4372

# Development of a Generalized Neural Network Model to Detect Faults in Building Energy Performance—Part I

**Marcus R.B. Breekweg** **Peter Gruber, Ph.D.** **Osman Ahmed, Ph.D., P.E.**
*Member ASHRAE*

## ABSTRACT

*A building energy management system (BEMS) generally monitors and manages energy usage in commercial buildings. With the ability to monitor a plant and to recall the collected data at a later time, actual building energy performance can be measured and compared with the expected performance. The comparison will help in detecting possible abnormalities with the building energy usage and in identifying opportunities to optimize the building energy performance.*

*In order to predict expected building energy performance, a reasonably accurate building energy model is needed. The building energy model is complex, multi-dimensional, and nonlinear. The relationship between building occupation and environmental conditions and the whole building energy usage can be established by using nonlinear models, e.g., artificial neural networks. There are many publications on using neural networks to estimate the energy usage. Most of these studies on load estimation are, however, based on one single data set. For such a neural network model to be commercially attractive, the model must be directly applicable to any building without any major design adjustments. The objective of this study is to design a neural network model that is generally applicable without major design adjustments and, at the same time, provides an estimation that is accurate enough for commercial purposes.*

*This paper consists of two parts. Part I discusses various neural network methods, their design issues, benefits, and limitations. The discussions create a framework for selection of neural network models and their application to real building data, which is covered in Part II, along with results and analysis.*

## INTRODUCTION

Over past decades, energy costs have increased throughout the world (IEA 1982). During this period, considerable savings have been demonstrated through better control of building energy usage (Levermore 1992). A key factor to obtain better control is supervision of the energy usage. With information on the current and past energy usage on all levels of the building, from components to whole buildings, building management can be improved.

In general, energy usage in commercial buildings is supervised by means of a building energy management system (BEMS). Two of the benefits of a BEMS are (1) its ability to collect and communicate relevant data from a remote site to a central station and (2) its capability of automatically monitoring and controlling a plant. The building energy performance is usually improved by implementing control strategies based on information obtained by the BEMS. Typical energy savings between 10% and 20% are reported for commercial buildings by implementing a BEMS (Levermore 1992).

For a BEMS to reach its full potential, information on the performance has to be available. Collection of (or a part of) the relevant data is always incorporated in a BEMS and the building operator is generally responsible for data analysis. However, developing a building energy prediction model as a BEMS add-on tool to provide automated energy performance evaluation is possible. This evaluation will assist the building operator detect irregularities in the building performance early on and thereby optimize the building's energy cost-effectively.

In recent years, a nonlinear modeling technique known as neural network has been widely used for building thermal load prediction. Conventional prediction of the building thermal load occurs through developing complex mathematical

**Marcus R.B. Breekweg** is a client partner at Cambridge Technology Partners, Amsterdam, the Netherlands. **Peter Gruber** is a senior scientist at Siemens L&S Division, Zug, Switzerland. **Osman Ahmed** is a senior principal engineer at Siemens Building Technologies, Buffalo Grove, Illinois.

models on the building's thermal behavior and components or through various statistical methods. Both of these methods are complex, labor intensive, and require special knowledge that is difficult to obtain within the building owner/operator community. In contrast, neural networks have proven extremely effective at capturing the building thermal characteristics by learning from past patterns of actual thermal load and observed inputs, such as weather variables and occupancy schedule. The computational algorithm for a neural network is simple and does not require any special knowledge from the end user.

In an ASHRAE-sponsored competition to predict hourly energy usage, neural networks outperformed statistical methods, including both linear and nonlinear methods (Kreider and Haberl 1994). The top two methods were Baysian nonlinear modeling and a feed-forward multi-layer perceptron. Ferrano and Wong (1990) used a back-propagation neural network to predict hourly cooling load in order to develop a control strategy for ice thermal storage. Gibson and Kraft (1993) used a recurrent neural network in order to predict a building's steady-state and peak electric demands. In a second energy predictor shootout, again sponsored by ASHRAE (Haberl and Thamilseran 1996), a neural network proved to be most accurate in predicting hourly thermal load. Kawashima et al. (1995) compared an artificial neural network with conventional statistical means of linear modeling, autoregressive moving average and exponential weighted moving average, to predict building thermal load for the next 24 hours. The simple back-propagation neural network that predicted the thermal load was the most accurate compared to the statistical methods. Based on published research and observations, only neural networks were considered as the nonlinear modeling method in this study.

The building energy prediction add-on tool using a neural network is usually implemented in the following two phases (see Figure 1):

- In the first phase, the historical data of measured energy and dependent variables are used to train the model. The purpose of the training is to ensure that the model is capable of capturing the building energy characteristics. The first phase is often referred to as the *training phase* and the historical data as the *training data*. The quality of training data is an important issue, as training with poor quality data will hide faulty system behavior. Data prefiltering and processing to ensure that the systems do not have any faults in the beginning is suggested by establishing a benchmark training database. The study presented here did not apply any specific methodology for identifying faults in the training data. The training data limitations and faults were only discovered as a result of investigating poor prediction.
- In the second phase, the model is used to predict the building energy consumption by applying the most recent past data of selected independent variables, such as weather input and building occupancy. The predicted energy is then compared with the measured energy consumption for the detection of possible abnormalities or faults. Continuous execution and integration of both training and prediction phases is preferable. However, the success of continuous adaptation to the building energy performance characteristics is highly dependent on the quality of the training data as discussed above.

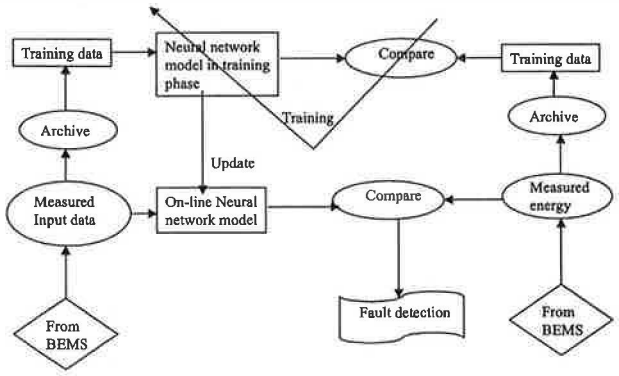


***Figure 1*** *Overall method of detecting building energy performance using a neural network.*

## SCOPE

While there are many papers concerning load modeling and forecasting, most of them propose models that are optimized for a specific data set or building. It is very time-consuming and, therefore, commercially unacceptable to design a new model for every new building. The costs of such a modeling tool would be far too high. The goal of this study is to develop a general applicable neural network model that will predict the thermal load within the same accuracy range for any given building data. The model is used for function approximation, not classification. The function is to give the energy usage at a given point in time with certain weather and building occupancy conditions.

The neural network accuracy must be high enough to be commercially attractive. It is expected that the prediction accuracy, expressed in terms of coefficient of variation (CV), should not be more than 5.0%. For the neural network model to be generally applicable and cost-effective, it should have a predefined structure and other model parameters should be optimized automatically. It should be possible to use the neural network on a new building without going through a detailed neural network design phase. The calculation must be fast enough to evaluate the past 24 hours of data overnight. This paper consists of two sections.

- Part I defines the need and scope of this study and discusses neural networks as nonlinear modeling techniques as the basis for predicting building thermal load and the neural network training issues.
- Part II discusses selecting models, results, conclusions, and recommendations.

## DATA CHARACTERISTICS

As previously mentioned, the scope of this study focuses on how to find a general applicable model for the energy usage of buildings based on historical data. Since it will not be a physical model, the characteristics of the data are very important.

## MEASURED DATA

The data used in this study are obtained from several buildings with different BEMS. The data sets are stored in different ways in a database. This makes the data specific to the buildings as well as to the BEMS. Some data sets might include hourly time-series, whereas other data sets contain only daily values. Furthermore, one data set may include almost all weather variables (e.g., ambient temperature, solar radiation, wind speed, and ambient humidity), while another set contains only information about the maximum and minimum daily temperature. This diversity requires preprocessing to obtain a similarly relevant input set for the model. The model itself must be designed so that it can perform with any given input set without altering the model structure.

Another problem related to neural networks based on historical data is that not all relevant and necessary factors are included in the data set. For example, some buildings are operated mainly by human intervention, and the data set from such buildings might look inconsistent due to the variability of human operators. If there are several building operators, building operation is influenced by different people at different times. These factors cause extra noise on the building's energy consumption signal that is hard to compensate for or incorporate in the model.

A third problem is missing data points. Most likely, data points will be missing in every data set. This can be caused by a failing sensor or by failing data transmission. In any case, the system should be able to report this failure or make a good estimate with the remaining data.

In general, the following information can be relevant for the modeling:

1. Ambient temperature
2. Building occupancy (related to 5)
3. Date-stamp
4. Day (e.g., Monday), can be derived from the date-stamp
5. Day type (working or nonworking day), can be derived from 3 and 4
6. Energy usage (of any form)
7. Holiday season
8. Humidity
9. Indoor temperature
10. On or off status of equipment
11. Solar radiation
12. Time-stamp
13. Wind direction
14. Wind speed

Some data sets include many of these variables, while others are limited only to the date-stamp, daily ambient temperature, and the daily whole building energy consumption. The indoor temperature is very valuable for energy modeling since the energy usage depends largely on the difference of the indoor temperature and the set point of the HVAC system. Unfortunately, none of the data sets used in this study included indoor temperature.

To summarize, the system has to work with data sets that are far from ideal. As a result, the system should not be totally data driven. A pure black box model, totally dependent on the data, is not likely to be robust enough. Physical considerations must be included to make the system more robust.

## PHYSICAL CONSIDERATIONS

Even if nonlinear model structures are to be applied, there is no reason to estimate the known relationship between independent variables and the expected output. On the contrary, prior physical knowledge should be used to gain insight into the system. Therefore, a combination of black box modeling and physical considerations was chosen as a preferred method of modeling. This can be called "grey box modeling." Grey box modeling provides some physical insight, but the exact relationships remain unknown. For the models used in this project, physical insight is used where available.

## NEURAL NETWORK AS NONLINEAR MODELING TECHNIQUES

In this study, nonlinear modeling techniques are used for function approximation—the dependency of building energy usage on building occupancy and weather conditions. This section provides an overview of nonlinear modeling techniques considered for function approximation. All of these nonlinear models are either true neural network models or closely related to neural network models. The terms used in the following sections are commonly used in the neural network literature (Sjöberg et al. 1995).

In this study, designing neural network models can be separated with the following tasks:

1. Selecting input.
2. Selecting the activation function.[1]
3. Determining the number of hidden layers, the number of activation functions (or neurons) per layer,[2] and the interconnection of the function units.
4. Selecting the "learning" algorithm.

The input selection is discussed in detail in Part II of this paper while the other tasks are discussed below.

---

1. In the neural network literature, this is also referred to as *transfer function.*
2. The number of activation functions per layer is the same as the number of nodes, neurons, or units per layer in neural network literature.

## Selecting the Activation Function

The activation function is used to perform a mapping from the $\mathbf{R}^n \rightarrow \mathbf{R}$ with $n$ as the dimension. This mapping can be linear or nonlinear. All nonlinear model techniques used in this study are based on using multiple activation functions in a structured model. The structuring of the activation functions is discussed below.

### The Single-Variable Activation Function

The activation function $\kappa$ determines the (non) linear mapping of the argument $x$ to an output $y$. There are various activation functions that can be used. The four most commonly used activation functions for $x \in \mathbf{R}$ are as follows:

*Linear Activation Function*

$$y = \kappa(x) = \alpha \cdot x \tag{1}$$

where $\alpha \in R$.

*Unit Step Activation Function*

$$y = \kappa(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \tag{2}$$

*Sigmoid Activation Function (logarithmic)*

$$y = \kappa(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

*Radial Activation Function*

$$y = \kappa(x) = e^{-\frac{x^2}{2}} \tag{4}$$

Figure 2 shows these five activation functions. The sigmoid activation function is similar to the unit-step activation function, although the latter is a discrete function. The radial activation function can be seen as a smooth, continuous version of the unit interval function,

$$y = \kappa(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & else \end{cases} \tag{5}$$

Equation 5 is a variant of Equation 2 since this function can be obtained as the difference of two unit-step activation functions.

### Construction of Multi-Variable Activation Functions

The activation functions, illustrated in Figure 2, are single-variable activation functions. For multi-dimensional problems, multi-variable functions must be created. Often these functions are constructed from the single-variable activation functions. The following two methods can be used to construct multi-variable functions from single-variable activation functions.

*Ridge Construction*

For all $w \in \mathbf{R}^n$, $b \in \mathbf{R}$, $x \in \mathbf{R}^n$, and any single-variable activation function $k$, a ridge function has the form

$$y = \kappa(\bar{w}^T \cdot \bar{x} + b), \tag{6}$$

where $\bar{w}$ is a dilation or scaling parameter (weight) and $b$ is a position or transition parameter (bias).

Ridge constructions are normally used together with sigmoid, unit step, or linear activation functions.

*Radial Construction*

For any single-variable activation function $k$, a radial function with $b \in \mathbf{R}^n$ and $x \in \mathbf{R}^n$ is given by

$$y = \kappa \cdot (\|\bar{x} - \bar{b}\|_W) \tag{7}$$

where $\|\cdot\|_W$ is any chosen norm, typically a quadratic norm:

$$\|\bar{x}\|_W^2 = \bar{x}^T \cdot W \cdot \bar{x} \tag{8}$$

where $W$ is a possible scaling matrix. In simple cases, this is a diagonal matrix.

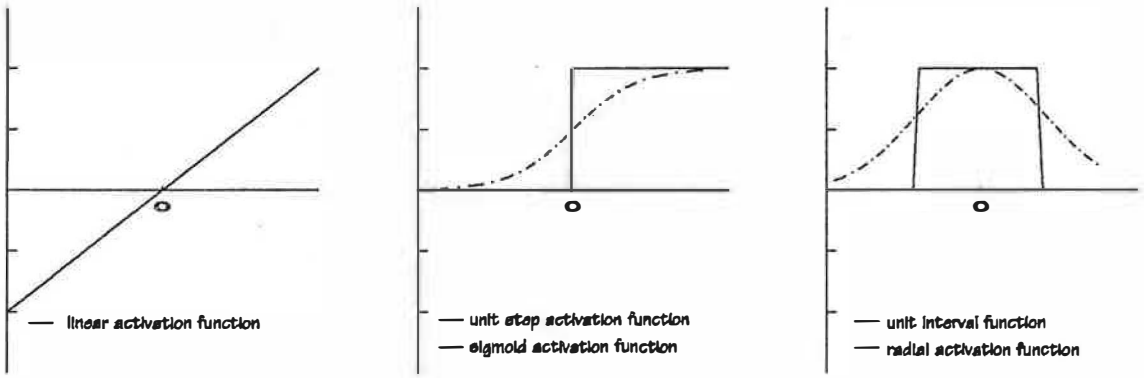


***Figure 2*** *Five activation functions. The last two graphs show the similarity between the continuous activation functions and the discrete activation functions (the unit interval function is shifted by -0.5 for comparison with the radial activation function).*

Figure 3 gives a graphical representation of a multi-variable activation function or neuron. The right-hand side of Figure 3 illustrates two different combinations of activation function and construction.

Radial constructions are mainly used with radial activation functions. The sigmoid activation function, used with radial construction, forms a bell-shaped function similar to the radial activation function. Figure 4 shows the output space of a radial activation function with radial construction and a logarithmic sigmoid activation function with ridge construction with a two-dimensional input space. The choice of the name ridge and radial construction is clear from the figure.

### Optimal Choice of Multiple-Variable Activation Functions

The choice of multiple-variable activation functions depends on the underlying distribution function of the data set. All of the described activation functions with the right model or network structures are capable of approximating any reasonable function with enough data (Hassoun 1995; Sjöberg et al. 1995; Veelenturf 1995). There are no general rules for choosing multiple-variable activation functions, but some observations can be made.

*Problem of Dimensionality.* An important consideration for the choice between activation functions obtained by radial and ridge constructions is the dimensionality of the data set. Since a data set is a finite set in practical cases, the dimension of the data set plays an important role. In a domain $\mathbf{R}^n$ with a moderately sized $n$, the data distribution is very sparse in any bounded region of the vector space. For example, if a minimum of $N$ data points is necessary to estimate a function $g$: $\mathbf{R} \rightarrow \mathbf{R}$, $N^n$ data points or vectors are necessary to obtain a similar vector density in the input space for an estimation with $n$ input variables ($g$: $\mathbf{R}^n \rightarrow \mathbf{R}$). In the case of (hourly) benchmarking, a data set generally consists of $10^3$ to $10^4$ vectors. For a density of 0.1 in an unit space, only three or four input dimensions are possible.

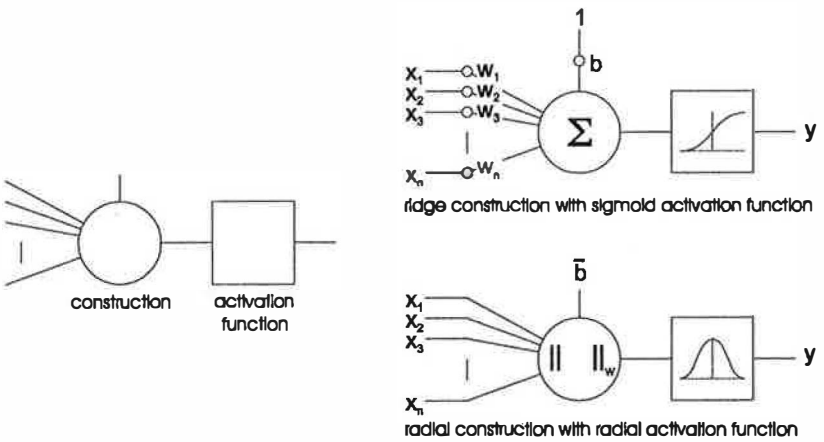


***Figure 3*** *Graphical representation of a logarithmic sigmoid activation function with ridge construction and a radial activation function with radial construction.*

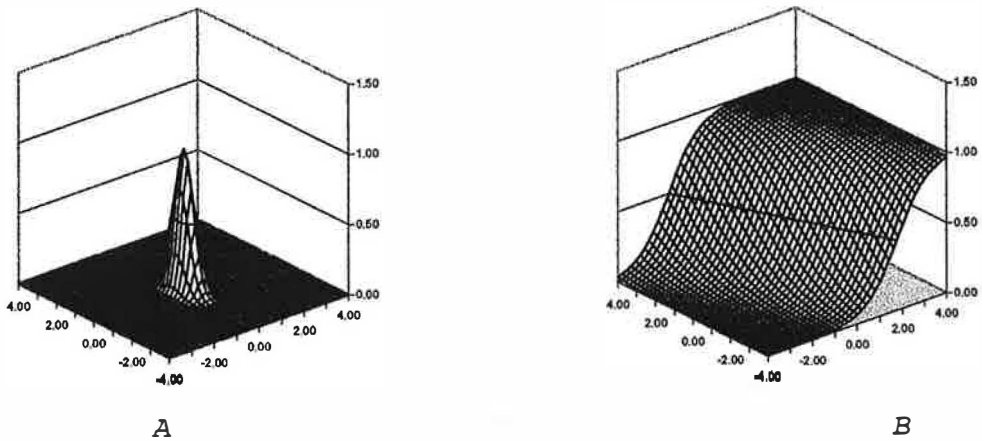


***Figure 4*** *A radial activation function with radial construction (A) and a logarithmic sigmoid activation function with ridge construction (B) with a two-dimensional input space.*

For radial constructions, the computational costs are mainly in the calculation of the norm $\|\vec{x} - \vec{b}\|$; for the ridge constructions, these costs depend on the inner-product $\vec{w}^T \cdot \vec{x}$. Therefore, we may expect these functions to perform well for higher dimensions. Unfortunately, models based on the radial constructions will not support any model statements outside the areas where observations are made. This reduces not only extrapolation possibilities but also interpolation possibilities in sparse areas of the vector-space. Since the data distribution for higher dimensions becomes very sparse, many areas of the input space are not covered. Ridge construction can handle higher dimensions better because they extrapolate into unsupported data regions, yet with unknown reliability. Ridge constructions offer a few directional selective features that can be extrapolated into these unsupported data regions. Whether this support is reasonable or not depends mainly on the application. The extrapolation (as well as the interpolation) scheme deteriorates with higher dimensions. Figure 4 shows the local support of the radial construction and the directional support of the ridge construction.

**Summary**

For small dimensions ($n \leq 6$), radial constructions (with radial activation functions) and ridge constructions (with sigmoid functions) should be examined. For larger dimensions, radial constructions might not support model statements outside the data regions; the ridge constructions will produce certain statements, but whether or not they are reasonable depends entirely on the application.

Furthermore, having as much information in a minimum number of input dimensions is preferable. As previously mentioned, both constructions are capable of approximating reasonable functions. However, sufficient data with respect to the number of input dimensions must be available.

## Determining the Number of Hidden Layers and Multi-Variable Activation Functions

The structure of many (non)linear models is multi-layered. Figure 5 shows two multi-layer structures. These structures are usually referred to as *neural networks*. Consider the two-layer structure of A, of which the first layer consists of a set of inputs. These inputs are connected with a layer built up from several multi-variable activation functions. The outputs of these activation functions are then (non)linearly combined to form the model's output. In this case, a single hidden-layer structure is created. The layer receiving the input signals is called the *hidden layer* because the outputs of this layer do not appear explicitly in the output. It is also possible to consider the outputs of the hidden layer again as inputs to a second hidden layer, as shown in B. This allows for obtaining a structure with several hidden layers. The number of layers and number of activation functions per layer to be used are not well defined.

**Number of Hidden Layers**

The question of how many hidden layers should be used cannot be answered by existing theory. Previous publications (Cybenko 1989; Hornik 1989; Funahasi 1989; Veelenturf 1995) have proved that a one-hidden-layer feed-forward neural network is capable of uniform approximation of continuous multi-variant function to any desired degree of accuracy. An overview on theorems related to this subject is available in a previous research publication (Hassoun 1995). However, Veelenturf (1995) has shown that in some cases, it might be profitable to use more than one hidden layer. This is especially true when the function to be estimated is expected to contain discontinuities. Sontag (1993) also provides insights to the importance of a second hidden layer in a nonlinear structure. Since there are no straightforward answers to this design variable, it is best to begin by using the simplest, i.e., to use a single hidden-layer structure.

**Number of Multi-Variable Activation Functions**

There are no general rules for determining how many activation functions or number of function units in the hidden layer or the number of hidden layers are needed. It is often recommended to use trial and error to get the best result. However, there are two things that can be noted concerning the number of function units in the hidden layer.

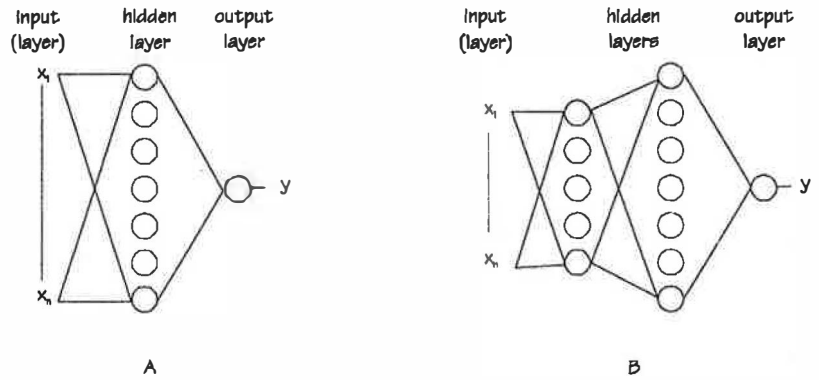


***Figure 5*** *Multi-layer structures of activation functions.*

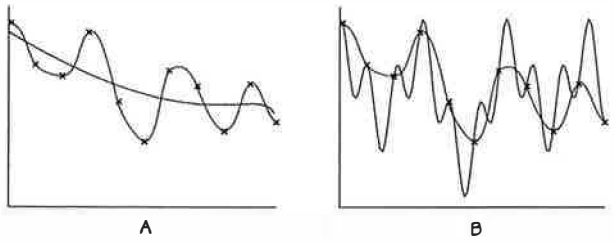


***Figure 6*** *Illustration A shows underfitting. Clearly, the estimations fail to go through the data points. Illustration B shows overfitting, where the data used to train the model are estimated well, but the generalization is poor.*

If the number of function units is too small, the model does not have enough free parameters to approximate the underlying function well. This is often referred to as *underfitting*. On the other hand, if the number of function units is too large, the model has too many free parameters and may lose its generalizing (interpolation) capabilities. The data used to train the model will be well "estimated," but statements about new data points will be useless. This is called *overtraining* or *overfitting*. The underfitting and overfitting functions are illustrated in Figure 6.

Thus, if long-term training still results in large errors (in the training set), the problem most likely lies in a lack of hidden function units. It may happen, however, that one fails to incorporate an important input variable in the data set, resulting in a high noise level. In the case of overfitting, there are probably too many hidden function units. It is possible to reduce overfitting errors without changing the model's structure by stopping training before the minimum error on the training set is reached. The optimum training time can be obtained by using cross-validation (Hassoun 1995). With cross-validation, an extra set (other than the training set) is used to test the estimation result. The error of this set decreases monotonically to a minimum, after which the error starts to increase even as the training set error decreases.

#### Interconnection of the Function Units

The interconnection of the function units determines if the model's structure is dynamic or static. Every function unit (or neuron) is connected to all function units of the previous and following level. This type of network is called a *feed-forward network*, since the input signals are fed forward. If static structures are used to model a physical system that is dynamic, extra pre-processing must be performed to incorporate the dynamics of the physical system into the model.

The recurrent network forms another group of neural network paradigms. Here, certain function unit outputs are connected to the inputs of other function units in the previous or the same layer. Recurrent networks obviously are dynamic models. There are many examples of recurrent networks in the area of function approximation. Recurrent networks are, however, understood poorly and can have instabilities due to their structure. This report only covers static model structures.

## Common Nonlinear Models and Neural Network Structures

This section gives a brief overview of the neural network that is considered for function approximation. For basic information on neural network structures, the publications by Hassoun (1995), Hertz et al. (1991), Lippmann (1987), and Veelenturf (1995) are excellent source of references.

1. *Multi-layer perceptron (MLP).* An MLP has one or more hidden layers with ridge constructions. The activation function is usually a logarithmic or tan*h* sigmoid function. Figure 5 shows two different kinds of multi-layer structures. In the neural network literature, this is often referred to as an MLP or feed-forward network. The MLP architecture is most popular in practical applications and has been used for function approximation in many cases (Coda 1994). Publications by Hassoun (1995), Hertz et al. (1991), Lippmann (1987), and Veelenturf (1995) can be consulted for detailed information on MLP.

2. *Radial basis function (RBF) networks, conditional mean, and general regression neural network (GRNN)*

   *RBF networks.* It is clear from the name that radial activation functions (Equation 4) are used with radial construction (Equation 7). There are two types of RBF structures. The first type uses the radial activation function directly without normalization of the outputs of the units. The output of such an RBF network is a number of superimposed activation functions. Consequently, this output can be quite uneven. For such a network to be capable of fitting even the simplest functions, many neurons (or units) are required.

   Since radial activation functions can have these problems with interpolation and extrapolation, caused by the local activation of the radial activation functions, the second type of RBF structure is mainly used. In the case of normalized RBF networks, the outputs of all the hidden units are normalized and summed to result in one. The normalized RBF networks were introduced in Moody and Darken (1989) and can be written as (with the Gaussian kernel, the radial activation functions with radial construction are often referred to as *kernels*):

$$\hat{Y}(\bar{x}) = \frac{\sum_{i=1}^{n} Y_i \cdot \varpi_i \cdot \exp\left(-\frac{(\bar{x}-\bar{x}_i)^T \cdot (\bar{x}-\bar{x}_i)}{2h_i^2}\right)}{\sum_{i=1}^{n} \varpi_i \cdot \exp\left(-\frac{(\bar{x}-\bar{x}_i)^T \cdot (\bar{x}-\bar{x}_i)}{2h_i^2}\right)}, \qquad (9)$$

where

$\bar{x}$ = test input vector,

$\bar{x}_i$ = centroid of the $i$th unit,

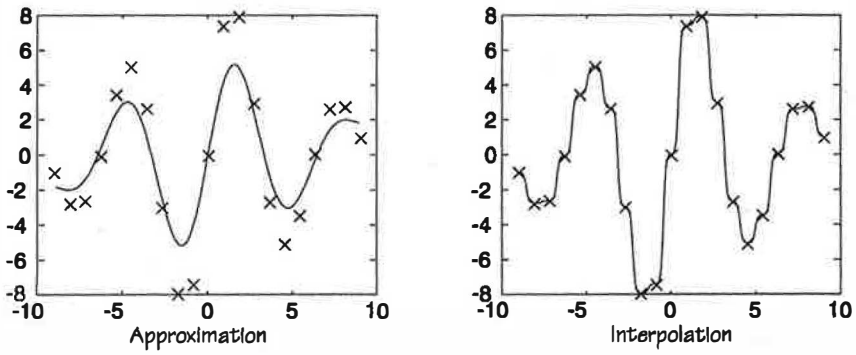


*Figure 7* *Output of the conditional mean with the Gaussian kernel function as an approximator (h = 1) and as an interpolator (h = 0.25) is shown above on the LHS and RHS, respectively.*

$h_i$ = smoothing parameter (width) of the $i$th kernel,

$\varpi_i$ = weight (height) of the $i$th kernel,

$Y_i$ = output corresponding to $\bar{x}_i$,

$\hat{Y}(\bar{x})$ = estimation output of $\bar{x}$.

There are several subtypes of normalized RBF networks. These networks differ by their optimized parameters. One can choose to have the width or the height of each kernel optimized separately or both. The simplest version is to have all heights (weights $\omega_i$) set to one and to have one global width (smoothing parameter) for all kernels instead of an individual smoothing parameter per kernel.

*Conditional mean.* If each training input vector is taken as an RBF centroid for the simplest network, the outputs are simply weighted averages of the target values and form a conditional mean. The conditional mean is, in fact, the kernel interpolation function introduced by Nadaraya (1964).

$$\hat{Y}(\bar{x}) = \frac{\sum_{i=1}^{n} Y_i \cdot K\left(\frac{(\bar{x}-\bar{x}_i)}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{(\bar{x}-\bar{x}_i)}{h}\right)} \qquad (10)$$

$\bar{x}$ = test input vector

$\bar{x}_i$ = training vector $i$

$h$ = smoothing parameter

$K$ = kernel[3]

$Y_i$ = output corresponding to $\bar{x}_i$

$\hat{Y}(\bar{x})$ = estimation output of $\bar{x}$

Using the Gaussian kernel, the following conditional mean (also called regression) for discrete data is formed:

$$\hat{Y}(\bar{x}) = \frac{\sum_{i=1}^{n} Y_i \cdot \exp\left(-\frac{(\bar{x}-\bar{x}_i)^T \cdot (\bar{x}-\bar{x}_i)}{2h^2}\right)}{\sum_{i=1}^{n} \exp\left(-\frac{(\bar{x}-\bar{x}_i)^T \cdot (\bar{x}-\bar{x}_i)}{2h^2}\right)}. \qquad (11)$$

The conditional mean is a weighted average, with the weighting depending on the kernel shape and Euclidean distance.

The smoothing parameter (or kernel width) $h$ determines if the conditional mean is an interpolator. When $h$ is large, the conditional mean is an approximator (for $h \to \infty \Rightarrow$ conditional mean $\to$ [unconditional] mean); whereas if $h$ is small, $\hat{Y}(\bar{x})$ tends to go exactly through the observed $Y_i$ and interpolates in between.[4] Figure 7 shows the conditional mean with Gaussian kernel as an approximator and as an interpolator.

Note the resemblance to Figure 6 in the case of under- and overfitting. For the conditional mean, the smoothing parameter $h$ influences the accuracy of the function estimation, instead of the number of activation functions or neurons for MLPs.

*General regression neural network (GRNN).* GRNN was introduced in 1991 by Specht (1991) and is equivalent to the Nadaraya kernel regression estimator or conditional mean. The GRNN includes a training method to determine an optimal value for smoothing parameter $h$. In his paper, Specht presents a method for finding both the optimum smoothing parameter and an adaptive algorithm

3. In the literature, a kernel transfer function is often designated with the symbol $K$ instead of $\kappa$.

4. As the smoothing parameter approaches zero, the weights approach one for the nearest neighbor and zero for all other vectors. Thus, making the conditional mean a nearest neighbor classifier of the test vectors with respect to the training vectors.

for the conditional mean. The adaptive algorithm creates an $h$ that depends on the data distribution in the area. With the conditional mean, the smoothing parameter is constant over the entire data set. (For more information see Specht 1991.) For more information on RBF networks, consult the works of Chen et al. (1991), Hassoun (1995), and Moody and Darken (1989).

## The Learning Algorithm

The selection of a learning algorithm depends on the type of model and the selected performance criterion. This section lists the preferred algorithm for the MLP model and the RBF networks.

1. For the MLP model, back-propagation (BP) algorithms are commonly used. The basic BP algorithm is the gradient descent rule. Gradient descent simply stands for a technique in which parameters, such as weights and biases, are moved in the opposite direction of the error gradient in order to minimize the estimation error (e.g., the root mean square error). There are several enhanced BP algorithms and variations of BP. In this study, BP with Levenberg-Marquardt (LM) optimization (Beale and Demuth 1994) is used for MLP networks. The LM optimization is a hybrid method based on the Gauss-Newton method (Atkinson 1989) and the gradient decent rule. Whereas the Gauss-Newton method improves the speed of convergence of the parameters to the "nearest" (local) minimum, the gradient descent rule with a fixed learning rate allows the LM optimization to escape from a shallow local minimum. A scalar determines the influence of the Gauss-Newton and the gradient descent rule. If the estimation error is decreasing, the scalar is updated to give the Gauss-Newton method more influence. If the estimation error is increasing, the scalar is updated in the opposite direction to give more weight to the gradient descent term of the hybrid function. For further information on BP-based algorithms, consult the work of Hassoun (1995) and Veelenturf (1995).
2. An algorithm for normalized RBF networks needs to update one or multiple variables and, therefore, depends on which network architecture is used. In this study, only the simple normalized RBF architecture, namely, conditional mean or GRNN, is used. For more on different types of RBF algorithms, consult literature published by Chen et al. (199), Hassoun (1995), Moody and Darken (1989), Preuβ and Tresp (1994), and Preuβ (1994). For this model, the smoothing parameter $h$ is often calculated with the holdout method. For a particular value of $h$, the holdout method stands for removal of one training vector at a time and output estimation for that vector with the use of remaining training vectors. This process is repeated for every sample, and an error measure (e.g., the mean square error [MSE]) between the actual outputs and the estimations is calculated. The smoothing parameter $h$ that minimizes this error is used. If the training set is large, the computation of the smoothing parameter can be very time consuming. For general information on learning algorithms, Hassoun (1995), Hertz et al. (1991), Sjöberg et al. (1995), and Veelenturf (1995) are excellent sources.

### Global vs. Local Training

In the previous section on the GRNN learning algorithm, it is mentioned that the smoothing parameter computation for this model is time consuming. Training effort can be reduced by using only a subset of all the training data to calculate this parameter. If this subset only covers a certain area of the total vector space, the smoothing parameter is trained locally. This is referred to as *local training*. On the other hand, if the whole training data set is used or the subset also covers the whole vector space (e.g., with vector quantization), the smoothing parameter is trained globally.

This section discusses these two methods of training, global and local training.[5]

### Global Training

With global training, the whole training set is used or a representative subset is selected from the entire set. This training subset might be chosen randomly (e.g., vector quantization). The subset should incorporate the data points that bound the data set (e.g., maximum and minimum temperatures over a year). This way extrapolation is not performed on areas that could actually be supported by data.

When a network is trained with such a training set, a globally (or generally) fitted model is created. This model is valid for the whole data set. It can be used directly on any data set with the same distribution (e.g., in the case of benchmarking, on the same building). A disadvantage of this method is that in order to generalize over the whole data set, some specific variances or local patterns in a certain region of the data set are lost. Adaptation of the network to fit a local pattern may reduce accuracy in other regions of the input space. This can be solved by using activation functions with only local receptive fields (radial construction) instead of directional activation functions (ridge construction).

### Local Training

In contrast to global training, function approximation can also be optimized locally. The function estimation in a small bounded area of the input space is then based solely on training vectors found in the local area of the input space.

Figure 8 shows a local estimation of a function with one dimension (time) only. The signal repeats itself every 24 hours. For the estimation, only one sigmoid activation function is used. Figure 8 illustrates that the function estimate is valid only for the three-hour range indicated by the two vertical lines and for the same time frame on all other days. Outside of this range, the model does not make a correct estimation.

5. Note that global or local training is independent of the learning algorithm but has to do with the location of the training vectors used with respect to the whole vector space. All learning algorithms can be used for both global training and local training.

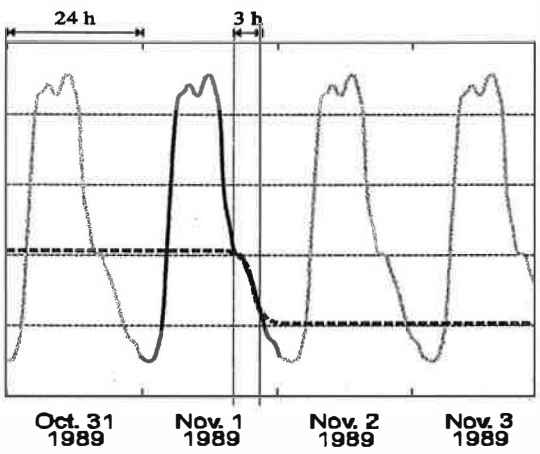


***Figure 8*** *Local estimation of a one-dimensional function.*

Local fitting can be compared to linearization. The function estimation is valid locally within a certain region. Outside of that region, the estimation loses its validity.

There are three advantages of local training.

- The model can be less complex, since the limited region of the input space can be mapped using less free parameters.
- Computation time for training is reduced, since fewer training vectors are used and also because the model can have fewer free parameters.
- Another advantage is that the model might better adapt to region-specific patterns.

Using a limited number of training vectors, however, can be a disadvantage. If the training set has noise and also includes a small percentage of outliers (e.g., faulty data), these outliers could influence the performance since the percentage of outliers might be high for a selected subset.

Local training can be performed by defining classes (e.g., working day and nonworking day or daytime and nighttime). Depending on the class to which a test vector belongs, a specific training set is used to train a network. Defining of classes can be performed by a clustering algorithm. Four clustering methods are discussed in the "Clustering" section of this paper.

Another possibility of local training is to separately select the training vectors for each case and test vector based on a distance criterion relative to the test vector. This method is referred to as *case-based reasoning* (CBR) and is discussed later.

*Clustering.* Cluster methods that can be considered are

- knowledge-based clustering
- K-mean clustering
- Kohonen's self-organizing feature map (SOFM) algorithm
- self-creating and organizing neural networks (SCONNs)

*Knowledge-based clustering.* Knowledge-based clustering is possible in advance due to prior system knowledge. Thus, different clusters based on time of the year and building occupancy can be made. For example, it may be well known that the building systems respond differently when the building is unoccupied, as many systems are shut down during these hours. This kind of preselection is valuable where a neural network can be constructed with working day and nonworking day classes. The model can be a hybrid form between local and global training. A global model was created but built with two locally trained neural networks, the outputs of which were combined in a globally trained combination network (Breekweg 1995). The outputs of locally trained neural networks can be combined to create a global network and subsequently trained globally.

*K-mean clustering.* This clustering algorithm assumes a fixed number of clusters $k$ and operates by associating a data point $\bar{x}_i$ to the cluster with the centroid $\bar{w}_j$ closest (e.g., Euclidean distance) to $\bar{x}_i$ and then updates the centroids for the revised clusters. The K-mean algorithm converges to a locally optimal cluster configuration, which is influenced through initial assignment of the $k$ centroids. It is a popular clustering method due to its simplicity.

*Kohonen's self-organizing feature map (SOFM).* The self-organizing feature map (SOFM) was first introduced by Teuvo Kohonen in 1982 (Kohonen 1989; Hassoun 1995; Veelenturf 1995). The SOFM is a neural network that organizes itself only as a function of its inputs. This neural network is often used for clustering or vector quantization (VQ).

The SOFM can group data into a number of categories or clusters. The algorithm, similar to the K-mean algorithm, assumes a predefined fixed number of clusters. The centroids of the clusters are the weights of the network's neurons. The neurons are ordered (in a lattice) in one or multiple dimensions. After training, neighboring neurons correspond to neighboring clusters in the input space. The mapping of the input space is such that the clusters have a similar topology as the original input space. This way, the centroids of the clusters form a VQ of the original input space.

This clustering method has received much attention over the last decade, particularly because of its topology or feature mapping. The SOFM cluster configuration is not influenced by the initial assignment of the centroids, as is true for the K-mean algorithm. Instead, the SOFM cluster configuration is influenced by the order in which the data are presented to the algorithm.

*Self-creating and organizing neural networks (SCONNs).* Self-creating and organizing neural networks (SCONNs) create an adaptive vector quantization (VQ). There are two

different SCONNs: the first one creates an adaptive uniform VQ, and the second one creates an adaptive nonuniform VQ. A SCONN begins with only one neuron and sufficiently broad activation level. The activation level decreases depending on the time or the activation history. With every new input vector, a decision is made whether to adapt the weights of the existing neurons or to create a new neuron. The SCONN differs from K-mean clustering and SOFMs mainly by finding the optimum number of clusters. No fixed number of clusters is assumed. Furthermore, the neurons of the SCONN are not arranged in a lattice but in a tree structure. Additionally, topology or feature mapping capabilities can be incorporated in SCONNs. Choi and Park (1994) have published a comparison between SOFM and SCONNs.

*Case-based reasoning (CBR).* The difference between clustering and case-based reasoning (CBR) is that with CBR, a new training subset is selected from the input space for each new test case. With clustering, however, a lot of computational time is invested in clustering the entire input space. After clustering is completed, two different test vectors that both fall in the domain of attraction of the same cluster are trained with the same subset. All other training vectors that do not belong to the same subsets are not considered for training. Selecting a given number of training vectors with best resemblance to each new test vector, e.g., Euclidean distance, is much faster, as is the case with CBR.

There are four different ways to select training vectors from the total training set with CBR.

- knowledge-based decision
- fixed radius
- fixed number of vectors
- adaptive radius with a preset minimum number of training vectors

*Knowledge-based decisions.* Knowledge-based preselection is possible regarding the test vector. For example, select all data points that have less than a given number of hour or day differences to the test vector. In other words, choose the data points that have approximately the same time of day or the same time of year.

*Fixed radius.* All training vectors that are closer to the given test vector within a fixed distance $\alpha$ are selected for training. The distance can be an Euclidean distance. With a fixed radius, however, the number of the training vectors will vary for different test vectors. If $\alpha$ is chosen too small, the number of training vectors selected in a sparse area of the input space may not be sufficient for an estimation.

*Fixed number of vectors.* The closest $N$ training vectors are selected. The number of vectors to be selected remains an open issue. The other problem that arises is that vectors $\bar{v}_{N-m}$ to $\bar{v}_N$ might have a very large distance to the test vector compared to $\bar{v}_1$ to $\bar{v}_{N-m-1}$, thereby compromising the possible adaptation to local patterns of the function.

*Adaptive radius with a preset minimum number of training vectors.* The number of training vectors that are relevant for a good prediction of the output for a given test vector should depend on the density of training vectors in the area around the test vector. A minimum number of training vectors should be selected for proper training of a nonlinear model. Extra training vectors are selected based on the underlying probability density. If a test vector is located in a dense area of the vector space, many resembling training vectors are available. The radius should then be small in order to keep the training set small. However, for a sparse area of the vector space, the radius should be chosen larger to obtain the minimum number of training vectors.

This last method presumes prior knowledge of the underlying distribution function of the vector space. Since this knowledge is not available, it is necessary to get an indication of the probability density function of this space. An indication of the probability density function can be made by means of a histogram. However, this method has the disadvantage that it can only compute discrete density estimation. Furthermore, the influence of the dimensionality of the vector space is considerable.

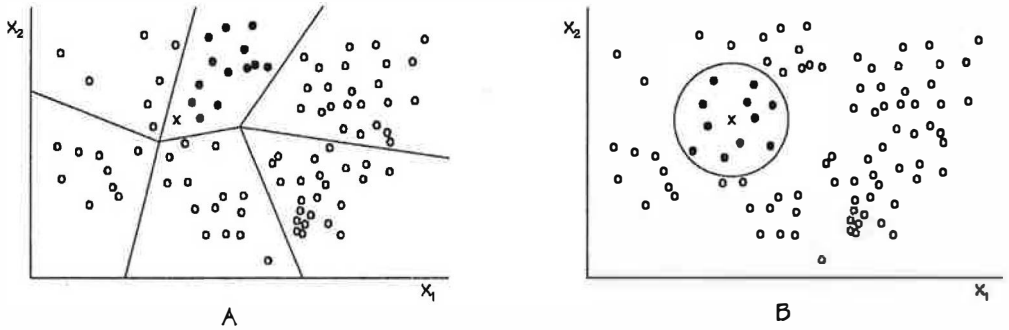


***Figure 9*** *Illustration A shows the selection of the training vectors using clustering. Illustration B gives an impression of case based selection of training input vectors resembling a test input vector. In B, information on how the other vectors are clustered is not of interest.*

A method for a continuous estimation of the density function, which is less influenced by the dimensionality, can be implemented with kernel estimators (radial activation functions with radial construction). The kernel estimator was introduced by Parzen (1962) and is known as the Parzen density estimation. In the case of the Parzen density estimation, a radius can be calculated to obtain an indication of the underlying distribution function. The radius ρ for the selection of the training set should be related to the probability estimate $\hat{f}(\bar{x})$ according to the following formula:

$$\rho(\hat{f}) \cong \frac{1}{\hat{f}(\bar{x})}. \tag{12}$$

A high probability estimate indicates that there are several resembling training vectors. The radius chosen should be small since one wants a limited set of training vectors that closely resembles the test vector. On the other hand, a low probability estimate indicates that the number of resembling training vectors is small. Consequently, a larger radius is necessary to find sufficient training vectors. It may be wise to set a fixed minimum number of training vectors that are necessary to train the neural network. A combination between knowledge-based selection and one of the other three options is also possible.

## CONCLUSION

Part I of this paper essentially develops a framework for neural network-based methods for detecting faults in building energy performance. Several neural network design factors that are key to success are discussed, including data characteristics, physical considerations, activation function, number of hidden layers, interconnection between activation functions, and the learning algorithms. The learning algorithm is investigated in more detail, including local vs. global training, case-based reasoning, and clustering. Additionally, several common neural network structures are also presented. The discussions in Part I set the stage for selecting a good prediction model and then applying it to real data. Both the model selection and results are covered in Part II of this paper.

## REFERENCES


Atkinson, K.E. 1989. *An introduction to numerical analysis*. 2d ed. Singapore: John Wiley & Sons.

Beale, M., and H. Demuth. 1994. *Neural network toolbox for use with MatLab*. Version 2.0a. Natwick, Mass.: The MathWorks, Inc., January.

Breekweg, M.R.B. 1995. Energy management in commercial buildings using neural networks. Internship report. University of Twente, Enschede, the Netherlands.

Chen, S., C.F.N. Cowan, and P.M. Grant. 1991. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, March 2 (2): 302-309.

Choi, D.-I., and S.-H. Park. 1994. Self-creating and organizing neural networks. *IEEE Transactions on Neural Networks 5 (4):* 561-575.

Coda, F.M. 1994. Predicting hourly building energy use: The great energy predictor shootout. A collection of papers from the ASHRAE meeting at Orlando, Florida, June 1994. *ASHRAE Technical Data Bulletin*, vol. 10, no. 5.

Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function. *Mathematical Control Signals Systems*, vol. 2, pp. 303-314.

Ferrano, F.J., and K.V. Wong. 1990. Prediction of thermal storage loads using a neural network. *ASHRAE Transactions* 96 (2): 723-726.

Funahasi, K.-I. 1989. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, March 2 (3): 183-192.

Gibson, G.L., and T.T. Kraft. 1993. Electric demand prediction using artificial neural network technology. *ASHRAE Journal* 35 (3): 60-68.

Hassoun, M.H. 1995. *Fundamentals of artificial neural networks*. Cambridge, Mass.: The MIT Press.

Hertz, J.A., A. Krogh, and R.G. Palmer. 1991. *Introduction to the theory of neural computation*. Reading, Mass.: Addison-Wesley.

Hornik, K., M. Stinchcombe, and H. White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks*, May 2 (5): 359-366.

Haberl, J.S., and S. Thamilseran. 1996. The great energy predictor shootout II: Measuring Retrofit Savings—Overview and discussion of results. *ASHRAE Transactions* 102 (2): 419-435.

IEA. 1982. *World Energy Outlook*. Paris: International Energy Agency and Organisation for Economic Cooperation and Development.

Kawashima, M., C.E. Dorgan, and J.W. Mitchell. 1995. Hourly thermal load prediction for the next 24 hours by ARIMA, EWMA, LR and an artificial neural network. *ASHRAE Transactions* 101 (1): 186-200.

Kreider, J.F., and J.S. Haberl. 1994. Predicting hourly building energy usage. *ASHRAE Journal* 36 (6): 72-81.

Kohonen, T. 1989. *Self-organization and associative memory*. 3d ed. Berlin: Springer Verlag.

Levermore, G.J. 1992. *Building energy management systems: An application to heating control*. London: E & FN Spon.

Lippmann, R.P. 1987. An introduction to computing with neural nets. *IEEE ASSP Magazine*, April, pp. 4-22.

Moody, J., and C. Darken. 1989. Fast learning in networks of locally-tuned processing units. *NeuralComputation* 1(2): 281-294.

Nadaraya, E. 1964. *On estimating regression*. In: *Theory of probability and applications*, chapter 9, pp. 141-142.

Parzen, E. 1962. On estimation of a probability density function and mode. *Annal of Mathematical Statistics.*

Preuß, H.P., and V. Tresp. 1994. Neuro-Fuzzy (in German). *I Automatisierungstechnische Praxis*, May 36 (5): 10-24.

Preuß, H.P. 1994. Methoden der nichtlinearen Modellierung —vom Interpolationspolynom zum Neuronalen Netz (in German). *Automatisierungstechnik*, October 42 (10): 449-457.

Sjöberg, J., Q. Zhang, L. Ljung, A. Benveniste, B. Deylon, P.-Y. Glorennec, H. Hjalmarsson, and A. Juditsky. 1995. Nonlinear black-box modeling in system identification: A unified overview. *Automatica*, June.

Sontag, E. 1993. Neural networks for control. In: *Essays on Control: Perspectives in the Theory and its Applications.* H. Trentelman and J. Willems, Progress in Systems and Control Theory, vol. 14: 339-380.

Specht, D.F. 1991. A general regression neural network. *IEEE Transactions on Neural Networks*, November, 2 (6): 568-576.

Veelenturf, L.P.J. 1995. *Analysis and applications of artificial neural networks.* United Kingdom: Prentice Hall.