computing. In John Caulfield, Shu-Heng Chen, Heng-Da Cheng, Richard Duro, Vasant Honavar, Etienne Kerre, Mi Lu, Manuel Grana Romay, Timothy Shih, Dan Ventura, Paul Wang, and Yuanyuan Yang, editors, *Proceedings of the 6th Joint Conference on Information Science, March 8-13, 2002, Research Triangle Park, North Carolina, USA*, pages 606–609. JCIS / Association for Intelligent Machinery, Inc., 2002. ISBN 0-9707890-1-7.

[13] Rutger ter Borg and Léon Rothkrantz. Smooth bayesian kernel machines. In Wlodzislaw Duch et al., editor, *Proceedings of the 15th International Conference on Artificial Neural Networks (ICANN'05)*, volume 3697 of *Lecture Notes in Computer Science*, pages 577–582. Springer-Verlag, 2005. ISBN 3-540-28755-8. URL `http://www.ibspan.waw.pl/ICANN-2005/`.

[14] Rutger ter Borg and Léon Rothkrantz. Modelling electricity load with kernel machines. *IEEE Transactions on Power Systems*, submitted, 2005. ISSN 0885-8950.

[15] Rutger ter Borg and Léon Rothkrantz. Short-term wind power prediction with radial basis functions. *Neural Network World*, submitted, 2005.

[16] Svetlana Borovkova. *Estimation and Prediction for Nonlinear Time Series*. PhD thesis, University of Groningen, 1998. URL `http://www.ub.rug.nl/eldoc/dis/science/s.a.borovkova/`.

[17] Adrian Bors and Ioannis Pitas. Median radial basis function neural network. *IEEE Transactions on Neural Networks*, 7(6):1351–1364, 1996. ISSN 1045-9227. URL `http://www-users.cs.york.ac.uk/~adrian/Papers/Journals/TNN96.pdf`.

[18] Bernhard Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory (COLT'92)*, pages 144–152, Pittsburgh, 1992. ACM Press. URL `http://www.clopinet.com/isabelle/Papers/colt92.ps.Z`.

[19] Philip Brierley and Bill Batty. Electric load modelling with neural networks: An insight into the black box. In *Proceedings of the 4th International Conference on*

Neural Information Processing and Intelligent Information Systems (ICONIP'97), volume 2, pages 1326–1329, Singapore, 1997. Springer-Verlag. URL `http://www.cranfield.ac.uk/public/me/fo941992/papers/iconip97.ps`.

[20] Philip Brierley and Bill Batty. Neural data mining and modelling for electric load prediction. In Abhay Bulsari, Fernandez de Canete, and Sirpa Kallio, editors, *Proceedings of the 4th International Conference on Engineering Applications of Neural Networks (EANN'98)*, Turku, Finland, 1998. Systems Engineering Association. ISBN 951-97868-0-5. URL `http://www.cranfield.ac.uk/public/me/fo941992/papers/eann98.ps`.

[21] Werner Brockmann and Steffen Kuthe. Different models to forecast electricity loads. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL `http://neuron.tuke.sk/competition/reports/wernerbrockmann.ps`.

[22] Christopher Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998. ISSN 1384-5810. URL `http://citeseer.nj.nec.com/burges98tutorial.html`.

[23] Francesco Camastra and Anna Maria Colla. Short-term load forecasting based on correlation dimension estimation and neural nets. In Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud, editors, *Proceedings of the 7th International Conference on Artificial Neural Networks (ICANN'97)*, volume 1327 of *Lecture Notes in Computer Science*, pages 1035–1040. Springer-Verlag, 1997. ISBN 3-540-63631-5. URL `ftp://ftp.disi.unige.it/person/camastraf/icann97.ps`.

[24] Francesco Camastra and Anna Maria Colla. Neural short-term prediction based on dynamics reconstruction. *Neural Processing Letters*, 9(1):45–52, 1999. ISSN 1370-4621. URL `ftp://ftp.disi.unige.it/person/camastraf/npl99.ps`.

[25] Bradley Carlin and Thomas Louis. *Bayes and Emperical Bayes Methods for Data Analysis*. Chapman & Hall/CRC, New York, second edition, 2000. ISBN 58488-170-4.

[26] Otávio Carpinteiro and Alexandre Alves da Silva. A hierarchical self-organising map model in short-term load forecasting. In *Proceedings of the 5th International Conference on Engineering Applications of Neural Networks (EANN'99)*, pages 75–80, 1999. ISBN 83-7174-512-512-5. URL `http://www.iee.efei.br/~otavio/tmp/eann99.pdf`.

[27] Otávio Carpinteiro and Alexandre Alves da Silva. A hierarchical self-organising map model in short-term load forecasting. *Journal of Intelligent and Robotic Systems*, 31:105–113, 2001. ISSN 0921-0296. URL `http://gerson.iee.efei.br/~otavio/tmp/jirs.pdf`.

[28] Martin Casdagli. Nonlinear prediction of chaotic time series. *Physica D: Nonlinear Phenomena*, 35(3):335–356, 1989.

[29] Enrique Castillo, Angel Cobo, José Manuel Gutiérrez, and Rosa Eva Pruneda. *An Introduction to Functional Networks with Applications. A Neural-Based Paradigm*, volume 473 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, 1998. ISBN 0-7923-8332-X. URL http://www.wkap.nl/prod/b/0-7923-8332-X.

[30] Enrique Castillo, Bertha Guijarro, and Ampara Alonso. Electricity load forecasting using functional networks. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL http://neuron.tuke.sk/competition/reports/berthaguijarro.pdf.

[31] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In Todd Leen, Thomas Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems (NIPS'00)*, volume 13, pages 409–415, Cambridge, Massachusetts, USA, April 2001. The MIT Press. ISBN 0-262-12241-3. URL http://books.nips.cc/papers/files/nips13/CauwenberghsPoggio.pdf.

[32] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. URL http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[33] Ming-Wei Chang, Bo-Juen Chen, and Chih-Jen Lin. EUNITE network competition: Electricity load forecasting. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL http://neuron.tuke.sk/competition/reports/chih-jenlin.ps.

[34] Guenjun Chen, Kai-kwong Li, Tak-shing Chung, Hongbin Sun, and Guoqing Tang. Application of an innovative combined forecasting method in power system load forecasting. *Electric Power Systems Research*, 59(2):131–137, 2001. ISSN 0378-7796.

[35] Hong Chen, Claudio Cañizares, and Ajit Singh. Ann-based short-term load forecasting in electricity markets. *IEEE Transactions on Power Systems*, 2001. ISSN 0885-8950.

[36] Roman Collobert and Samy Bengio. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001. ISSN 1533-7928. URL http://www.ai.mit.edu/projects/jmlr/papers/volume1/collobert01a/collobert01a.pdf.

[37] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines (and other kernel-based learning methods)*. Cambridge Univerisity Press, Cambridge, UK, 2000. ISBN 0-521-78019-5. URL http://www.support-vector.net.

[38] Thomas Czernichow, Alain Germond, and Bernadette Dorizzi. Improving recurrent network load forecasting. In *Proceedings of the 1995 IEEE International Conference on Neural Networks (ICNN'95)*, Piscataway, USA, 1995. IEEE Press. ISBN 0-

7803-2769-1. URL http://www-sim.int-evry.fr/publications/czernichow/noteedf.ps.gz.

[39] Charles Darwin. On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life. *Journal of researches during H.M.S. Beagle's Voyage round the world*, 1859.

[40] Adam Ding. Neural networks prediction with noisy predictors. *IEEE Transactions on Neural Networks*, 10(5):1196–1203, September 1999. ISSN 1045-9227. URL http://www.math.neu.edu/~ding/ann/paper1.ps.

[41] Harris Drucker, Chris Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In Michael Mozer, Michael Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 155–161, Cambridge, Massachusetts, USA, 1997. The MIT Press. ISBN 0-262-10065-7. URL http://www.kernel-machines.org/papers/druburkausmovap96.ps.gz.

[42] Mohamed El-Sharkawi, Robert Marks, and Mark Damborg. Short term electric load forecasting using an adaptively trained layered perceptron. In Mohamed El-Sharkawi and Robert Marks, editors, *Proceedings of the First International Forum on Applications of Neural Networks to Power Systems*, pages 3–6. Sofitware, 1991. ISBN 0780300653. URL http://cialab.ee.washington.edu/marks-stuff/publications/markspubarchive/1991_shorttermelectric.pdf.

[43] Yaakov Engel, Shie Mannor, and Ron Meir. Sparse online greedy support vector regression. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Machine Learning: ECML 2002*, volume 2430 of *Lecture Notes in Computer Science*, Berlin, 2002. Springer-Verlag. ISBN 3-540-44036-4. URL http://www.ee.technion.ac.il/~rmeir/publications/ecml02sogsvr.pdf.

[44] Yaakov Engel, Shie Mannor, and Ron Meir. The kernel recursive least squares algorithm. ICNC03 001, Interdisciplinary Center for Neural Computation, Hebrew University, Jerusalem, Israel, 2003. URL http://visl.technion.ac.il/~yaki/papers/krls-report.pdf.

[45] David Esp. Adaptive logic networks for East Slovakian electrical load forecasting. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL http://neuron.tuke.sk/competition/reports/davidesp.pdf.

[46] Scott Fahlman. An empirical study of learning speed in back-propagation networks. CS 88-162, Carnegie-Mellon University, 1988. URL http://www.cs.cmu.edu/afs/cs.cmu.edu/user/sef/www/publications/qp-tr.ps.

[47] José Fidalgo and João Peças Lopes. Load forecasting dealing with medium voltage network reconfiguration. In Verleysen [126], pages 407–412. ISBN 2-930-30700-5. URL www.dice.ucl.ac.be/proceedings/esann/esannpdf/es2000-34.pdf.

[48] Mário Figueiredo. Adaptive sparseness using Jeffreys prior. In Thomas Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS'01)*, volume 14, Cambridge, Massachusetts, USA, 2002. The MIT Press. ISBN 0-262-04208-8. URL http://www-2.cs.cmu.edu/groups/nips/nips2001/papers/psgz/aa07.ps.gz.

[49] Mário Figueiredo. Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1150–1159, 2003. ISSN 0162-8828. URL http://www.lx.it.pt/~mtf/IEEE_TPAMI_2003.pdf.

[50] Eithne Fitzgerald. Directive 96/61/EC of the European parliament and of the council of 24 September 1996 concerning integrated pollution prevention and control. *EU Official Journal*, L(61):26–40, 10 1996. ISSN 0378-6978.

[51] Nicole Fontaine and Charles Picqué. Directive 2001/77/EC of the European parliament and of the council of 27 September 2001 on the promotion of electricity produced from renewable energy sources in the internal electricity market. *EU Official Journal*, L(283):33–40, 10 2001. ISSN 0378-6978. URL http://europa.eu.int/eur-lex/pri/en/oj/dat/2001/l_283/l_28320011027en00330040.pdf.

[52] Andrew Gelman, John Carlin, Hal Stern, and Donald Rubin. *Bayesian Data Analysis*. Chapman & Hall, London, 1995. ISBN 0-412-03991-5.

[53] Marc Genton. Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research*, 2:299–312, 2001. ISSN 1533-7928. URL http://www.ai.mit.edu/projects/jmlr/papers/volume2/genton01a/genton01a.pdf.

[54] Gregor Giebel, Richard Brownsword, and George Kariniotakis. The state-of-the-art in short-term prediction of wind power: A literature overview. Deliverable report D1.1, Project ANEMOS, Roskilde, Denmark, 2003. URL http://anemos.cma.fr/download/ANEMOS_D1.1_StateOfTheArt_v1.1.pdf.

[55] Peter Green and Bernard Silverman. *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*. Chapman & Hall, 1993. ISBN 0412300400.

[56] Jeff Haberl and Sabaratnam Thamilseran. The great energy predictor shootout II: Measuring retrofit savings. *ASHRAE Journal*, 40(1):49–56, 1998. ISSN 0001-2491.

[57] Klaus Hänsch and Sean Barrett. Directive 96/92/EC of the European parliament and of the council of 19th of December 1996 concerning common rules for the internal market in electricity. *EU Official Journal*, L(27):20–29, January 1997. ISSN 0378-6978. URL http://www.ceer-eu.org/pdf/dir_96_92.pdf.

[58] Robert Hecht-Nielsen. *Neurocomputing*. Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 1990. ISBN 0-201-09255-3.

[59] Henrique Steinherz Hippert, Carlos Eduardo Pedreira, and Reinaldo Castro Souza. Neural networks for short-term load forecasting: A review and evaluation. *IEEE Transactions on Power Systems*, 16(1):44–55, 2001. ISSN 0885-8950.

[60] John Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, Michigan, 1975.

[61] Cheng-Hsiung Hsieh. Grey neural network and its application to short term load forecasting problem. *IEICE Transactions on Information and Systems*, E85-D(5): 897–902, 2002. URL `http://search.ieice.org/2002/pdf/e85-d_5_897.pdf`.

[62] Olaf Huwendiek and Werner Brockmann. Function approximation with decomposed fuzzy systems. *Fuzzy Sets and Systems*, 101(2):273–286, 1999. ISSN 0165-0114.

[63] Boris Igelnik and Yoh-Han Pao. Stochastic choice of radial basis functions in adaptive function approximation and the functional-link net. *IEEE Transactions on Neural Networks*, 6(6):1320–1329, 1995. ISSN 1045-9227.

[64] Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996. ISSN 1061-8600. URL `http://www.r-project.org`.

[65] Gregory Ivakhnenko. Inductive self-organising algorithm for maximum electricity load prediction. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL `http://neuron.tuke.sk/competition/reports/gregoryivakhnenko.pdf`.

[66] Thorsten Joachims. *Making Large-Scale SVM Learning Practical*, chapter 11. In , Schölkopf et al. [113], first edition, 1999. ISBN 0-262-19416-3. URL `http://www.cs.cornell.edu/people/tj/publications/joachims_99a.pdf`.

[67] Jacques de Jong. *Tariff Code*. Dienst uitvoering en Toezicht Energie, the Hague, the Netherlands, December 2001. URL `http://www.nma-dte.nl/nl/besluiten/elektriciteit/dte_codes/tarievencode/tarievencode.pdf`.

[68] Nicolaos Karayiannis. Reformulated radial basis neural networks trained by gradient descent. *IEEE Transactions on Neural Networks*, 10(3):657–671, 1999. ISSN 1045-9227. URL `http://www.egr.uh.edu/ece/faculty/karayiannis/Karayiannis_tnn_10(3)_99.pdf`.

[69] William Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, University of Chicago, 1939.

[70] Michael Kearns and Umesh Vazirani. *An Introduction to Computational Learning Theory*. The MIT Press, Cambridge, Massachusetts, USA, 1994. ISBN 0-262-11193-4. URL `http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=7334`.

[71] Iain King and John Tindle. Storage of half hourly electric metering data and forecasting with artificial neural network technology. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL `http://neuron.tuke.sk/competition/reports/iainking.pdf`.

[72] Daniel Kirschen. Demand-side view of electricity markets. *IEEE Transactions on Power Systems*, 18(2):520–527, 2003. ISSN 0885-8950.

[73] Teuvo Kohonen. Self-organized formation of topographically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[74] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *Proceedings of the 13th International Conference on Machine Learning*, pages 284–292, 1996. URL `http://robotics.stanford.edu/~koller/papers/ml96.ps`.

[75] Wojciech Kowalczyk. Averaging and data enrichment: two approaches to electricity load forecasting. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL `http://neuron.tuke.sk/competition/reports/wojtekkowalczyk.pdf`.

[76] John Koza. Hierarchical genetic algorithms operating on populations of computer programs. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, volume 1, pages 768–774, San Mateo, CA, 1989. Morgan Kaufmann. URL `http://www.genetic-programming.com/IJCAI89.ps`.

[77] Harold Kuhn and Albert Tucker. Nonlinear programming. In Jerzy Neyman, editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492. University of California Press, 1951.

[78] Yuh-Jye Lee and Olvi Mangasarian. SSVM: A smooth support vector machine for classification. *Computational Optimization and Applications*, 20(1):5–22, October 2001. ISSN 0926-6003.

[79] Achim Lewandowski, Frank Sandner, and Peter Protzel. Prediction of electricity load by modeling the temperature dependencies. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL `http://neuron.tuke.sk/competition/reports/achimlewandowski.ps`.

[80] Shuhui Li, Donald Wunsch, Edgar O'Hair, and Michael Giesselmann. Using neural networks to estimate wind turbine power generation. *IEEE Transactions on Energy Conversion*, 16(3):276–282, 9 2001. URL `http://www.ece.umr.edu/acil/Publications/JOURNAL/USING_NEURAL_NETWORKS.pdf`.

[81] Shuhui Li, Donald Wunsch, Egard O'Hair, and Michael Giesselmann. Comparative analysis of regression and artificial neural network models for wind turbine power curve estimation. *Journal of Solar Energy Engineering*, 123:327–332, 11 2001. ISSN 0199-6231. URL `http://www.ece.umr.edu/acil/Publications/JOURNAL/ASMEJSEE01.pdf`.

[82] Ahamd Lofti. Application of learning fuzzy inference systems in electricity load forecast. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL `http://neuron.tuke.sk/competition/reports/ahmadlotfi.pdf`.

[83] Anna Lotufo and Carlos Minussi. Electric power systems load forecasting: A survey. In *Proceedings of the IEEE Power Tech '99 Conference*, pages 1–6, 1999. URL http://www.dee.feis.unesp.br/dee/docentes/publicacoes/artigo_anna_buda_99.pdf.

[84] Junshiu Ma, James Theiler, and Simon Perkins. Accurate on-line support vector regression. *Neural Computation*, 15(11):2683–2704, November 2003. ISSN 0899-7667. URL http://nis-www.lanl.gov/~jt/Papers/aosvr-nc.pdf.

[85] David MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992. ISSN 0899-7667. URL http://www.inference.phy.cam.ac.uk/mackay/inter.nc.ps.gz.

[86] David MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(5):698–714, 1992. ISSN 0899-7667. URL http://www.inference.phy.cam.ac.uk/mackay/class.nc.ps.gz.

[87] David MacKay. Information-based objective functions for active data set selection. *Neural Computation*, 4(4):589–603, 1992. ISSN 0899-7667. URL http://www.inference.phy.cam.ac.uk/mackay/selection.nc.ps.gz.

[88] David MacKay. A practical bayesian framework for backprop networks. *Neural Computation*, 4(3):448–472, 1992. ISSN 0899-7667. URL http://www.inference.phy.cam.ac.uk/mackay/backprop.nc.ps.gz.

[89] Ebrahim Mamdani and Seto Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.

[90] Morgan Mangeas, Andreas Weigend, and Corinne Muller. Forecasting electricity demand using nonlinear mixture of experts. In *Proceedings of the World Congress on Neural Networks (WCNN'95)*, volume II, pages 48–53, 1995. URL http://rutcor.rutgers.edu/~amai/aimath98/extended_abstracts/mmangeas.ps.

[91] Francisco Marín, Fransisco García-Lagos, Gonzalo Joya, and Fransisco Sandoval. Peak load forecasting using kohonen classification and intervention analysis. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL http://neuron.tuke.sk/competition/reports/javiermarin.pdf.

[92] Warren McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[93] Nicholas Minorsky. Directional stability and automatically steered bodies. *Journal of American Society of Naval Engineers*, 34:280–309, 1922.

[94] Vladimiro Miranda and Claudio Monteiro. Fuzzy inference applied to spatial load forecasting. In *Proceedings of the IEEE Power Tech '99 Conference*, 1999. URL http://power.inescn.pt/inesc/artigos/bpt99_358_25.pdf.

[95] Luciano Moulin and Alexandre Alves da Silva. Neural network based short-term elecric load forecasting with confidence intervals. In *Proceedings of the 4th Brazilian Conference on Neural Networks (CBRN'99)*, pages 7–12, 1999. URL `http://www.ele.ita.br/cnrn/4cbrn/artigos-4cbrn/4cbrn_002.pdf`.

[96] Sayan Mukherjee, Edgar Osuna, and Federico Girosi. Nonlinear prediction of chaotic time series using support vector machines. In *Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing*, 1997. URL `ftp://ftp.ai.mit.edu/pub/cbcl/nnsp97.ps.gz`.

[97] Fransisco Ortega, María Teresa Rodríguez, César Menéndez, Nieves Roqueñi, Vicente Rodríguez, Veleriano Álvarez, Gemma Martínez, Joaquín Villanueva, and José Manuel Mesa. An hybrid approach to prediction of electric load with mars and kohonen maps. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL `http://neuron.tuke.sk/competition/reports/franciscoortega.pdf`.

[98] Peter Otto. Fuzzy based time series forecasting of electric load. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL `http://neuron.tuke.sk/competition/reports/peterotto.pdf`.

[99] Ilesh Patel and Brian Samuel. Energy market liberalisation in Europe - the case for full liberalisation. A report sponsored by TXU, Caminus, London, UK, February 2000. URL `http://www.txucorp.com/eu/uk/newsroom/pressreleases/uk/txupr21feb02report.pdf`.

[100] Emil Pelikán. Middle-term electric load forecasting by time series decomposition. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL `http://neuron.tuke.sk/competition/reports/emilpelikan.pdf`.

[101] Pierre Pinson, Nils Siebert, and George Kariniotakis. Forecasting of regional wind generation by a dynamic fuzzy-neural networks based upscaling approach. In *Proceedings of the European Wind Energy Conference (EWEC 2003), Madrid, Spain*. EWEA, 2003. URL `http://anemos.cma.fr/download/publications/pub_2003_paper_EWEC03_UpscalingPinson.pdf`.

[102] John Platt. *Fast Training of Support Vector Machines using Sequential Minimal Optimization*, chapter 12. In , Schölkopf et al. [113], first edition, 1999. ISBN 0-262-19416-3. URL `http://research.microsoft.com/~jplatt/smo-book.pdf`.

[103] Martin Pring. *Technical analysis explained: the successful investor's guide to spotting investment trends and turning points*. McGraw-Hill, New York, 1991. ISBN 0-07-051042-3.

[104] Jim Ramsay and Bernard Silverman. *Applied Functional Data Analysis*. Springer-Verlag, 2002. ISBN 0-387-95414-7. URL `http://www.psych.mcgill.ca/faculty/ramsay/appliedfda.html`.

[105] Jim Ramsay and Bernard Silverman. *Functional Data Analysis*. Springer Series in Statistics. Springer-Verlag, New York, 1997. ISBN 0-387-94956-9. URL `http://www.psych.mcgill.ca/faculty/ramsay/fda.html`.

[106] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Artificial Neural Networks*, pages 586–591, 1993.

[107] Fabio Rivieccio. SVM for an electricity load forecasting problem. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL `http://neuron.tuke.sk/competition/reports/fabiorivieccio.ps`.

[108] Frank Rosenblatt. The perceptron: a probablistic model for information storage and retrieval in the brain. *Psychological Review*, 65:386–408, 1958.

[109] Volker Roth. Sparse kernel regressors. In Georg Dorffner, Horst Bischof, and Kurt Hornik, editors, *Proceedings of the 12th International Conference on Artificial Neural Networks (ICANN'01)*, volume 2130 of *Lecture Notes in Computer Science*, pages 339–346. Springer-Verlag, 2001. ISBN 3-540-42486-5. URL `http://www.informatik.uni-bonn.de/~roth/icann_pdf.pdf`.

[110] Elena Savelieva, Alexey Kravetski, Sergey Chernov, Vasiliy Demyanov, Vadim Timonin, Rafael Arutyunyan, Leonid Bolshov, and Mikhail Kanevski. Application of MLP and stochastic simulations for electricity load in Russia. In Verleysen [126], pages 413–418. ISBN 2-930-30700-5. URL `http://www.dice.ucl.ac.be/proceedings/esann/esannpdf/es2000-502.pdf`.

[111] Martin Scheepers et al. Energie markt trends 2001. ECN-P 01-009, Energieonderzoek Centrum Nederland, Petten, the Netherlands, 2001. URL `ftp://ftp.ecn.nl/pub/www/library/report/2001/p01009.pdf`.

[112] Bernhard Schölkopf and Alexander Smola. *Learning with Kernels*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Massachusetts, USA, 2002. ISBN 0-262-19475-9. URL `http://www.learning-with-kernels.org`.

[113] Bernhard Schölkopf, Christhopher Burges, and Alex Smola, editors. *Advances in Kernel Methods - Support Vector Learning*. The MIT Press, Cambridge, Massachusetts, USA, first edition, 1999. ISBN 0-262-19416-3. URL `http://kernel-machines.org/nips97/book.html`.

[114] Tmonobu Senjyu, Hitoshi Takara, Katsumi Uezato, and Toshihisa Funabashi. One-hour-ahead load forecasting using neural network. *IEEE Transactions on Power Systems*, 17(1):113–118, 2002. ISSN 0885-8950.

[115] Colin Cowan Sheng Chen and Peter Grant. Orthogonal least squares larning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302–309, 3 1991. ISSN 1045-9227. URL http://itswww.epfl.ch/~coursnonlin/files/support/OLS_RBF.pdf.

[116] Johan Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999. URL ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/suykens/reports/lssvm_98_72.ps.gz.

[117] Floris Takens. Detecting strange attractors in fluid turbulence. In David Rand and Lai-Sang Young, editors, *Dynamical Systems and Turbulence*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381, Berlin, 1981. Springer-Verlag. ISBN 0-387-11171-9.

[118] Rob Tibshirani. Regression shrinkage and selection via the lasso. *Journal of The Royal Statistical Society: Series B*, 58(1):267–288, 1996. ISSN 1369-7412. URL http://www-stat.stanford.edu/~tibs/lasso/lasso.pdf.

[119] Michael Tipping. The relevance vector machine. In Sara Solla, Todd Leen, and Klaus-Robert Müller, editors, *Advances in Neural Information Processing Systems (NIPS'99)*, volume 12, pages 652–658, Cambridge, Massachusetts, USA, 2000. The MIT Press. ISBN 0-262-19450-3. URL ftp://ftp.research.microsoft.com/users/tipping/rvm_nips.ps.gz.

[120] Michael Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001. ISSN 1533-7928. URL http://www.ai.mit.edu/projects/jmlr/papers/volume1/tipping01a/tipping01a.ps.

[121] Michael Tipping and Anita Faul. Fast marginal likelihood maximisation for sparse bayesian models. In Cristopher Bishop and Brendan Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, January 3–6 2003, Key West, Florida*, 2003. ISBN 0-9727358-0-1. URL ftp://ftp.research.microsoft.com/users/mtipping/fastsbl.ps.gz.

[122] Leslie Valiant. A theory of the learnable. *Communications of the ACM*, pages 1134–1142, 1984. ISSN 0001-0782. URL http://www.cs.toronto.edu/~roweis/csc2515/readings/p1134-valiant.pdf.

[123] Vladimir Vapnik. *Estimation of Dependencies Based on Emperical Data*. Springer series in statistics. Springer-Verlag, New York, 1982. ISBN 0-387-90733-5. Translated from Russian.

[124] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Statistics for Engineering and Information Science. Springer-Verlag, New York, 1995. ISBN 0-387-98780-0. URL http://www.springer.de/cgi/svcat/search_book.pl?isbn=0-387-98780-0.

[125] Vladimir Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998. ISBN 0-471-03003-1. URL http://www.wiley.com/cda/product/0,,0471030031,00.html.

[126] Michel Verleysen, editor. *Proceedings of the 8th European Symposium on Artificial Neural Networks (ESANN'00)*, Brussels, Belgium, 2000. D-Facto. ISBN 2-930-30700-5. URL http://www.dice.ucl.ac.be/esann/proceedings/esann2000/content.htm.

[127] Albert Vreeman. APX weekly report. APX Weekly Report 23, Amsterdam Power Exchange, Amsterdam, the Netherlands, June 2002. URL http://www.apx.nl/marketresults/weekly/pdf/weekly_report_23.pdf.

[128] Grace Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics, 1990. ISBN 0-89871-244-0.

[129] Robert Weizenegger. The prediction of maximum electrical load based on limited historical data using a fuzzy logic based data mining and modelling algorithm. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL http://neuron.tuke.sk/competition/reports/robweizenegger.pdf.

[130] Paul Werbos. *Beyond Regression: New tools for Prediction and Analysis in the Behavioural Sciences*. PhD thesis, Harvard University, 1974.

[131] Jason Weston, André Elisseef, Bernhard Schölkopf, and Michael Tipping. Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3:1439–1461, 2003. ISSN 1533-7928. URL http://jmlr.csail.mit.edu/papers/volume3/weston03a/weston03a.pdf.

[132] Clint Whaley, Antoine Petitet, and Jack Dongarra. Automated empirical optimization of software and the ATLAS project. *Parallel Computing*, 27(1–2):3–35, 2001. URL http://www.netlib.org/lapack/lawns/lawn147.ps.

[133] Hans Wijers and Winnie Sorgdrager. Elektriciteitswet 1998. *Staatsblad*, 1998 (427):1–26, July 1998. ISSN 0920-2064. URL http://www.nma-dte.nl/en/publications/electricityact.pdf.

[134] Hung-Chih Wu and Chan-Nan Lu. A data mining approach for spatial modeling in small area load forecast. *IEEE Transactions on Power Systems*, 17(2):516–521, 2002. ISSN 0885-8950.

[135] Esa Ylipahkala. Estimating the uncertainty of electricity load. Master's thesis, Helsinky University of Technology, 1997. URL http://www.sal.hut.fi/Publications/pdf-files/tyli97.pdf.

[136] Lofti Zadeh. Fuzzy sets. *Information and Control*, 8:338–358, 1965.

[137] Ji Zhu, Saharon Rosset, Trevor Hastie, and Rob Tibshirani. 1-norm support vector machines. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems (NIPS'03)*, volume 16, Cambridge, Massachusetts, USA, 2004. The MIT Press. URL http://books.nips.cc/papers/files/nips16/NIPS2003_AA07.pdf.

[138] Dalibor Živčák. Electricity load forecasting using ANN. In *Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Aachen, Germany, 2001. Verlag Mainz. ISBN 3-89653-916-7. URL `http://neuron.tuke.sk/competition/reports/daliborzivcak.pdf`.

Summary

Ph.D. thesis “Electricity Load Modelling using Computation Intelligence”, by Rutger W. ter Borg.

Chapter 1 introduces my research. As a consequence of the liberalisation of the Dutch electricity market, program responsibility partners may have to pay considerable sums for the settlement of their imbalances. These costs are based on the difference between the estimated and two weeks later allocated electricity offtake. The larger the difference between these two, the higher the imbalance settlement costs, payable to the Dutch transmission system operator Tennet. The objective of this investigation is to develop an accurate model for electricity demands as well as a short-term wind-power-production forecasting system.

Chapter 2 reviews related work from the fields of electricity-demand modelling and short-term wind-power production forecasting. Structures of the most commonly used models are discussed, which in the case of the electricity demand models are causal models, time-series models, and mixed models. Wind-power production models can be classified to be either physical or statistical. Quality criteria to assess the models are also presented.

Chapter 3 treats a novel method in the area of Bayesian kernel machines. I consider the possibility of obtaining a kernel machine that is sparse in feature space and smooth in output space. Smooth in output space implies that the underlying function is assumed to have continuous derivatives up to some order. Smoothness is achieved by applying a roughness penalty, a concept from the area of functional data analysis. Sparseness is taken care of by automatic relevance determination. Both are combined in a Bayesian model, which has been implemented and tested. Test results are presented in this chapter.

Chapter 4 devises models based on kernel machines, one to describe electricity demands, and one to describe wind-power production. Several domain-specific representations have been designed which make use of the flexibility of the kernel machine. The kernels used by kernel machines have a clear interpretation as a distance measure. After a thorough data analysis, I propose representations of input variables that are compatible with the Gaussian kernel specific to electricity demand patterns. I used a multi-component setup of kernel machines to increase the orthogonality between several input variables. The model for short-term wind-power production uses numerical wind speeds and direction as its only inputs. The coarse resolution of one meter per second of the wind speeds

is refined by combining the weather data from several meteorological stations. Additionally, the wind direction is mapped on a circle so it is more compatible with a Gaussian kernel.

Chapter 5 gives an overview of the Kernel-Machine Library, a C++ library I wrote to facilitate the development and embedding of kernel machines, both for academic use and for developing real-world applications. After motivating the purpose of this library between other similar software packages, I set a number of requirements, which have been kept in consideration during the design of the library. Its implementation draws heavily upon features of modern C++ such as template meta-programming to achieve high performance while at the same time offering a comfortable interface. It enables compile-time selection of specialised algorithms, which makes it very efficient. Cross-platform compiling and correctness testing has also been performed.

Chapter 6 practises electricity load modelling with the devised techniques. For the electricity demand model, the multi component structure as presented in chapter 4 is extended in an experimental way. The proposed representations for day types and twilight are tested for their embeddability. The multi-component structure is filled with calendar, trend, temperature, radiation, and wind components. These components enable the electricity demands to be unravelled; several new explicit facts are discovered by using this system. This new explicit knowledge, such as the exact causal relations, is very useful in practice. The created system produces accurate and detailed predictions of the electricity demands. For the short-term wind-power production system, a number of weather stations is added incrementally to be able to select the input space composition.

Chapter 7 concludes on the presented methods, models, tools, and their experimental results when used for electricity load modelling.

Samenvatting

Proefschrift “Het modelleren van elektriciteitsbelasting met behulp van computational intelligence”, door Rutger W. ter Borg.

In hoofdstuk 1 wordt een introductie op mijn onderzoek gegeven. Door de liberalisering van de Nederlandse elektriciteitsmarkt kunnen de daarop opererende programmaverantwoordelijken (PV'ers) veel geld kwijt zijn aan de afwikkeling van hun onbalans. Deze kosten zijn gebaseerd op het verschil tussen het een dag vantevoren ingediende E-programma, waarin de PV'er opgeeft in welke mate hij het elektriciteitsnet verwacht te belasten, en de twee weken later aan hem geallocceerde belasting. Hoe groter het verschil tussen E-programma en door de netbeheerders toegewezen allocatie, des te hoger de onbalansbetalingen aan de landelijke netwerkbeheerder Tennet kunnen uitvallen. Het doel van dit onderzoek is om nauwkeurig voorspellende modellen te ontwikkelen voor zowel het verbruik van elektriciteit als voor korte-termijn voorspellingen van windenergieproductie.

In hoofdstuk 2 wordt de huidige aanpak besproken vanuit de onderzoeksgebieden van het modelleren van het verbruik van elektriciteit en het korte-termijn voorspellen van windenergie-productie. De opzet van de meest gebruikte modellen wordt besproken, in geval van modellen voor het verbruik van elektriciteit zijn dit causale modellen, tijdreeksmodellen en gecombineerde modellen. Wind energie modellen worden geklassificeerd als een fysiek model of als een statistisch model. Kwaliteitscriteria om de modellen te beoordelen worden ook gepresenteerd.

In hoofdstuk 3 wordt een nieuwe methode op het gebied van Bayesiaanse kernel-machines behandeld. Voorts wordt de mogelijkheid besproken om een kernel-machine te verkrijgen die glad is in de uitkomstruimte en leeg in de kenmerkruimte. Glad in de uitkomstruimte houdt in dat verondersteld wordt dat de onderliggende functie continue afgeleiden moet hebben tot op een bepaalde orde. Gladheid wordt verkregen door het toepassen van een straf op de ruwheid, een concept uit het gebied van de functionele data-analyse. Leegheid wordt verkregen door het gebruik van een automatische relevantie-determinant. Beide concepten worden gecombineerd in een Bayesiaans model, dat is geïmplementeerd en getest. Resultaten hiervan worden in dit hoofdstuk gepresenteerd.

In hoofdstuk 4 worden modellen ontworpen op basis van kernel-machines, één om de vraag naar elektriciteit te beschrijven, en één om de windenergie-

productie te beschrijven. Een aantal domeinspecifieke representaties zijn ontworpen die gebruik maken van de flexibiliteit van kernel-machines. De kernels die gebruikt worden door kernel-machines hebben een duidelijke interpretatie als afstandsmaat. Na een grondige data-analyse, worden representaties voorgesteld die goed aansluiten bij een Gaussianse kernel en specifiek zijn voor het verbruik van elektriciteit. Een multicomponenten-structuur van kernel-machines wordt geïntroduceerd om de orthogonaliteit van de invoervariabelen te vergroten. Het model voor korte-termijn windenergie-voorspellingen gebruikt enkel numerieke windsnelheden en de windrichting als invoer. De te grove metingen van windsnelheden in meters per seconde worden verfijnd door het combineren van weerdata van enkele weerstations. Hiernaast wordt de windrichting geprojecteerd op een cirkel waardoor deze beter aansluit bij een Gaussiaanse kernel.

In hoofdstuk 5 wordt een overzicht gegeven van de Kernel-Machine Library, een C++ bibliotheek die geschreven is voor het faciliteren en ontwikkelen van kernel-machines, zowel voor academisch gebruik als voor de ontwikkeling van applicaties die in de praktijk gebruikt kunnen worden. Na toegelicht te hebben welke doelgroep deze bibiliotheek wil bereiken in vergelijking met gelijksoortige software, wordt een aantal eisen gesteld dat tegen het licht gehouden is tijdens het ontwerp van de bibliotheek. De implementatie maakt gebruik van eigenschappen van modern C++, zoals template-metaprogrammeren, om goede prestaties te kunnen behalen en tegelijkertijd een comfortabele interface te kunnen bieden. Het maakt de keuze tussen gespecialiseerde algoritmen tijdens het compileren van een programma, wat het erg efficient maakt. Het compileren op verscheidene platformen en het testen op juistheid is ook uitgevoerd.

In hoofdstuk 6 wordt de praktijk van het modelleren van elektriciteitsbelasting toegelicht. In geval van het model voor het verbruik van elektriciteit wordt de componentenstructuur zoals geïntroduceerd in hoofdstuk 4 geëxpandeerd op een experimentele wijze. De voorgestelde representaties voor dagtypen en schemering worden getest op hun toepasselijkheid. De structuur van componenten wordt opgebouwd met een kalender-, trend-, temperatuur-, straling-, en windcomponent. Deze componenten maken het mogelijk de elektriciteitsverbruiken te ontrafelen. Verscheidene nieuwe feiten worden ontdekt met behulp van dit systeem. Het gecreëerde systeem produceert nauwkeurige en gedetailleerde predicties van het verbruik van elektriciteit. Voor het korte-termijn voorspellingssysteem van windenergie-productie wordt incrementeel een aantal weerstations toegevoegd om een zo optimaal mogelijke selectie van de samenstelling van de invoerruimte te kunnen maken.

In hoofdstuk 7 wordt geconcludeerd over de geïntroduceerde methoden, modellen en gereedschappen, en hun experimentele resultaten in het gebruik bij het voorspellen van elektriciteitsbelasting.

Curriculum Vitae

Personal details

Name	ter Borg
First names	Rutger Willem
Birth	Delfzijl, the Netherlands, November 19, 1974.
E-mail	rutger@terborg.net

Education

2001–present	Ph.D. (doctor) at Delft University of Technology, thesis *Electricity Load Modelling using Computational Intelligence*.
1994–2000	M.Sc. (ingenieur) in Technical Informatics at Delft University of Technology, dissertation *Extracting Fuzzy Rules using Genetic Programming*.
1996–1998	Foundation course (propedeuse) at Rotterdam School of Management at Erasmus University Rotterdam.
1993–1994	Mechanical Engineering at Delft University of Technology.
1987–1993	Pre-university education (vwo) at Willem de Zwijger College, Bussum. Subjects Dutch, English, Biology, Economics, Mathematics B, Physics, Chemistry. Additional course on Mathematics A.

Work experience

2005	Developed and implemented a gas consumption model at Nuon NV, Energy Sourcing, Quantitative Analysis Group.
2005	Developed and implemented a short-term wind power production forecasting system at Nuon NV, Energy Sourcing, Quantitative Analysis Group.
2001–2004	Researched, developed and implemented a predictive electricity demand model at Nuon NV, Energy Sourcing, Applied Research & Technology Group.
2000–2001	Conducted research to genetic algorithms. Set up a web hosting server of about 15 Internet domains.
1997–1999	Developed IT-related products at Kojac CV, Delft.
1996–1998	Coached extra lessons in exact sciences to students of pre-university level.

1997	Designed and implemented an automated invoicing system for meeting room rentals.

Other activities

2005	Participated the International Conference on Artificial Neural Networks (ICANN), Warshaw, Poland.
2004	Participated McKinsey & Company's Ph.D. Masterclass, Amsterdam.
2004	Participated the Eunite conference on invitation, Oulu, Finland.
2003	Member of an M.Sc. examination committee at Delft University of Technology.
1997	Prize winner *Best Business Plan 1997*, Erasmus University Rotterdam.
1994–1995	Building commission at the Delftsch Studenten Corps.

Languages

Nederlands	Moedertaal
English	Fluent
Française	Modéré
Deutsch	Mäßig
Español	Medio

Other interests

Sailing (on yachts), open water diving, skiing, playing a game of golf, playing squash, historic car rallies (participated the *Tulpenrallye* several times), (trading at) financial markets, scientific developments, current affairs.

The Hague, 7th November 2005