UPPSALA
UNIVERSITET

F12015
Examensarbete 30 hp
Maj 2012

Predicting demand in district heating systems

A neural network approach

Niclas Eriksson

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Predicting demand in district heating systems - A neural network approach

Niclas Eriksson

To run a district heating system as efficiently as possible correct unit-commitment decisions has to be made and in order to make those decisions a good forecast of heat demand for the coming planning period is necessary. With a high quality forecast the need for backup power and the risk for a too high production are lowered. This thesis takes a neural network approach to load forecasting and aims to provide a simple, yet powerful, tool that can provide accurate load forecasts from existing production data without the need for extensive model building.

The developed software is tested using real life data from two co-generation plants and the conclusion is that when the quality of the raw data is good, the software can produce very good forecasting results.

Handledare: Håkan Fjäder
Ämnesgranskare: Per Lötstedt
Examinator: Tomas Nyberg
ISSN: 1401-5757, UPTEC F12015
Sponsor: EnviLoop AB

Contents

1 Introduction

District heating is a system for distribution of heat produced in a central plant to customers in a limited geographic area. In a cold climate such as the one in Sweden estate heating is an important energy consumer and even when heating needs are low in summertime there is a demand for hot tap-water from both consumer and industries.

During 2010 the total power supplied to consumers from district heating systems in Sweden exceeded 55 TWh. With an average price of 0.77 Swedish crowns per KWh this corresponds to an annually delivered value in excess of 42 billion Swedish crowns [24]. As for almost any industry the operators of district heating systems are subject to competition. One of the biggest factors a supplier of energy can compete with is the cost per delivered KWh and this is true for suppliers of heating as well as for suppliers of electricity. To keep a low cost efficient use of resources is essential and the main resource usage that can be affected by daily operations is the cost for fuel. The efficient use of resources is important not only from a pricing or profitability perspective but also for society as a whole. Almost all energy production on a large scale is coupled to some sort of less desirable emissions and in order to keep those emissions as low as possible it is important to keep the efficiency as high as possible.

To be able to keep the price per produced KWh as low as possible, minimize fuel usage and avoid wasteful use of available resources short-term production planning is an essential tool. A good plan allows the production to meet demand without either over-production or shortages which can both be very expensive. To lay the foundation for good short-term production planning it is essential to have good short-term production forecasts that can accurately predict the demand for the coming planning period.

Despite the importance of good production forecasts and the large values being produced many smaller production facilities lack proper tools to forecast demand. This implies that there is a possibility to improve the day-to-day operation and the profitability for these production facilities.

2 Background

2.1 The district heating system

District heating is a system for distribution of heat generated in one, or more, centralized production facilities to customers within a limited geographic area. A schematic district heating system could be said to consist of three parts, production facilities, P, distribution network and customer stations C_i [18] as illustrated in Figure 2.1.

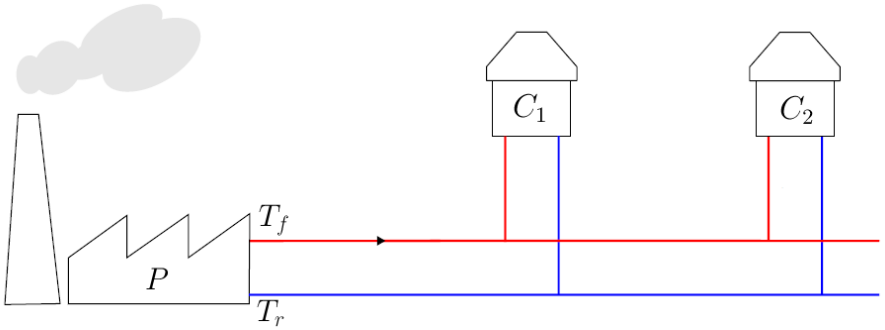


Figure 2.1: Schematic drawing of a simple district heating system [10]

The heat generated in the production facility is used to heat water, which is distributed through a network of pipes, referred to as the primary network, that connects the customer stations to the production facility. The customer stations consist of heat exchangers that transfer heat from the primary network to a secondary network that distributes the heat in the customer's property. The production facility in a district heating system can either be a cogeneration plant or a heat-only boiler station. The difference between the two kinds of plants is that the cogeneration plant can be used to generate heat and electrical power simultaneously while the heat-only boiler station only generates heat. The plants usually generate heat through burning of fuels such as oil, gas, biofuel or waste but sometimes heat generated as a residue from other industrial processes is used. Most production facilities have more than one production unit and usually the different units use different fuels to be able to respond to varying conditions. Since the different boiler units use different fuels and often have varying production capability the cost for starting and stopping the units vary and the different boilers also have varying power to cost ratios for their energy production.

The power that is distributed to the net from the production facility depends on the flow and on the difference between supply temperature T_f and return temperature T_r. A typical values are $T_f \approx 80°C \rightarrow 90°C$ and $T_r \approx 40°C \rightarrow 50°C$

2.2 Production planning

A prerequisite for good profitability, no matter the business, is effective use of the available resources. For a power producer one of the largest costs that can be influenced by the daily operations is the cost for fuel to the boilers. The amount of fuel that needs to be fed to the boilers is determined by what we will refer to as the *heat demand* [10] which can be formulated as

$$Q = Q_{loss} + \sum_{c=1}^{C} Q_c \tag{2.1}$$

where Q is the total heat volume to be distributed to the district heating network, Q_{loss} is the heat loss in the network and Q_c is the heat consumption for an individual consumer c. As mentioned earlier different boilers in a production

facility often have different starting, stopping and running cost that in combination with the boilers' minimum and maximum power levels determine under what conditions it will be necessary and profitable to run the boiler. To obtain good profitability it is essential to choose the right boiler, or mix of boilers, to cover the heat demand for any given situation without producing excess heat. Since most production facilities aim to keep T_f and T_r within a certain, small, range Q_{loss} can be considered a constant and for most cases the heat load Q can be treated as a function of only Q_c.

The consumer part of the heat load Q_c can be divided into two parts where one part depends on ambient factors such as outside temperature, wind speed and direction, humidity and solar radiation and the other part depends on social factors such as working hours, weekends, holidays and people's daily routines. We can therefore express Q_c as

$$Q_c = Q_{ambient} + Q_{social} + Q_{rnd} \tag{2.2}$$

where Q_{rnd} corresponds to random components that can not be adequately modelled as either social or ambient factors.

Production planning is typically divided into *short-term*, *mid-term* and *long-term* planning according to which time horizons are considered. Table 2.1 shows typical time horizon and usage for the three planning horizons. In this thesis we will focus on short term planning and specifically on the production of short-term forecasts to facilitate short-term production planning.

Table 2.1: Planning horizons

	Short-term	Mid-term	Long-term
Horizon	24 hours - 1 week	3 week - 12 months	2 years –>
Usage	Unit commitment	Fuel purchase	Investment planning

Today the production planning for many domestic heating systems is performed in a manual ad-hoc manner using historical averages and weather forecasts in an attempt to predict future heat demand. A tool that is used for short term planning is two dimensional contour plots showing historical power output as a function of time of day and outside temperature (Figure 2.2). The use of contour plots gives a good approximation of what the historical mean production has been during similar conditions but depends on the presence of an experienced operator to correct for effects from weekends, holidays, quick variations in ambient conditions etcetera. The dependence on experienced operators is a problem since not all operators have the necessary experience, and even if they have, the predictions could be much improved and simplified by using an automated system.

The issue with the available software is that they are either generic software for statistical analysis that demands expert knowledge to use or complex

highly tailored software suites for individual, large, cogeneration plants. Therefore there is a need to develop a simple, user friendly, interface for production planning using methods that can be applied to a wide range of district heating systems with little or no modification.

2.3 Time series

The main subject for this thesis could be said to be the treatment of *time series*.

A time series is "a sequence of observations taken sequentially in time" [4]. Time series are extremely common and many types of data are represented as time series. Examples include closing prices at stock markets, yearly population surveys, hourly sales in a super market and the result from hourly measurements of industrial processes. One of the special features of time series is that we expect successive observations to be dependent. This dependency is very important since it enables us to draw conclusions about coming values of the time series.

2.3.1 Notation

To avoid confusion we will adopt a notation that is largely based on the notation used in reference [19]. We will denote a single observed value in a time series as Y_t where the subscript t indicates at which time the observation was made. We presuppose that all observations Y_t has been made at discrete, equidistant points $t \in \mathbb{Z}$. We will denote the present time period (the latest available observation) by n and previous time periods as $n-1, n-2, ..., 1$.

Since we will be concerned with the prediction of future values of Y using past observations we denote an individual forecast as $\hat{Y}_t$ and the error for that forecast as $e_t = Y_t - \hat{Y}_t$. The forecasts will usually be made for a future time period $n+p$ and in that case the error e_{n+p} is unknown.

Table 2.2: Time series notation used, adopted from Makridakis et al. [19].

	Observed values					Forecasted values		
	Y_1	Y_2	...	Y_{n-1}	Y_n			
Period t	1	2	...	$n-1$	n	$n+1$	...	$n+m$
Estimated values	$\hat{Y}_1$	$\hat{Y}_2$	...	$\hat{Y}_{n-1}$	$\hat{Y}_n$	$\hat{Y}_{n+1}$	...	$\hat{Y}_{n+m}$
Error	e_1	e_2	...	e_{n-1}	e_n			

2.3.2 Stationarity

A time series may be either stationary or non-stationary. An stationary time series is a time series where the observations are drawn from an underlying *stationary process* [25]. A stochastic process $\{Z_t\}$ is said to be *strictly stationary* if the *cumulative distribution function*, F_Z is independent of t. That is, a stochastic process, where $F_Z(z_{t_{1+k}}, z_{t_{2+k}}, ..., z_{t_{n+k}})$ is the cumulative distribution function of the joint distribution of any set of n consecutive observations, is stationary if:

$$F_Z(z_{t_1}, z_{t_2}, ..., z_{t_n}) = F_Z(z_{t_{1+k}}, z_{t_{2+k}}, ..., z_{t_{n+k}}) \forall k \in \mathbb{Z} \tag{2.3}$$

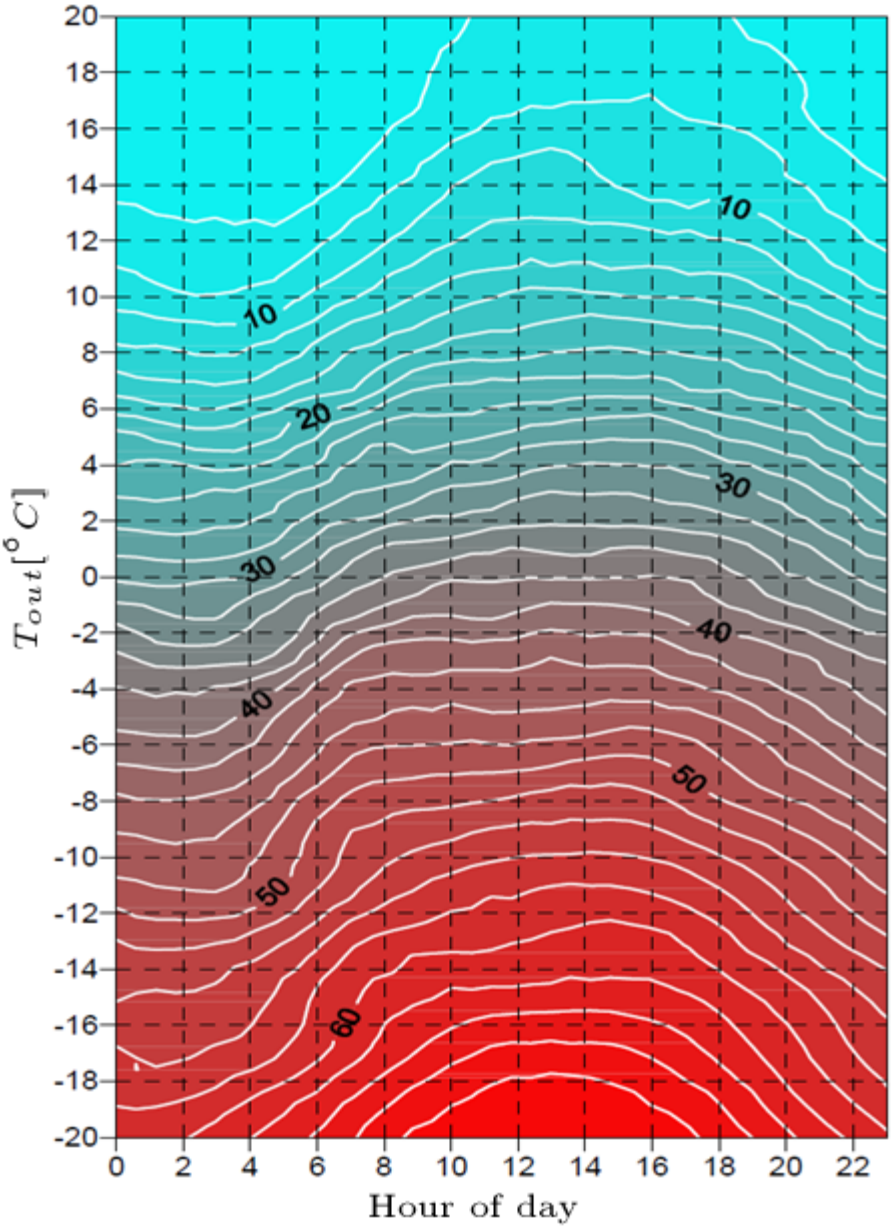


Figure 2.2: A simple aid for production planning. The isolines correspond to the average power output for a given hour and outside temperature.

We shall also define *weak stationarity*. A stochastic process is stationary in a weak sense if it has a time independent mean,

$$\mathbb{E}[z_t] = \mu_{z_t} = \mu_{z_{t+k}} \forall k \in \mathbb{Z} \tag{2.4}$$

and time independent autocorrelation

$$R_z(t_1, t_2) = R_z(t_1 + k, t_2 + k) = R_z(t_1 - t_2, 0) \forall k \in \mathbb{Z} \tag{2.5}$$

2.3.3 Trend

A *trend* is a very common feature of time series. A trend is some change in the properties of a time series that take place slowly over the whole span of the series being investigated. Judging if a long series of past data contains a trend can be done in several ways, using both mathematical and visual tools. Although its intuitively clear what a trend is it is hard to define what a trend is mathematically since there is no way of knowing if the time series is diverging from the previous average level because of short term oscillations , as a part of a cyclical movement or as a part of a long term trend. A trend is sometimes said to be a gradual change in the mean of the observations, but the trend could equally well be a change in some other statistical property, e.g. variance. Therefore we shall define a trend as *any systematic change that affects the level of a time series* [25].

It is very common in time series analysis to *de-trend* the data in a pre-processing stage. A trend is often removed when it is thought to obscure the relationships that are studied. If we are convinced that the trend is due to some deterministic property of the time series, one common way used to de-trend the series is to apply a *regression model* [25]. Common choices include linear, logarithmic and exponential regression.

If the trend is in fact stochastic and therefore can not be described using a deterministic model the regression method will fail to accurately de-trend unseen data. Therefore another commonly used method is *differencing* [4]. Differencing is a method where the values of successive observations are subtracted from each other in some well-defined, time dependent manner. We define the difference operator ∇, operating at z_t, as:

$$\nabla z_t = z_t - z_{t-1} \tag{2.6}$$

A convenient notation to introduce along with differencing is the *backward shift operator* B:

$$B^k z_t = z_{t-k} \tag{2.7}$$

Seasonality

Seasonality or periodic variation describes a cyclic, repetitive and predictable change in average level based on season. The periods of the variation cycle can differ from series to series, but common periods are yearly, monthly and daily.

3 Methods

The ability to predict future events is often a major competitive edge in many situations and therefore it is not surprising that attempts at analysing and

predicting systems of varying complexity is a very old science. This historical interest for prediction and forecasting techniques implies that a lot of effort has gone into developing increasingly complex prediction models and since a good forecast can be the only difference between success and failure for many companies the wide range of available models and techniques to choose from is not surprising.

According to [7] the four major techniques used to tackle the problems of forecasting are:

- Deterministic modeling
- Conceptual modeling
- Expert systems
- Statistical modeling

Depending on available information about the system, the complexity of the system and the purpose of the forecasts, any technique, or combination of techniques, could be employed.

3.1 Deterministic modeling

A deterministic model is an attempt to mathematically describe the physical laws that govern the behaviour of a system. An attempt to describe an entire central heating system, including plants, network and customers using a deterministic model is done by Arvatson [1]. Here the deterministic part of the load is modelled using physical relations. Arvatson uses several physical relations such as energy balances and continuity equations to describe how the district heating system will respond to various conditions. The complexity of the district heating system that Arvatson models with thousands of customers and several plants producing not only hot water, but also electricity leads to a very complex model which is tailored to a specific context. Even though Arvatson's model arrives at very good forecasting results, the complexity and limited possibilities of the model to adapt to a new setting and the fact that we are uninterested in a physical understanding of the system makes it unsuitable for our purposes.

3.2 Classical time series methods

There exist a wide range of methods that can be used to more or less accurately predict future heat demand. The classical methods depend on the assumption of linearity and stationarity, either in the raw data or as a result of some well-defined and invertible transformation of the raw data. While those assumptions simplify modelling and enable a coherent use of statistical measures it also limits the cases when the methods can be used and places demands on proper pre-processing of data. [6]

Examples of classical time series methods are ARIMA modelling, periodic AR modelling, Holt-Winters exponential smoothing and principal component analysis.

One of the most common classical time series methods is the ARIMA model, as popularized by Box and Jenkins in their book *Time Series Analysis: Forecasting and Control* [3] published in 1970. An ARIMA model consists of three parts;

autoregressive (AR), integrating (I) and moving average (MA). An ARIMA-model is often denoted as an ARIMA(p, d, q)-model where p is the order of the autoregressive filter, d the order of integration and q the order of the moving average filter. The general ARIMA process can be expressed as:

$$\left(1 - \sum_{i=1}^{p} \phi_i B^i\right) (1 - B)^d X_t = \left(1 + \sum_{i=1}^{q} \theta_i B^i\right) \varepsilon_t \tag{3.1}$$

where B is the back-shift operator, ε_t are error terms, X_t are the time series terms, ϕ_i are parameters for the autoregressive model and θ_i are parameters of the moving average model. [3]. ARIMA-models have been extensively used to forecast heat-demand and they often yield good results (see e.g. [10, 14] for examples).

3.3 Modelling with artificial neural networks

An artificial neural network (ANN) is an information processing tool capable of modelling complex data relationships [21, 16]. Research into ANNs started with biological neural networks as inspiration and was often an attempt to better understand the biological counterpart but today the research field has shifted to be more concerned with artificial implementations and their performance. An artificial neural network is characterised by its ability to learn, parallelism and robustness to errors and noise in both input data and in the network itself. Research into ANNs had a breakthrough in the 80s and since then ANNs have been used in a number of different applications ranging from pattern recognition to system control. [17]

ANNs has been applied to a wide range of forecasting problems with varying success but several authors report good results when applying ANNs to load forecasting (see e.g. [23, 22, 13, 6]. One of the main advantages of ANNs over other models is their inherent non-linearity that enables them to capture non-linear data behaviour. There is some evidence that the load series might primarily be linear [6][1] but there is no evidence whether this is the general case or if it is just a valid conclusion for a particular setting. Therefore the use of an ANN for prediction enables us to use many different inputs without considering their relationship to the dependent variable. The use of ANNs over classical time series methods also enables us to use a data-driven approach that allows a more flexible implementation that can be re-used in many different contexts.

Since we will use a neural network approach to load forecasting a summary of the relevant ANN theory follows below.

R. Rojas [21] lists three main elements that govern the behaviour of the artificial neural network:

- The structure of the nodes
- The topology of the network
- The learning algorithm used to find the weights of the network.

[1] Reference [6] is concerned with predicting electric load in the Czech Republic.

3.3.1 The neuron (node structure)

The basic building block in an ANN is the *neuron*, depicted in Figure 3.1 with n inputs. Each neuron is capable of receiving input over an incoming edge, either from another neuron or from an external source. Each input channel i can transfer a value $x_i \in \mathbb{R}$ and each channel is associated with a weight w_i. While unweighted networks exist, we shall only consider weighted networks. In a weighted network the weights are the most important parameter since adjustment of the weights is what allows the network to learn. The input arrives

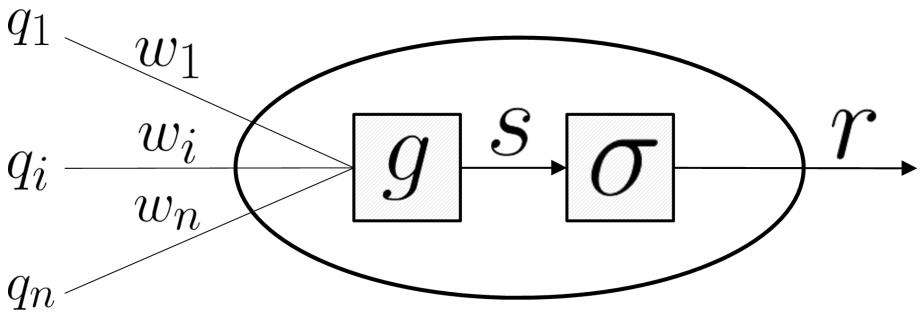


Figure 3.1: Schematic neuron with n inputs, transfer function g and activation function σ.

over the input channels to the integrating stage of the neuron where the multiple inputs are reduced to a single value which can be fed to the activation function. While any function $g : \mathbb{R}^n \to \mathbb{R}$ could be used the most common approach is to use a weighted sum where each input x_i is multiplied with the corresponding weight w_i (see equation 3.2).

$$s = \sum_{i}^{n} w_i q_i \tag{3.2}$$

After the integrating stage, when the inputs have been reduced to a single value, the value is fed to the activation function (see Equation 3.3 or 3.4). The activation function could theoretically be any function $\sigma : \mathbb{R} \to \mathbb{R}$, but in practice a few functions with desirable properties are used (see e.g. [8] for a comprehensive review of neural network activation functions). Some of the most common functions include the logistic function (3.3), commonly referred to as the *sigmoid function*

$$\text{sigmoid}(s) = \frac{1}{1 + e^{-cs}}, \tag{3.3}$$

and the *hyperbolic tangent function* (3.4)

$$\tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}. \tag{3.4}$$

Sometimes the literature refers to the activation function as a *squashing* function since it also serves as a mapping from an infinite range to a finite. In the case of the sigmoid function the mapping is $s : \mathbb{R} \to (0, 1)$ and in the case of the hyperbolic tangent the mapping is $\tanh : \mathbb{R} \to (-1, 1)$. To be considered as a suitable activation function we shall also require a function that has a continuous derivative since this is a property required by some network learning schemes.

Figure 3.2 shows the shape of the *tanh* and the sigmoid activation functions. As can be clearly seen both functions have some similarities with a step function and as we increase c for the sigmoid function we can see that the steepness of the curve increases and as c approaches ∞ the sigmoid function converges towards a step function.

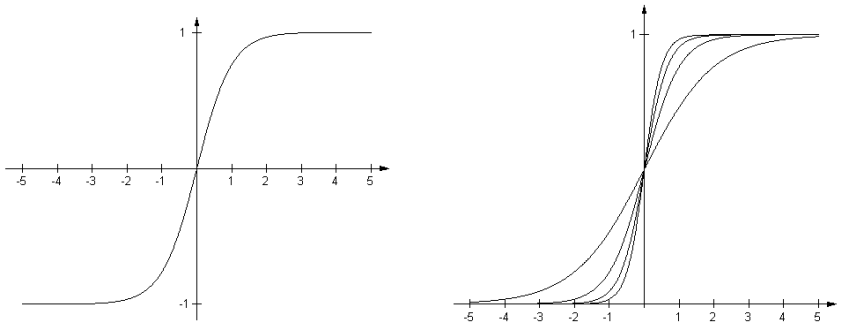


Figure 3.2: Tanh and Sigmoid ($c = 1$, $c = 2$, $c = 3$, $c = 4$) activation functions

It is common to introduce a bias to the neuron. Although Figure 3.1 shows no explicit bias term the bias could easily be implemented by adding an extra input with a fixed input value $x_b = -1$. This gives a neuron that can be considered to have $n + 1$ inputs. The bias gives us the ability to offset the output from the neuron by adjusting the weight w_b associated with the bias. Figure 3.3 shows the effect on the neurons output y from changing the bias weight for a simple neuron with only one input x_1.

Using the notation from Figure 3.1 we can write the entire output y from a neuron, with bias, as a function of the neurons weights and inputs. This expression is given in equation 3.5

$$r(q, w) = \sigma\left(\sum_{i}^{n+1} w_i q_i\right) \tag{3.5}$$

where σ corresponds to the activation function (e.g sigmoid or tanh), $w_{n+1} = w_b$ and $x_{n+1} = x_b = -1$.

3.3.2 Neural network topology

The *topology* of a neural network specifies how the neurons are laid out in the neural network. While the neurons are responsible for the basic properties of the ANN the high level behaviour of the network is largely determined by the topology. The neurons are commonly grouped into layers. A layer can consist of one or more neurons and the connections between the neurons are then defined in terms of connections between the layers. One of the most common type

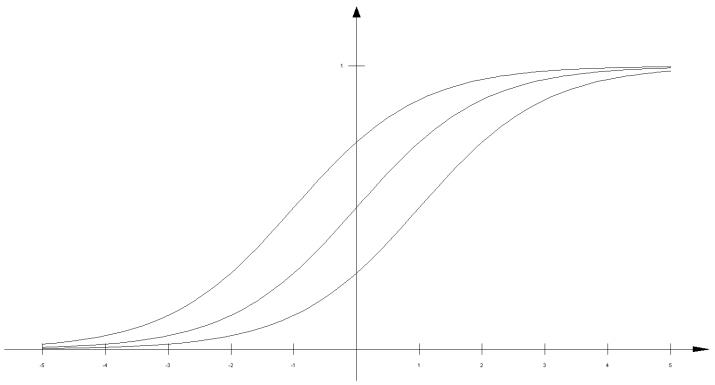


Figure 3.3: Effects of adjusting the bias weight for a simple neuron with one input, example using sigmoid activation (3.3) and $w_b = -1, 0, 1$.

of ANNs is the *fully connected feed-forward* network. A network is said to be fully connected if all neurons in one layer are connected to all neurons in the next layer. A feed forward network only passes input forward through the layers; there are no loops in the network that allows data to be passed backwards through the layers. *Recurrent networks* are different from feed forward networks in that they allow data to be passed back to earlier layers.

The most common networks consist of three kind of layers who are referred to as *input layer*, *hidden layer* and *output layer*. The input layer is often not regarded as a real layer since it usually does not transform the data in any way but simply serves as a mean to present data to the hidden layer. The purpose of the neurons in the hidden layer is to transform the input into an intermediary representation that can be presented to the neurons in the output layer, where the final network output is generated from the intermediate representation. [12]

It can be proven that a fully connected feed forward network with one or more hidden layer is a *universal function approximator* [21]. This means that a neural network with a *finite*, but *sufficient*, number of neurons can approximate any continuous function to an arbitrary precision. Thereby we know that failure of a neural network application to learn a pattern depends on inferior learning algorithms, an inadequate number of hidden units or the fact that there is a non-deterministic pattern in the data that cannot be learnt [15].

Figure 3.4 shows a schematic drawing of a feed forward network with p inputs, L hidden units and m output units. Given a feed-forward network with p inputs, m outputs and a fully connected structure with notation as shown in Figure 3.4, using Equation 3.5 we can write y_k as:

$$y_k = \sigma^y \left(\sum_{j=1}^{m+1} w_{kj}^y \sigma^h \left(\sum_{i=1}^{p+1} w_{ji}^h x_i \right) \right) \tag{3.6}$$

where σ^h and σ^y is the activation function used in the hidden- and output layer respectively.

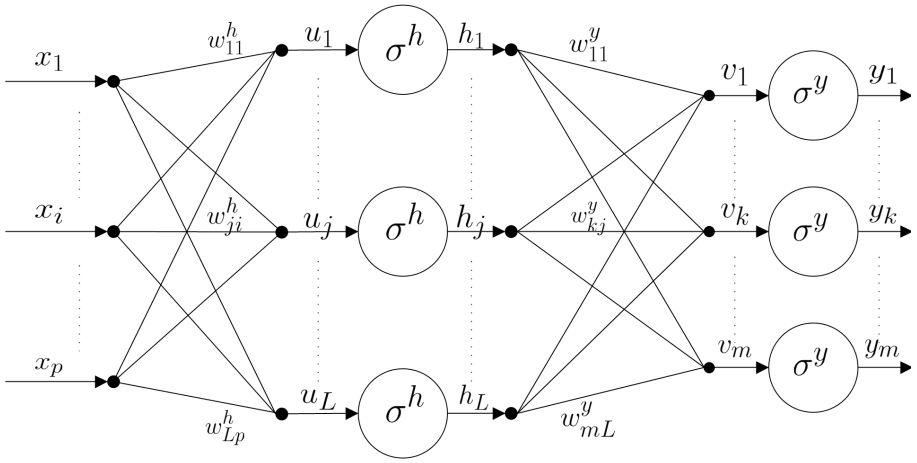


Figure 3.4: Schematic fully connected feed-forward network with p inputs, one hidden layer with L neurons and m output neurons. σ denotes the activation function.

3.3.3 Learning algorithm

A neural network is taught by iteratively adjusting the weights until the correct result is produced. The methods for adjusting the weights can be divided in *supervised*, *reinforcement* and *unsupervised* training methods [9]. An unsupervised training method presents the network with input data but provides no target outputs or feedback on the output. Unsupervised learning is mainly concerned with finding patterns in data. In reinforcement learning the outputs produced by the network is translated into a scalar reward (or sometimes punishment) given as feedback on the output and the network is allowed to adjust its output to maximize the reward (or minimize the punishment). In supervised learning the network is presented with an input and then the networks output is compared to some desired target output for the given input and the network is adjusted accordingly. In this thesis we will only use supervised learning schemes since these are best suited for time series regression.

Parameter (weight) estimation in a neural network using minimization of an error function becomes a nonlinear optimization problem. There exists a very wide range of optimization methods that has been applied to neural networks (see e.g. [5, 20] for a review of nonlinear optimization strategies) and an attempt to compare and review them would be well out of the scope for this text. Since there is no theoretical or empirical evidence for the existence of a single best method for any given problem [5] we will focus on methods that are simple and stable. For supervised learning we will have an external teacher that presents a set of inputs, or *patterns*, that are feed through the network and then compared to a desired output by the teacher.

Backpropagation with gradient descent

Gradient descent is a general first-order method for optimization that aims to minimize a given function $f(\mathbf{x})$. Minimization is achieved by recognizing that if $f(\mathbf{x})$ is differentiable in the neighbourhood of a point $\mathbf{a}$, $f(\mathbf{x})$ decreases fastest if we move from $\mathbf{a}$ in the negative direction of the gradient of $\mathbf{a}$, that is: $-\nabla\mathbf{a}$.

When applied in an artificial neural network context gradient descent is used to minimize the value of a predefined *cost-function*, sometimes referred to as an error-function. In theory any function f can be used as a cost function, but in practice a few functions are widely used. One of the most commonly used function is the *mean-squared error* function [12] as defined in equation 3.7.

$$E = \frac{1}{2}(d - y))^2 \tag{3.7}$$

utilizing equation 3.6 and substituting into 3.7 we can express the error at output neuron k for pattern presentation n as a function of the weights w in the network:

$$E_k^{(n)}(w) = \frac{1}{2}\left[d_k^{(n)} - \sigma^y\left(\sum_{j=1}^{m+1} w_{kj}^y \sigma^h\left(\sum_{i=1}^{p+1} w_{ji}^h x_i^{(n)}\right)\right)\right]^2 \tag{3.8}$$

By using the cost function which defines optimal weight values in the sense that they minimize the networks deviation from the desired values we can improve our network by changing the values of w in such a fashion that the value of the cost function decreases. One way of accomplishing this is to define a weight update rule based on gradient descent that changes the weights by a factor Δw_{ik} proportional to the gradient of E with respect to the weights w at the current location.[12]

$$\Delta w_{ij}^{(n)} = -\eta \frac{\partial E^{(n)}}{\partial w_{ij}} \tag{3.9}$$

To use the update rule defined in 3.9, also known as the *delta rule* [20] we need a way of computing $\partial E/\partial w_{ij}$, by applying the chain rule we get the *delta rule*

$$\Delta w_{ij}^{(n)} = -\eta \delta_i^{(n)} h_j^{(n)} \tag{3.10}$$

where h_j is the output from unit j to unit i and δ_k is the *local gradient* for unit k. For an neuron in the output layer with a linear activation function we have that:

$$\delta_k = d_k - y_k \tag{3.11}$$

where d_k is the desired response for output neuron k and y_k is the actual output from neuron k [11].

If we instead would consider a hidden neuron we encounter a problem since there is no desired response d_k for the neurons in the hidden layer and therefore we can not determine E with only local information. The solution to this problem is to *back-propagate* the error signals through the network [11]. Figure 3.5 shows the basic idea behind back propagation where the function signals (the actual values that are being passed through the network to eventually create the output) are shown with solid lines and the back-propagation of the error

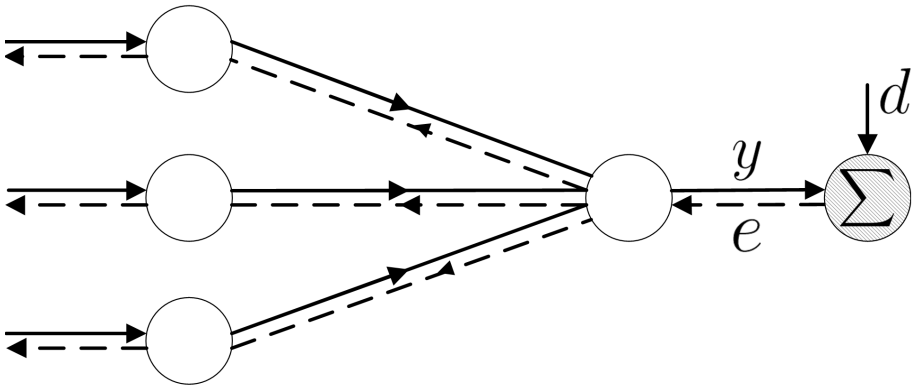


Figure 3.5: Illustration of the basic signals in a neural network, solid lines illustrate forward propagation of the function signal and dashed lines illustrate back propagation of the error signal (adapted from [11]).

signals (derived as a function of the produced output) are shown with dashed lines.

When the error signal is backpropagated to the neurons in the hidden layer we get the local gradient $\delta_j^{(n)}$ for a hidden neuron j

$$\delta_j^{(n)} = -\frac{\partial \tilde{E}^{(n)}}{\partial h_j^{(n)}} \frac{\partial h_j^{(n)}}{\partial u_j^{(n)}} = -\frac{\partial \tilde{E}^{(n)}}{\partial h_j^{(n)}} \sigma^{h\prime}\left(h_j^{(n)}\right) \tag{3.12}$$

where u is the net input to the neuron, h is the output, σ^h is the activation function used for the hidden layer (see Figure 3.4) and

$$\tilde{E}^{(n)} = \sum_{k=0}^{m} E_k^{(n)} \tag{3.13}$$

We need to calculate the derivative $\partial E_k^{(n)} / \partial h_j^{(n)}$ and using the chain rule we can write the derivative as

$$\frac{\partial E_k^{(n)}}{\partial h_j^{(n)}} = \frac{\partial E^{(n)}}{\partial v_k^{(n)}} \frac{\partial v_k^{(n)}}{\partial h_j^{(n)}} \tag{3.14}$$

Using (3.7) and considering Figure 3.4 we can write E as a function of v

$$E_k^{(n)}(v) = \frac{1}{2}\left(d_k^{(n)} - y_k^{(n)}(v)\right)^2 = \frac{1}{2}\left(d_k^{(n)} - \sigma^y\left(v_k^{(n)}\right)\right)^2 \tag{3.15}$$

which gives us that

$$\frac{\partial E_k^{(n)}}{\partial v_k^{(n)}} = -\sigma^{y\prime}\left(v_k^{(n)}\right) \tag{3.16}$$

We also know by considering Figure 3.4 that

$$v_k^{(n)} = \sum_{j=0}^{L} w_{kj}^{(n)} h_j^{(n)} \tag{3.17}$$

which directly gives us that

$$\frac{\partial v_k^{(n)}}{\partial h_j^{(n)}} = w_{kj}^{(n)} \tag{3.18}$$

By substituting (3.16) and (3.18) into (3.14) and (3.14) into (3.12) we finally arrive at

$$\delta_j^{(n)} = \sigma^{y\prime}\left(v_j^{(n)}\right) \sum_{k=0}^{m} \delta_k^{(n)} w_{kj}^{(n)} \tag{3.19}$$

and thereby we have a method to update the weights not only in the output layer, but also in the hidden layer and the results can easily be generalized to use an arbitrary number of hidden layers.

Momentum

Momentum is a heuristic technique that aim to improve the convergence rate for gradient descent backpropagation. In normal gradient descent backpropagation the rate of learning is determined by the learning rate, η, and by adjusting η we can adjust how much the neural network adapts its weights in each iteration. The problem is that for small values of η the network will learn very slowly and might get stuck in a local minima on the error surface and for large values of η the network may fail to converge into a minima and start to oscillate. Momentum attempts to increase the rate of learning without the need to increase η by redefining the weight update rule as

$$\Delta w_{z\tilde{z}}^{(n)} = \alpha \Delta w_{z\tilde{z}}^{(n-1)} + \eta \delta_z^{(n)} r_{\tilde{z}} \tag{3.20}$$

where $w_{z\tilde{z}}$ designates a weight on the connection from neuron $\tilde{z}$ to the neuron z, $r_{\tilde{z}}$ is the output from neuron $\tilde{z}$ (see Figure 3.1) and α is the *momentum constant* ($\alpha > 0$). Equation 3.20 is known as the *generalized delta rule* and if we let $\alpha = 0$ we get the normal delta rule. The inclusion of a momentum term can also reduce the risk of the network to get stuck in a local minimum on the weight surface by providing the extra momentum necessary to force the network out of shallow local minimums. [11]

Stochastic and batch training

Gradient descent back propagation can be performed either in *batch mode* or in *stochastic mode*. In stochastic mode an input pattern is presented to the network and the corresponding output is computed, the error is backpropagated and the weights are immediately updated according to the weight update rule. Batch mode works just like stochastic mode, except that the weights are not updated before all patterns in an epoch has been presented to the network. It is only in batch mode that the method can be said to perform true gradient descent and stochastic mode is named stochastic since it performs a *stochastic approximation* to gradient descent which makes it less likely to get stuck in a local minima [11].

All theory in this chapter has been derived for stochastic training, but we can easily adopt it for batch training by redefining the cost function as

$$E_k = \frac{1}{2N} \sum_{n=1}^{N} \sum_{k=1}^{m} \left(d_k^{(n)} - y_k^{(n)}\right)^2 \tag{3.21}$$

and wait until the entire training set has been presented before updating the weights.

When to stop training

Terminating the training of a neural network can be achieved in a wide range of ways, from simply training for a specified number of epochs to more advanced techniques involving several relevant measures of the neural networks performance. Since there is no general proof for convergence of the back propagation algorithm there is no universal stopping criterion that can be used to determine convergence. We can however use a mathematic foundation to establish useful heuristics for when to stop the training. If we let w denote a weight vector and w^* denote a weight vector corresponding to a minimum on the error surface we know that $\nabla w = \vec{0}$ for $w = w^*$. This implies that we could define the stopping criterion as $\nabla w < \epsilon$ where ϵ is some sufficiently small threshold. While $\nabla w < \epsilon$ is a good theoretical stopping criterion it is not very useful in practice since the network may require long training times to achieve a sufficiently small value for ∇w and to make matters worse it is expensive to compute ∇w [11].

More useful stopping criteria would not be dependent upon calculating additional measures besides those that are already being used in the neural network training. We therefore define a stopping rule based on the cost function $E(w)$. We could simply stop training when $E(w)$ reaches a sufficiently small value but it is also straightforward to see that if we have a set of weights w that are close to a minimum w^* the cost function $E(w)$ will be almost stationary. We will therefore use a stopping criterion based on the per cent per epoch change in the cost function together with the absolute value of the error function to stop training when it has been successful. To be able to stop training even if convergence is not achieved we can limit the maximum number of epochs allowed before training is considered to have failed.

3.4 Preprocessing of data

When there exist observations of an explanatory variable we can choose to either include that series of explanatory variables as an input to our neural network model, or we can choose to preprocess the dependent time series using a regression technique. This process is often refered to as deflating

3.4.1 Error treatment

Since the data is measured from an industrial process any problem with this process will be reflected in the data. The district heating network is often large and can be affected by many more or less unpredictable events such as broken pipes, major rebuilds, service work etc. All those factors may influence the amount of heat that can be delivered to the district heating network in different ways. Not only is the district heating network vulnerable but the production facility may be affected by problems as well. The boiler where heat is produced is a complex system that can break, need service or impose constraints on the production of heat in many different ways. In addition to that the system that measures data can affect the measurements in a wide range of different ways. The measurements may be contaminated by noise, offset by a bias from

a faulty calibration, lost as the consequence from a power outage or influenced in a number of different ways. Since we need to create a model that accurately describes the heat demand from the network all those possible errors in the data present a problem.

If we allow our model to learn from faulty data, it will learn an incorrect behaviour and therefore we will need to remove as many errors as possible from the data. Since the removal of errors is hard to automate when the error sources and the consequences of the errors may be so diverse, the error removal pre-processing stage will have to be done manually. Even when the pre-processing is done manually it may be hard to accurately remove errors, how should one differ between a low heat production as a result from a broken boiler and low heat production as a result from an unusually hot week in the winter? Despite those problems there may be measurements that are so obviously faulty that they will have to be removed, but caution has to be exercised in this process!

3.4.2 Standardization

Standardization is the process of converting a measurement to a dimensionless quantity. Standardized values enable us to compare values from different sources without the need to consider the unit of measurement. Standardization is particularly useful in multiple regressions since it allows direct comparison of the different regression coefficients. A common way to standardize data [2] given a sample set of values from X is shown here:

$$Z_{X_i} = \frac{X_i - \bar{X}}{\sigma_s} \tag{3.22}$$

where Z_X is the standardized variable, $\bar{X}$ is the sample mean,

$$\bar{X} = \frac{1}{n}\sum_{i}^{n} X_i \tag{3.23}$$

and σ_X the sample variance

$$\sigma_s = \sqrt{\frac{1}{n-1}\sum_{i}^{n}(X_i - \bar{X})^2} \tag{3.24}$$

Standardization is also useful when using neural networks since input standardization can reduce a tendency where inputs with large absolute values tend to affect the output more than inputs with small absolute values. Standardization in a neural network context can also reduce the risk for what is known as node saturation. Node saturation means that the output of a node is driven into a region where the derivative of the activation function is close to zero which effectively prevents the node from learning since weight adjustment no longer introduces any major change in the net output of the neuron (see Figure 3.2 for reference).

3.5 Measuring forecast accuracy

3.5.1 Error measure

Measuring error is a vital part of forecasting, but also a troublesome part. Error can be measured in a wide range of ways and there is no strict rule that

can determine a correct measurement for any given application. Several authors recommends using an error measure that reflects how a forecast error is perceived in the context where the forecast is to be used and also to compare the results with different error measures. According to operators a suitable error measure could be *mean average percentage error* (MAPE) and therefore we shall use MAPE as our main error measurement but we define two standard statistical error measures

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{e_t}{Y_t}\right|, \tag{3.25}$$

$$MSE = \frac{1}{n}\sum_{t=1}^{n} e_t^2, \tag{3.26}$$

where MSE is *mean squared error* and e_t is the *one-step forecast error* [19] defined as

$$e_t = Y_t - \hat{Y}_t \tag{3.27}$$

3.5.2 Cross validation

Cross-validation is a technique from statistics that produces an independent assessment of how well predictions from a model will generalize to previously unseen data. The foundation in cross-validation is the random split of data into two parts, one *training set* and one *test set*. When cross validation is used for training of a neural network we divide the training set again to form two new sets, one used for *parameter estimation* (training) and one for *evaluation of the model's performance* [11].

We train the network on the parameter estimation set and after every epoch we pause training and test the network on the performance evaluation set. If we find that the performance of the network as measured on the evaluation set has failed to improve from the last epoch we can reduce the learning rate from epoch to epoch to allow the network to improve its generalization. When the error on the evaluation set starts to rise, or stops to improve, again we have likely reached a minimum in the generalization error and can terminate the training.

When training has completed, the test set from the first split can be used in order to get a completely independent measure of the networks ability to generalize. The test set is needed because the error measure from the verification set will not be completely independent since the verification set has been used both to adapt parameters (learning rate) and to decide when to halt training.

4 Data

Data for this thesis was supplied from two different power plants. Data supplied by the plants include hourly mean of power output and the hourly mean of measured temperature during every hour. The units are MW and $^{\circ}C$ respectively.

Table 4.1 presents an overview of some basic facts for the two plants that supplied test data. *Max power* refers to the total power that can be supplied from all available *boilers* at the plant to the domestic heating network. This power level is rarely used in practice but could in theory be slightly exceeded since the plants are equipped with accumulators that can be used to increase net output from the plant to the network during periods of peak demand.

Table 4.1: Basic facts for the two plants

	Plant 1	Plant 2
Max power	$\sim$ 105 MW	$\sim$ 70 MW
Number of boilers	5	3
Years of raw data available	$\sim$ 3 years	$\sim$ 6 years
Number of datapoints used	15219	28261
Number of datapoints remowed	2905	20272

4.1 Plant number 1

A brief summary of the data provided from plant one can be seen in Table 4.2. Figure 4.1 shows outdoor temperature and power output from plant one as a function of time.

Table 4.2: Summary statistics for plant number 1

	Min	Max	Mean	Std. deviation	Covariance
Outdoor temp	-19.42	34.04	6.69	8.07	-0.84
Power	0	71.08	20.44	12.77	-0.84

We can see in Figure 4.1 that the measured power contains several errors. Figure 4.2 shows a magnification of the most important errors. The errors marked here persists over a long time and their nature is such that automatic treatment becomes hard. In order to improve predictions we choose to remove these marked regions from Figure 4.2. Figure 4.4 shows the processed data where a total of 2905 data pairs has been removed (as can be seen in Table 4.1). For this plant all the errors where in measured power and the source for these errors is most likely a faulty measurement system. The cross correlation ρ between outdoor temperature and output power is very strong, as showed in table 4.2.

As can be clearly seen in Figure 4.3 the data for plant one seem to show two distinctly different correlations between power and temperature. Two possible regression lines have been sketched with red in the figure for increased clarity.

It is obvious that this dual dependency shown in Figure 4.3 is problematic since it does not allow us to resolve power as a one-to-one function of temperature. Therefore the cause of this duality was traced and it turned out to be a major change in the domestic heating system. This change is not obvious when the time series is plotted but can be noted in Figure 4.1 as an increase in variance from the end of year one and on. To correct this issue all values from this period were removed.

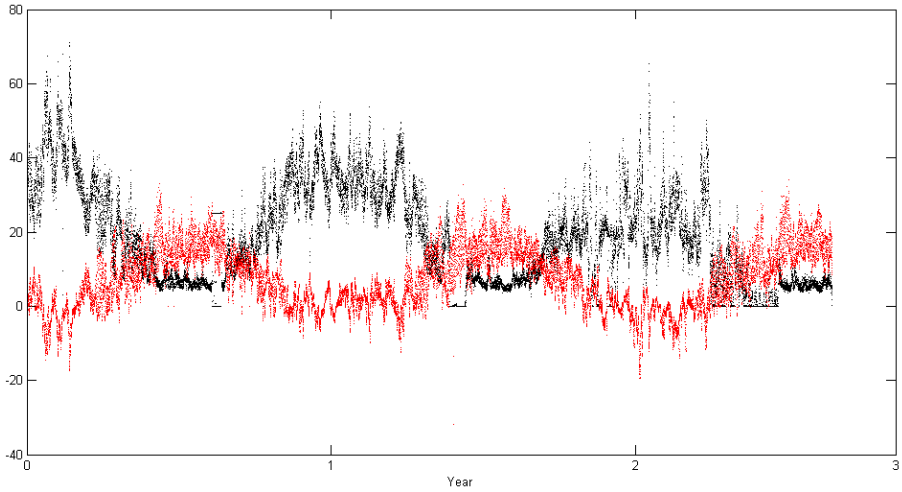


Figure 4.1: Power output (black) and measured temperature (red) for the entire dataset from plant number one

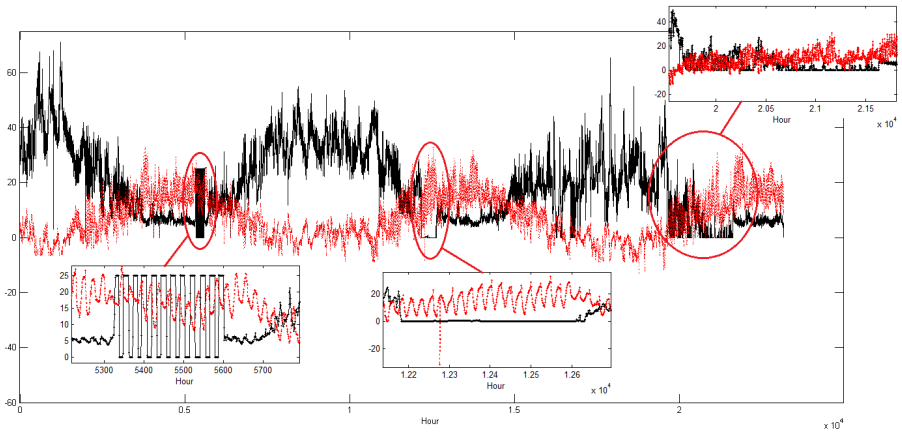


Figure 4.2: Magnification of errors for plant number one

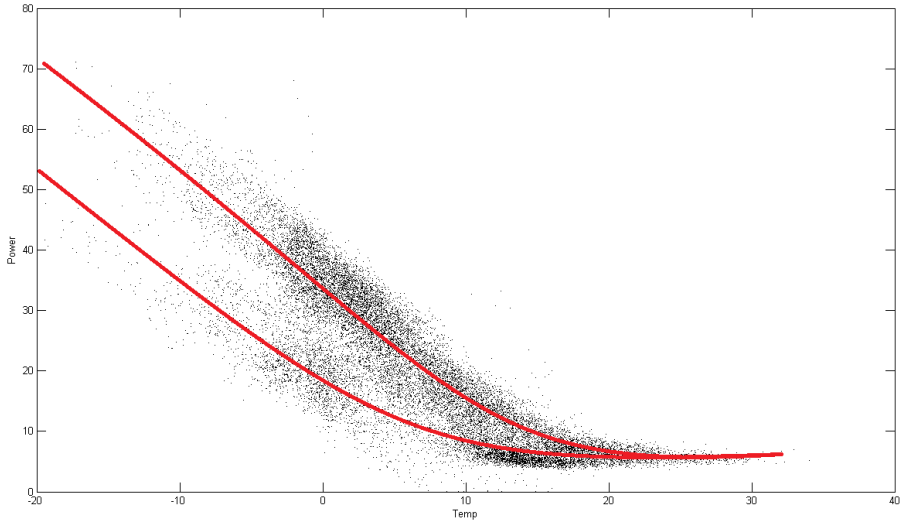


Figure 4.3: Scatterplot temp vs. power for plant one

4.1.1 Corrected data

Table 4.3 shows summary statistics for the data from plant one after all corrections were made. The most notable change is the correlation factor, ρ, between temperature and power. Before the corrections the correlation was -0.84, after the corrections the correlation has increased in magnitude to -0.92 implying that we managed to remove the duality in temperature dependency caused by the changes in the domestic heating system.

Table 4.3: Summary statistics for plant number 1, after corrections

	Min	Max	Mean	Std. deviation	Covariance
Outdoor temp	-17.29	32.95	7.27	7.50	-0.92
Power	2.62	71.08	21.62	13.07	-0.92

4.2 Plant number 2

A brief summary of the data provided from plant two can be seen in Table 4.4. Figure 4.6 shows outdoor temperature and power output from plant two as a function of time.

Table 4.4: Summary statistics for plant number 2

	Min	Max	Mean	Std. deviation	Covariance
Outdoor temp	-21.44	39.02	5.03	8.23	-0.57
Power	-0.08	47.95	13.81	9.17	-0.57

We can see in Figure 4.6 that the measured power for plant two is almost free from errors. There are some values in the first few months where a different

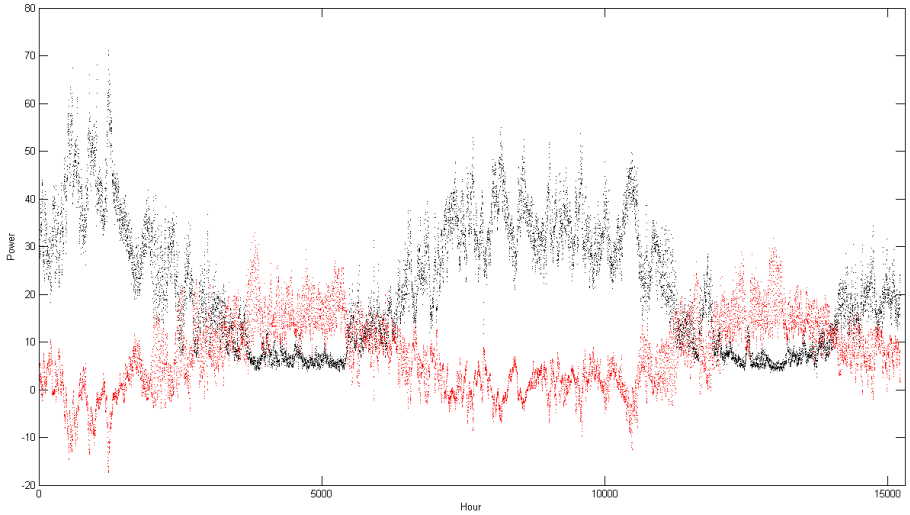


Figure 4.4: Power output (black) and measured temperature (red) for plant number one. Bad value pairs have been remove.

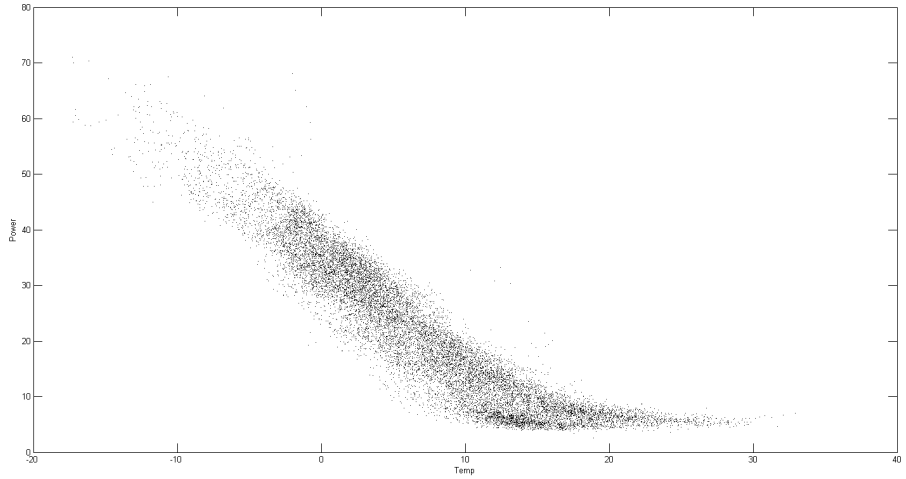


Figure 4.5: Scatterplot temp vs. power for plant one, all bad data pairs have been removed.

sampling technique was used that will have to be removed. There are also some values where an error in the measurement system have caused negative power values. These values are obviously faulty and they also occur at a time when temperature measurements are missing.

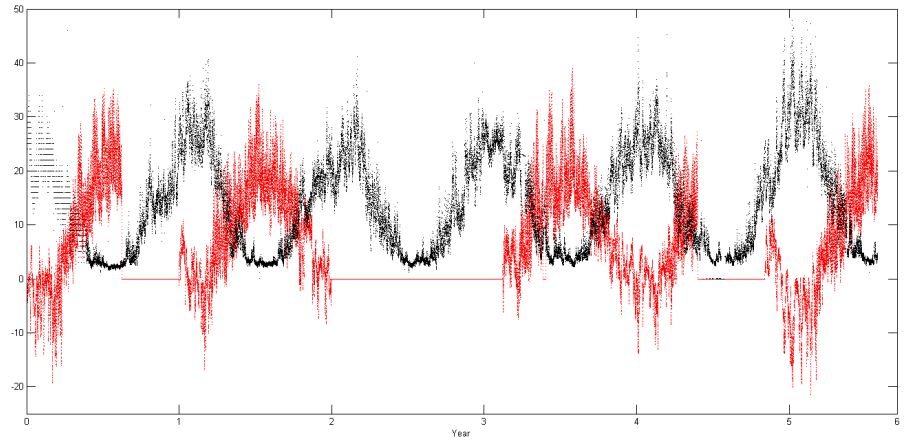


Figure 4.6: Power output (black) and measured temperature (red) for the entire dataset from plant number two.

The major problem in this dataset is that outdoor temperature measurements are missing for large parts of the data. Figure 4.6 shows the missing temperature values as a zero reading. The data pairs where temperature is missing will have to be removed in order to allow us to develop a model that explicitly takes outdoor temperature into consideration. Removing the value pairs where temperature data is missing will also result in the removal of the problematic negative power values.

4.2.1 Corrected data

Table 4.5 shows summary statistics for the data from plant two after all corrections were made. As for plant one the most notable change is the correlation factor, ρ, that increases in magnitude from -0.57 to -0.88. This increase is largely due to the removal of missing temperature values, interpreted as zeros, which of course didn't have any correlation with the output power.

Table 4.5: Summary statistics for plant number 2, after corrections.

	Min	Max	Mean	Std. deviation	Covariance
Outdoor temp	-21.44	39.02	8.50	9.04	-0.88
Power	0	47.95	14.52	9.79	-0.88

The time series resulting from the removal of all data pairs where temperature measurements are missing and all data pairs where the bad sampling technique for power measurement was used can be seen in Figure 4.7.

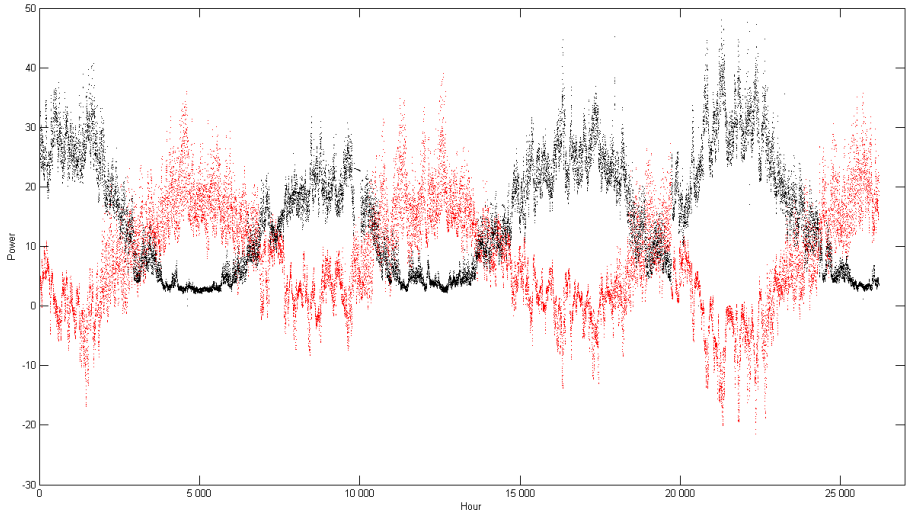


Figure 4.7: Power output (black) and measured temperature (red) for plant number two. Bad value pairs have been removed

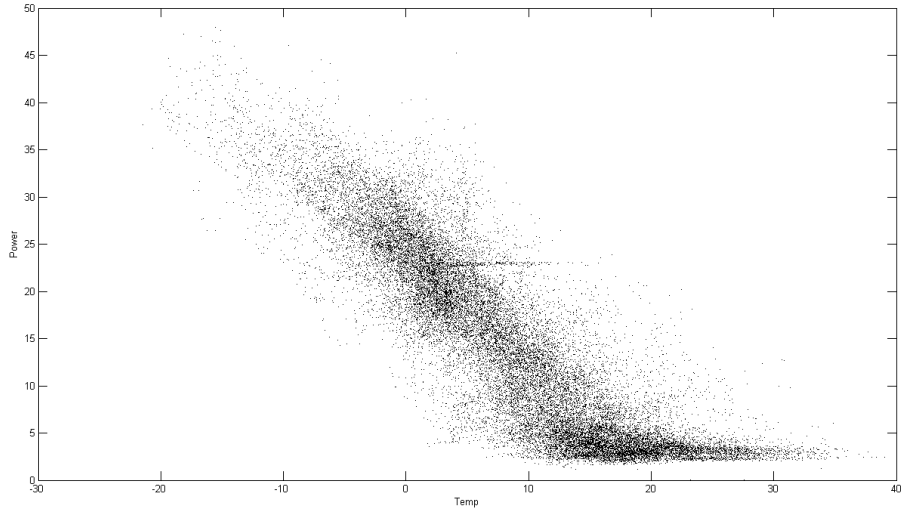


Figure 4.8: Scatterplot temp vs. power for plant two, all bad data pairs have been removed.

5 Results

To verify the forecast seven out-of-sample data sets were used from plant one and two respectively. The sets were randomly chosen out of the original datasets and the length of the set was chosen to reflect a length that might be interesting for operators to predict in real life.

Table 5.1: Out of sample sets plant one

Set	Start time	Length	Naive MAPE
1	2008-10-27 00:00	24h	0.1334
2	2007-10-21 09:00	48h	0.08248
3	2007-11-25 06:00	24h	0.06586
4	2007-06-26 06:00	24h	0.2588
5	2008-09-03 23:00	24h	0.1406
6	2008-06-18 18:00	24h	0.07609
7	2007-01-23 05:00	24h	0.08770

Table 5.2: Out of sample sets plant two

Set	Start time	Length	Naive MAPE
1	2010-01-03 20:00	24h	0.06421
2	2008-03-24 08:00	24h	0.1090
3	2009-12-12 06:00	24h	0.04014
4	2009-03-13 22:00	24h	0.1935
5	2008-12-11 03:00	24h	0.09757
6	2008-06-30 06:00	24h	0.1745
7	2008-10-03 18:00	24h	0.2323

Note that all forecasts in this section has been produced a-posteriori using actual, measured, temperature as opposed to forecasted temperature. We use measured temperatures since the weather forecasts are beyond our influence and we are only interested in a measurement of how well the developed software performs.

5.1 Preprocessing of data

5.1.1 Normalization, encoding and filtering

It is straightforward to include inputs and targets that represent a scale measurement such as power or temperature. These measurements can simply be normalized using (3.22) and then they are ready to be used as input and target respectively.

Date and time information is more problematic to include since that type of information somehow has to be interpreted numerically. There are two common ways to encode nominal values in such a manner that they can be used as inputs to a neural network. The first and most obvious is to simply represent them as a number, in the case of weekdays we can represent Monday as zero and Sunday

as seven or use any other well defined scheme to map a day to a number. One of the major drawbacks with mapping nominal cases numerically is that the transformation imposes an ordering on the variables that is not likely to map to a true ordering of the variables. If we map Monday as zero and Sunday as seven we are implicitly ordering the variables and therefore the neural network may reflect this false ordering in its output. An alternative way of mapping date and time information for use as input to a neural network is to use a binary representation. If we wish to represent the day of the week as an input to the neural network we may create seven distinct input neurons and feed each neuron with a binary input which represents the current day of the week.

Both methods of representing date and time information have been tried out and it was found that encoding date and time information binary did not yield better prediction results, but came at the cost of a significant increase in network complexity and training time. Therefore only numeric encoding of date and time information will be used.

5.2 Selecting the number of hidden neurons

In order to get an indication of how many hidden neurons that are needed to model the particular problem while preventing over-fitting three neural networks with different number of hidden neurons were fitted to test data from plant one and plant two. The results from forecasting for the different test sets are presented in Table 5.3 and Table 5.4.

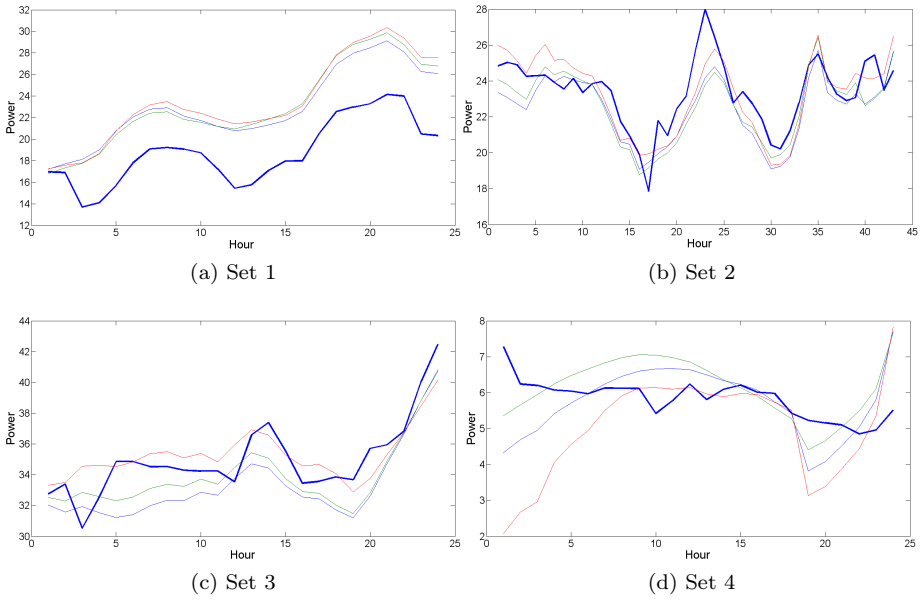
Table 5.3: Results with different number of hidden neurons, plant one

	20 neurons		10 neurons		5 neurons	
Set	MAPE	MSE	MAPE	MSE	MAPE	MSE
1	0.2258	19.07	0.2347	21.71	0.2541	25.10
2	0.04871	2.012	0.04666	1.963	0.04400	1.442
3	0.05125	4.038	0.03820	2.407	0.03028	1.950
4	0.1258	1.067	0.1231	0.8171	0.1927	3.061
5	0.1407	1.832	0.1515	2.1754	0.1087	1.144
6	0.08220	0.4778	0.08491	0.5446	0.1043	0.7656
7	0.03343	4.542	0.02511	2.630	0.03167	3.542

For plant one it is worth noting that none of the three models with varying number of hidden neurons presented above differ much for any of the test sets except for set number four where the model with five neurons get a considerably higher error.

Figure 5.1 and Figure 5.2 reinforce the interpretation of Table 5.3 where there seems to be no major difference between the three models on the test sets from plant one. Judging by the figures all three models seem to have captured approximately the same high level behaviour from the data with some variations in the exact level for the prediction.

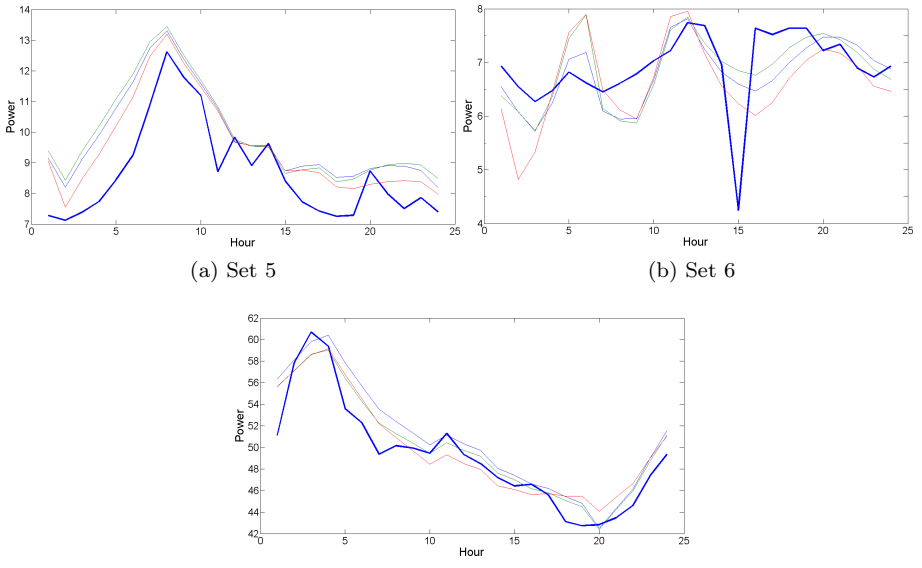
If we compare the results for plant one (Table 5.3) to the results for plant two (Table 5.4) we can see that the models for plant one on average are more successful for generalization. When considering the results for plant two it is important to note that test set number four contains a major anomaly (see



(a) Set 1 (b) Set 2

(c) Set 3 (d) Set 4

Figure 5.1: Neural network forecasting output for different number of hidden neurons, plant one. Thick blue line corresponds to the target, the green line to the model with 20 neurons, the cyan to the 10 neuron model and the red to the 5 neuron model.



(a) Set 5 (b) Set 6

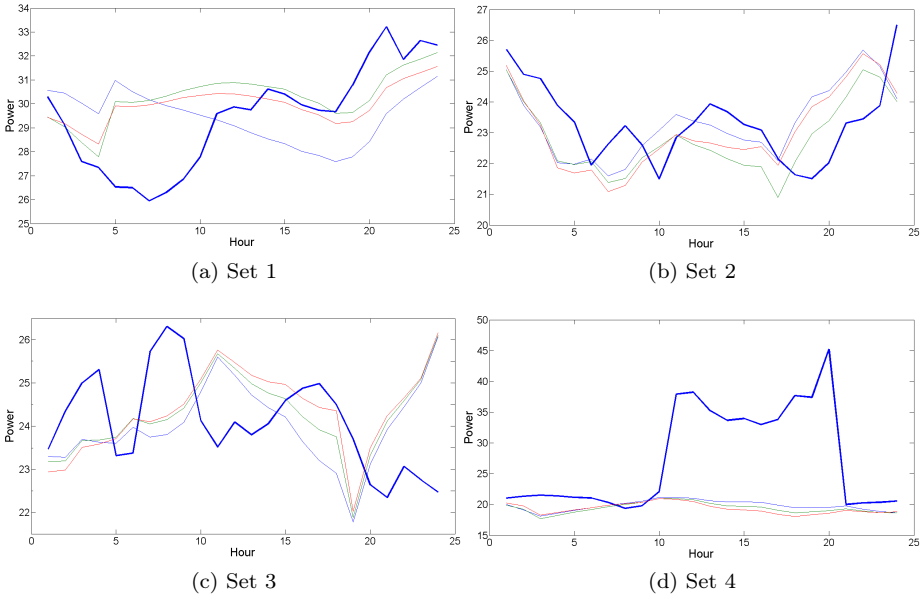
(c) Set 7

Figure 5.2: Neural network forecasting output for different number of hidden neurons, plant one. Thick blue line corresponds to the target, the green line to the model with 20 neurons, the cyan to the 10 neuron model and the red to the 5 neuron model.

Table 5.4: Results with different number of hidden neurons, plant two.

	20 neurons		10 neurons		5 neurons	
Set	MAPE	MSE	MAPE	MSE	MAPE	MSE
1	0.07886	6.559	0.05218	4.062	0.05409	3.949
2	0.05043	1.974	0.0486	1.601	0.05266	1.972
3	0.05681	2.534	0.05489	2.321	0.05554	2.399
4	0.2252	118.8	0.2369	127.5	0.2375	133.7
5	0.04288	1.319	0.04917	1.880	0.05014	2.091
6	0.2740	2.664	0.2641	2.709	0.29861	3.229
7	0.07866	1.330	0.08619	1.493	0.09105	1.676

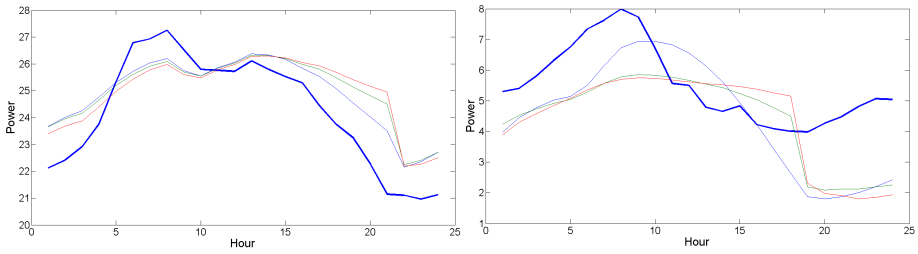
Figure 5.3d) which impacts the results. Test set four has still been included since it serves as a good illustration of some problematic characteristics in the data from plant two. Test set number six for plant one corresponds to a summer time low load scenario and as for low load test sets four and six (Figure 5.1d and 5.2b) from plant two we see an increase in MAPE on this test set compared to the others from the respective plants.



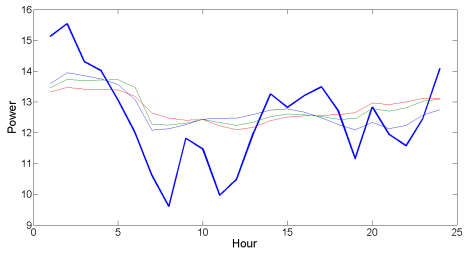
(a) Set 1 (b) Set 2

(c) Set 3 (d) Set 4

Figure 5.3: Neural network forecasting output for different number of hidden neurons, plant two. Thick blue line corresponds to the target, the green line to the model with 20 neurons, the cyan to the 10 neuron model and the red to the 5 neuron model.



(a) Set 5 (b) Set 6



(c) Set 7

Figure 5.4: Neural network forecasting output for different number of hidden neurons, plant two. Thick blue line corresponds to the target, the green line to the model with 20 neurons, the cyan to the 10 neuron model and the red to the 5 neuron model.

5.3 Selection of additional input data

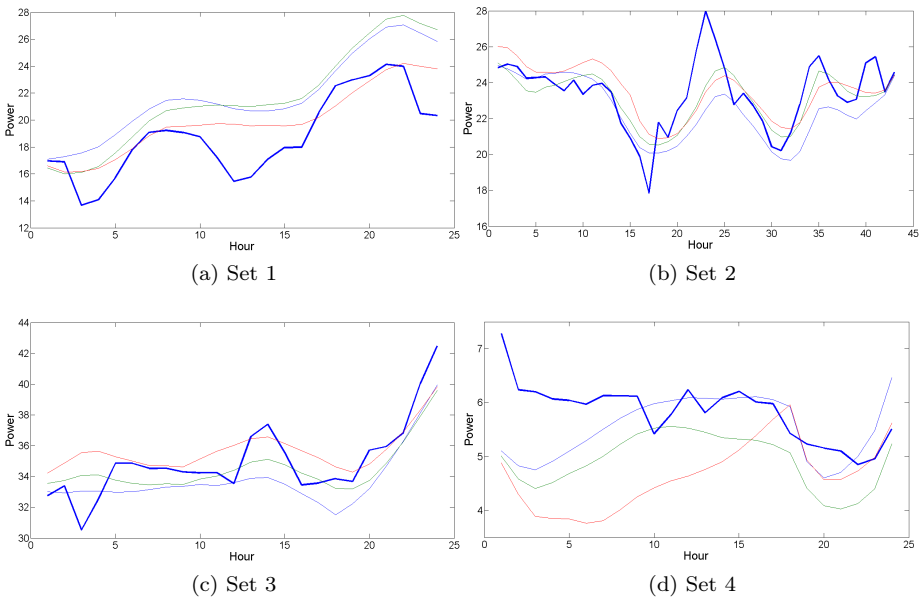
Since the load series is expected to be auto-regressive the inclusion of time lagged values as inputs might be able to improve the models predictive ability. In order to explore which inputs might help improve the network's ability to generalize several models with varying time lagged inputs and filters were fitted to the data and evaluated on the out of sample sets.

Table 5.5: Results with different time-lags of power values, plant one.

	Only B^1		B^1 and B^{24}		B^1, B^{24} and B^{168}	
Set	MAPE	MSE	MAPE	MSE	MAPE	MSE
1	0.1663	11.15	0.1087	6.326	0.08782	3.927
2	0.05228	3.07289	0.03872	1.573	0.05276	2.396
3	0.04156	2.948	0.03372	2.111	0.03608	2.884
4	0.09798	0.6118	0.1532	1.097	0.2050	2.211
5	0.1161	1.381	0.06589	0.5242	0.06065	0.7129
6	0.1019	0.8350	0.1211	0.9718	0.1279	1.400
7	0.03820	5.177	0.04610	7.181	0.05317	7.571

Table 5.6, Figure 5.7 and Figure 5.8 shows how the result differ when including a varied number of time lags, B^1, B^{24} and B^{168} as defined in (2.7), into the model for plant one. As can be clearly seen no single model performs best for all test sets but the model with B^1, B^{24} and B^{168} performs better or

comparable to the two other models on every test set except for set four and to some extent set six. It is worth noting that while the model with B^1, B^{24} and B^{168} gives a higher MAPE error for set four and six the MSE error for the models predictions is of roughly the same magnitude as for the other test sets.



(a) Set 1 (b) Set 2

(c) Set 3 (d) Set 4

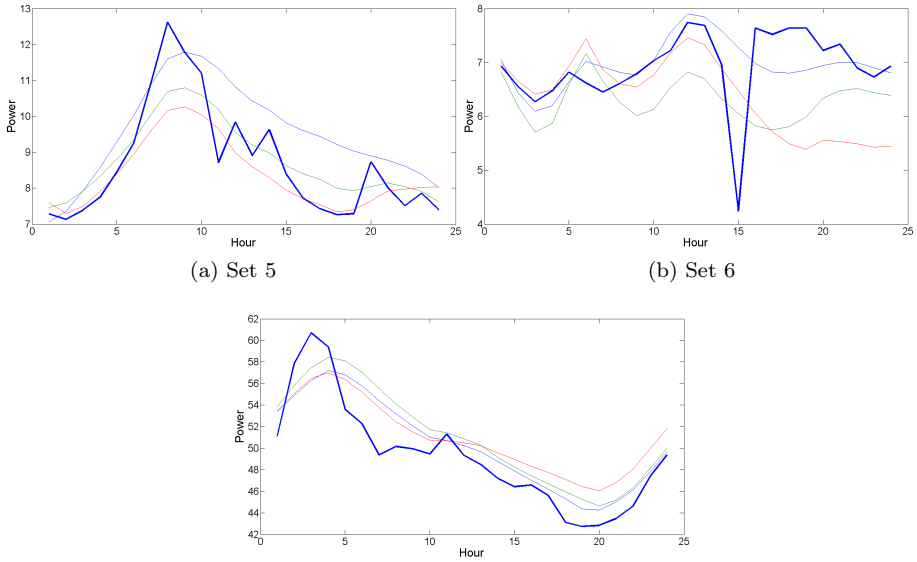
Figure 5.5: Neural network forecasting output for different number of time lags, plant one. Thick blue line is the target, thin blue line is the model with B^1, the cyan B^1 and B^{24} and the red B^1, B^{24} and B^{168}.

If we study Figure 5.5d (Set 4) we can see that all models seem to have a much worse fit on this test set then on the other test sets. Looking at Figure 5.6b (Set 6) we see clearly that the target value has an unexpected drop at hour 15 which contribute to give larger errors on that test set. We can also note, once again, that both set four and set six corresponds to low total power output scenarios.

Table 5.6: Results with different time-lags of power values, plant two.

	Only B^1		B^1 and B^{24}		B^1, B^{24} and B^{168}	
Set	MAPE	MSE	MAPE	MSE	MAPE	MSE
1	0.1324	19.11	0.1440	22.42	0.1034	11.18
2	0.08047	4.456	0.1392	12.60	0.1223	10.57
3	0.04301	1.405	0.04846	1.968	0.04191	1.225
4	0.1997	83.77	0.2013	85.08	0.2052	80.03
5	0.08047	5.431	0.08033	5.308	0.08292	5.441
6	0.1166	0.8507	0.1862	1.504	0.1805	1.343
7	0.09143	1.853	0.1036	2.243	0.08812	1.785

Just as for the case with no time lagged values the results for plant two are

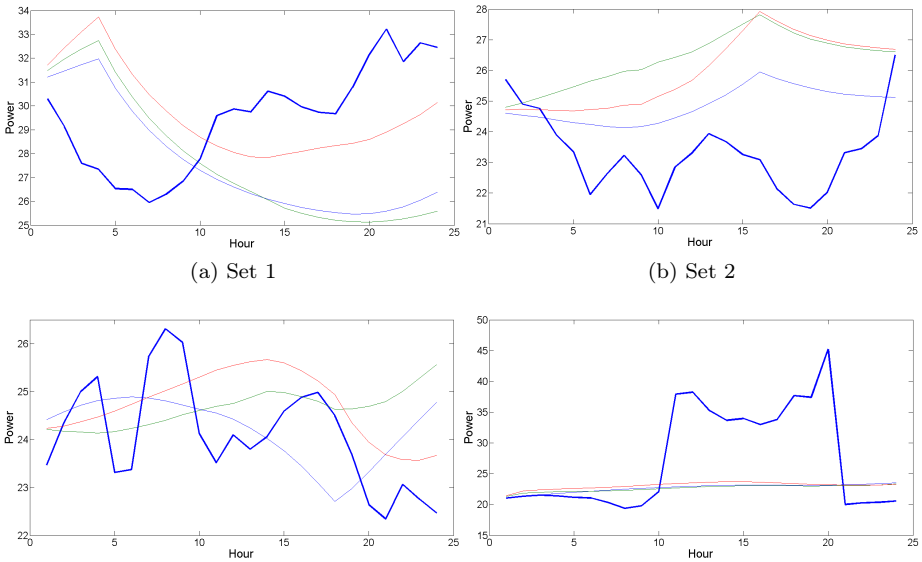


(a) Set 5

(b) Set 6

(c) Set 7

Figure 5.6: Neural network forecasting output for different number of time lags, plant one. Thick blue line is the target, thin blue line is the model with B^1, the cyan B^1 and B^{24} and the red B^1, B^{24} and B^{168}.



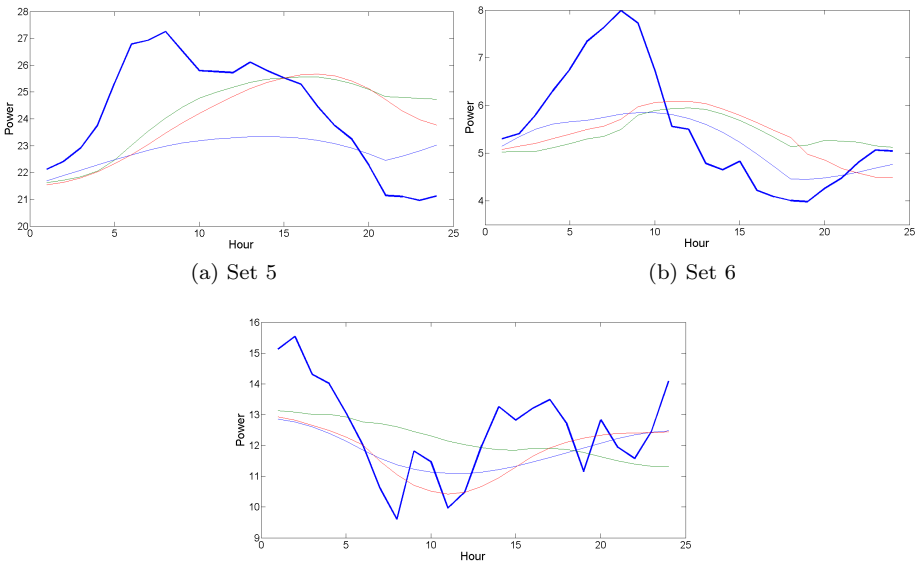
(a) Set 1

(b) Set 2

(c) Set 3

(d) Set 4

Figure 5.7: Neural network forecasting output for different number of time lags, plant two. Thick blue line is the target, thin blue line is the model with B^1, the cyan B^1 and B^{24} and the red B^1, B^{24} and B^{168}.



(a) Set 5

(b) Set 6

(c) Set 7

Figure 5.8: Neural network forecasting output for different number of time lags, plant two. Thick blue line is the target, thin blue line is the model with B^1, the cyan B^1 and B^{24} and the red B^1, B^{24} and B^{168}.

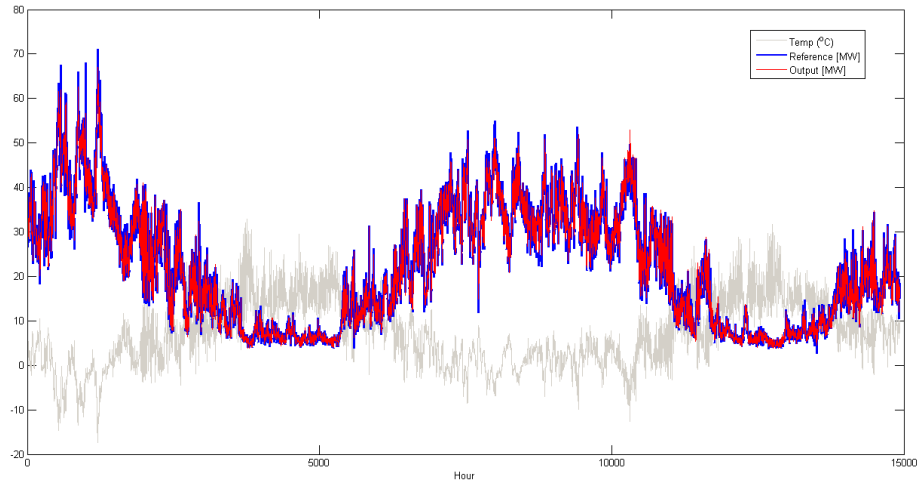


Figure 5.9: Output from final neural net model for the entire dataset from plant one, including points from which the model was estimated. The gray line is the outside temperature, the blue line is the measured output power and the red line is the output from the neural network model.

not as good as the results for plant one. For plant one we saw a significant reduction of both MAPE and MSE when using time lagged values as input but for plant two the error reduction is not as large, nor does the error fall on all test sets. Judging from the sharp transitions between different days as can be seen especially in Figure 5.7a and Figure 5.7b it seems possible that the model has over-fitted the dataset. Despite this adjustment of the early stopping, parameters does not change the fitted model much and when early stopping is set too aggressive the network fails to learn anything at all from the data so perhaps the problem is not due to over fitting after all.

To increase prediction accuracy further a number of other inputs, based either on mean power for the past x hours or the current deviation from the mean temperature for the past y hours.

5.3.1 Results with explicit temperature modelling

Both dataset show a strong correlation between temperature and power. Therefor it could be suitable to try and model this dependency in an explicit way. It is obvious from Figure 4.5 and 4.8 that the relation between power and temperature is non-linear.

It is natural to expect that production is almost independent of outside temperature when the temperature is at the highest or lowest levels. When outside temperature is high almost all consumption will be due to tap water heating which can be regarded as temperature independent. When outside temperature is really low, all boilers will be operating at max capacity and therefore a further decrease in temperature will not increase heat production.

This behaviour where the generated power tends to some constant when the temperature becomes high and some other constant when the temperature becomes low could be modelled using a sigmoid function with coefficients as in equation 5.1.

$$f(x) = \frac{a}{1 + e^{-bx}} + c \tag{5.1}$$

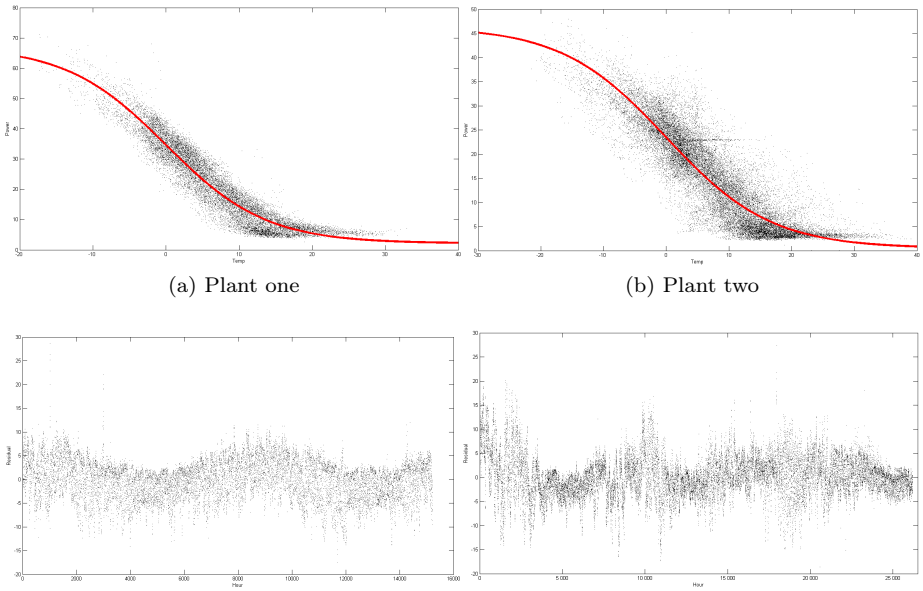
Estimations for a, b and c for the two datasets are shown in table 5.7. The corresponding regression line is plotted on top of a scatter plot for the two datasets in Figure 5.10a and 5.10b respectivley. The two lower plots in Figure 5.10 shows the residual from the two fitted models.

Table 5.7: Estimated parameters and godness of fit for sigmoid function.

	a	b	c	R^2	RMSE
Plant 1	-63.04	0.1500	66.25	0.9636	2.492
Plant 2	-45.91	0.1196	46.39	0.8589	3.603

Table 5.7 shows that the selected sigmoid model for temperature dependency is a better fit for the dataset from plant one than for the dataset from plant two. However, with a R^2-value of 0.9636 for plant one and 0.8589 for plant 2 the model can be considered a reasonably good fit for both data sets. The fit could likely be improved by using a model with more degrees of freedom,

but the sigmoid function is a good match for the a priori expectations on data behaviour, and the good fit seen on actual data makes it a good choice.



(a) Plant one

(b) Plant two

(c) Adjusted values (residuals) plant one

(d) Adjusted values (residuals) plant two

Figure 5.10: Fitted regression model and corresponding adjusted series.

What is slightly surprising is that there is such a big difference in how the residuals behave if we compare plant one to plant two. For plant one the residuals (see Figure 5.10c) seem to vary seasonally as smooth curve with a slightly higher mean level during the winters than during the summer. Despite this seasonality that obviously could not be explained by temperature difference alone the temperature corrected variance seems to be largely constant during the year. For the residuals from plant two (see Figure 5.10d) on the other hand there is slight evidence for the same seasonal change in level but the variance behaves very differently here. The variance seems to change abruptly between different time periods for no obvious reason. This change in variance might indicate a non-stationarity that can be problematic to modell and this can help explain the problems that occurred while modelling this dataset.

When the explicit temperature curve had been fitted a neural network was trained on the residuals in an attempt to model them. When the residuals had been modelled the fitted temperature was added back in and the errors were calculated. The results from this modelling is presented in Table 5.8 and Table 5.9.

5.4 Developed software

The goal of this thesis was to develop an application that could be used in every day operation to predict heat demand for the district heating network. To be used daily the application has to be reasonably fast and it also has to be simple enough so that the operators do not feel they are wasting their time using it.

Table 5.8: Results from explicit temperature modelling, plant one.

Set	MAPE	MSE
1	0.2415	22.0564
2	0.0535	2.2658
3	0.0908	4.1548
4	0.1582	3.0188
5	0.3698	13.6094
6	0.5725	19.8380
7	0.2124	111.8

Table 5.9: Results from explicit temperature modelling, plant two.

Set	MAPE	MSE
1	0.08634	7.741
2	0.05577	2.435
3	0.05711	2.383
4	0.2060	104.6
5	0.03405	0.9301
6	0.1752	1.179
7	0.08636	1.616

The application will also have to be able to adapt to different circumstances. If someone wants to use e.g. wind forecast as an explanatory variable they should easily be able to do that without modification of the application.

ANNs were considered a good fit for the demands of good predictive power. Simple and fast user experience once set up and the possibility to include a wide range of different explanatory variables without the need for complex or time consuming model building it was decided that ANNs should be used for the developed software.

The implemented ANN is of the multi-layered, fully connected, feed-forward type. The number of hidden neurons and the activation function used in the neurons can be customized at runtime to best suit the current setting. The implementation uses stochastic gradient descent with momentum to optimize the weights. Figure 5.12 shows a schematic view of the neural network structure and the inputs that was used. The inputs *Hour of day*, *Day of week* and *Month* was coded numerically as $0-24$, $1-7$ and $1-12$ respectivley. Outside *temperature* is the only measured explanatory variable that is used but if other explanatory variables were to be used they would be included in the same way. When predictions are made for more than one hour ahead the first predictions are recursively feed back to the network as inputs to further predictions as needed.

For training, the dataset is divided into one training and one verification part. The order of the training part is randomized and patterns are presented one by one and after each training pattern presentation the weights are updated. The stochastic gradient descent has the advantage over batch gradient descent that it is less likely to get stuck in a local error minimum. When all patterns

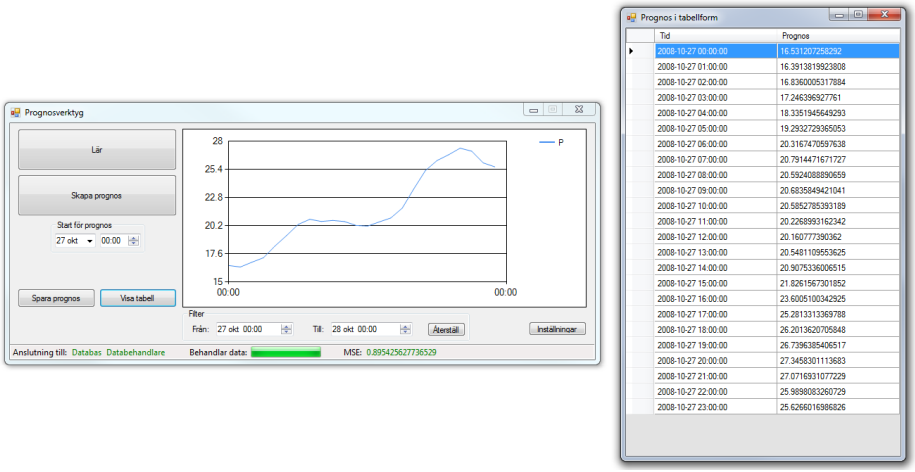


Figure 5.11: User interface for the neural predictor software with a forecast produced for 24 hours ahead.

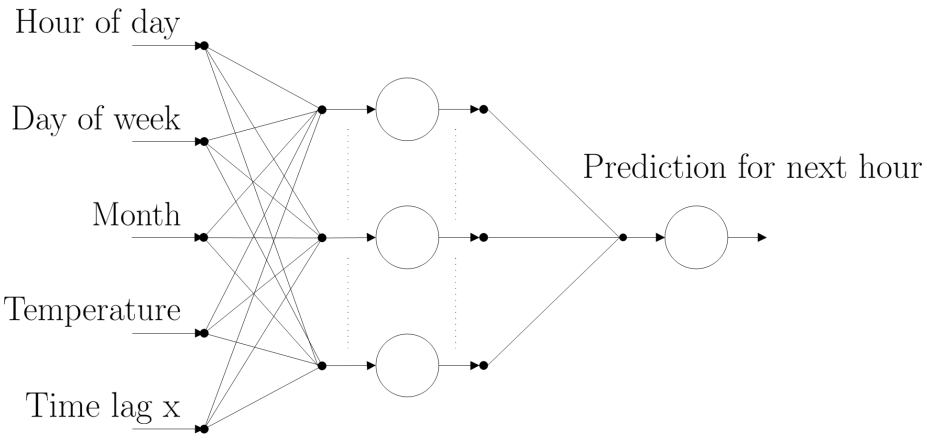


Figure 5.12: The network structure used in the neural network application.

in the training set have been presented and the weights have been updated accordingly, the error is calculated for the verification set.

After each error calculation for the verification set, the error is compared to the error in the last epoch. The error for the verification set is only allowed to increase for a fixed number of epochs before the weights in the network are restored to the values they had before the verification error started to increase. This limitation prevents over-fitting and attempts to prevent the network from learning noise in the training set. When training has failed to decrease error on the verification set further and therefore has restored the weights the learning rate is decreased and the training is continued to allow further fine-tuning of the weights. When the learning rate has decreased beyond a certain threshold and the error for the verification set starts to increase again the weights are restored a final time and training is terminated.

6 Discussion

This report considers short-term forecasting of heat-load in a domestic heating system. Short term is considered to be from one hour to one week ahead and the main focus is on the 24 hours ahead forecast. The predictions are performed for two distinct data sets using the developed software with a neural network approach. The neural network is of the fully connected, feed forward type. The network is trained using stochastic back propagation with gradient descent and momentum.

The developed software has been tested on two real life datasets using several different settings. It was shown that varying the number of neurons used in the model did not result in large changes of the error produced by the model. This is seen as an indication that the system used for early stopping works well on actual data. It is also seen as an indication for the model's ability to find a global minimum and consequently converge to this minimum with good performance. It was also shown that explicit modelling of the load's dependence on temperature where a neural network were used to predict the temperature corrected load series did not improve the predictions. On the contrary, for one of the data sets, the predictions was much worse. This indicates that the neural network is able to learn a more complex load-temperature relation than can be modelled using simple regression.

There is a tendency that the MAPE error from low load times (e.g. during the summer) is larger. This behaviour is expected since the load will not depend on outside temperature during hot summer days. Therefore the nature of the load will be more stochastic and thereby harder to predict. Another reason for the larger errors during the summer could be that some of the hour-to-hour variation from winter months that the network learns well is reflected in the summer months as well. Perhaps this tendency could be reduced by creating one network for each month or creating one network for low load scenarios and one for high load scenarios. The problem with this approach is that it reduces the amount of data that is available to train each network and that it increases model complexity.

For data set one the neural network model arrives at very good results and the results become's even better when time lagged power terms are included into the model. For the other dataset the results are not as good and neither do they improve so much when time lagged power terms are included. This is likely due to weaker correlation between load and temperature for this dataset and the fact that this dataset contains much noise and measurements that could possibly be considered faulty. This big difference between the two datasets points to one of the weaknesses of using a data-driven approach to model future loads as compared to using a deterministic model - the data-driven approach is highly dependent on the quality of available data.

Possible improvements to the models that would be interesting to try in the future include creating load forecasts from ensemble weather prediction and using other explanatory variables to complement temperature. The use of additional explanatory data such as wind speed and solar radiation is an obivious suggestion and it is likley to improve predictions, but the historical data has to be available to train the network.

References

[1] L. Arvastson. *Stochastic modeling and operational optimization in district heating systems*. Lund, 2001. Diss. Lund : Univ. 2001.

[2] Gunnar Blom and Björn Holmquist. *Statistikteori med tillämpningar*. Studentlitteratur, Lund, 3., [omarb. och utvidgade] uppl. edition, 1998.

[3] George E. P. Box and Gwilym M. Jenkins. *Time series analysis forecasting and control*. San Francisco, 1970.

[4] G.E.P. Box, G.M. Jenkins, and G.C. Reinsel. *Time series analysis: forecasting and control*. Wiley, Oxford, 4. ed. edition, 2008.

[5] Vladimir S. Cherkassky and Filip M. Mulier. *Learning from data : concepts, theory and methods*. Wiley, New York, 1998.

[6] Georges A. Darbellay and Marek Slama. Forecasting the short-term demand for electricity: Do neural networks stand a better chance? *International Journal of Forecasting*, 16(1):71 – 83, 2000.

[7] X. Ding, S. Canu, and T. Denoeux. Neural network based models for forecasting. In *Proceedings of ADT'95*, pages 243–252. Wiley and Sons, 1995.

[8] W. Duch and N. Jankowski. Survey of neural transfer functions. *Neural Computing Surveys*, 2(1):163–212, 1999.

[9] Z. Ghahramani. Unsupervised learning. *Lecture Notes in Computer Science*, 3176:72–112, 2004.

[10] S. Grosswindhager, A. Voigt, and M. Kozek. Online short-term forecast of system heat load in district heating networks. Institute of Mechatronics, Vienna University of Technology.

[11] Simon S. Haykin. *Neural networks : a comprehensive foundation*. Macmillan, New York, 1994.

[12] John Hertz, Richard G. Palmer, and Anders Krogh. *Introduction to the theory of neural computation*. Addison-Wesley, Redwood City, Calif., 1991.

[13] H.S. Hippert, C.E. Pedreira, and R.C. Souza. Neural networks for short-term load forecasting: a review and evaluation. *Power Systems, IEEE Transactions on*, 16(1):44 –55, feb 2001.

[14] Ching-Lai Hor, Simon J Watson, and Shanti Majithia. Daily load forecasting and maximum demand estimation using arima and garch. *2006 International Conference on Probabilistic Methods Applied to Power Systems*, pages 1–6, 2006.

[15] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[16] A.K. Jain and K.M. Jianchang Mao Mohiuddin. Artificial neural networks: a tutorial. *Computer*, 29(3):31 –44, mar 1996.

[17] Anders Lansner. Neuronnätverk. "www.ne.se/neuronnatverk". Accessed March 27, 2012.

[18] K. Lindqvist and N-O. Sellin. Fjärrvärmeteknik. Technical report, Högskolan i Eskilstuna/Västerås, 1985.

[19] Spyros Makridakis, Steven C. Wheelwright, and Rob J. Hyndman. *Forecasting : methods and applications*. Wiley, New York, 3. ed. edition, 1998.

[20] Miguel Moreira and Emile Fiesler. Neural networks with adaptive learning rate and momentum terms. Idiap-RR Idiap-RR-04-1995, IDIAP, Martigny, Switzerland, 10 1995.

[21] R. Rojas. *Neural Networks: A Systematic Introduction*. Springer-Vlg, Berlin, 1996.

[22] M. Sforna and F. Proverbio. A neural network operator oriented short-term and online load forecasting environment. *Electric Power Systems Research*, 33(2):139 – 149, 1995.

[23] Dipti Srinivasan, A.C. Liew, and C.S. Chang. A neural network short-term load forecaster. *Electric Power Systems Research*, 28(3):227 – 234, 1994.

[24] Sonya Trad. Svensk fjärrvärme - statistik och pris. "www.svenskfjarrvarme.se/Statistik--Pris". Accessed March 27, 2012.

[25] W. Vandaele. *Applied time series and Box-Jenkins models*. Academic Press, Orlando, Fla., 1983.

A Visualization of data flow

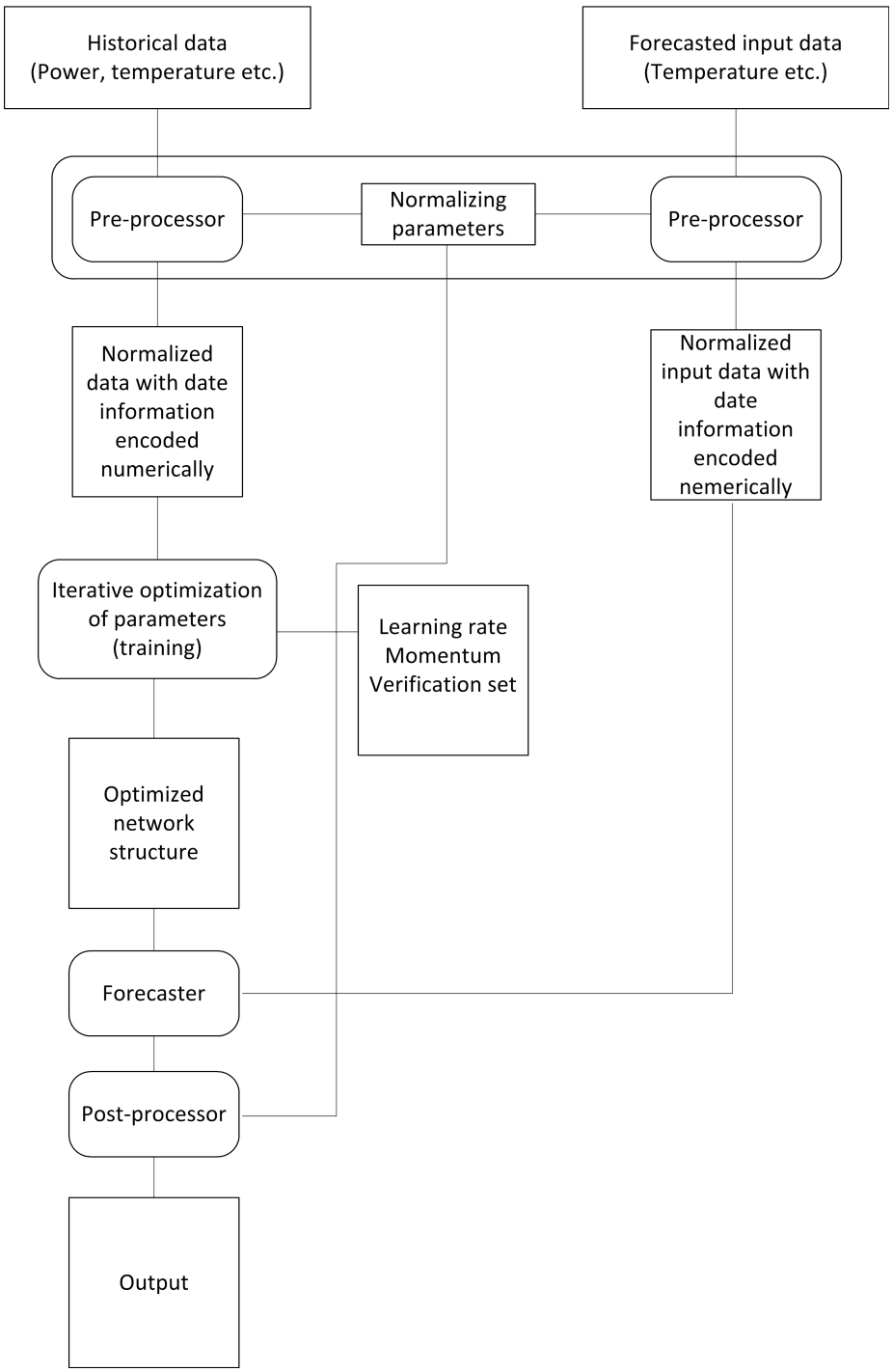


Figure A.1: Simplified visualization of how data is treated by the neural network predictor.

B Manual for the developed software

- Configure the application by altering required values in the application.settings file (only required at first startup).
- Start the program and select which filters to apply to input data.
- Press "Lär" ("Learn") to load data from the database or file specified in application.settings.
- The application will now read data, create a network and start to process the data (training) to configure the networks parameters, when training has finished a message will show the mean squared error (MSE) of the network .
- The network is now ready to forecast data. Select the start date and time for the forecast. If the network was trained using any filters a database with up-to-date values of the filtered inputs must be available or forecasting will fail.
- Press "Skapa prognos" ("Create forecast") to create a forecast. The forecasted values will be showed in the plot window.
- If closer inspection of the forecast is required, press "Visa tabell" ("Show table") to show the forecast as a spreadsheet.