September 1993
LU TP 93-24

Predicting System Loads with Artificial Neural Networks – Methods and Results from "the Great Energy Predictor Shootout"

Mattias Ohlsson[1], Carsten Peterson[2],
Hong Pi[3], Thorsteinn Rögnvaldsson[4] and Bo Söderberg[5]

To appear in the **1994 Annual Proceedings of the American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc.**

Abstract:

We devise a feed-forward Artificial Neural Network (ANN) procedure for predicting utility loads and present the resulting predictions for two test problems given by "The Great Energy Predictor Shootout - The First Building Data Analysis and Prediction Competition" [1]. Key ingredients in our approach are a method (δ -test) for determining relevant inputs and the Multilayer Perceptron. These methods are briefly reviewed together with comments on alternative schemes like fitting to polynomials and the use of recurrent networks.

[1] mattias@thep.lu.se
[2] carsten@thep.lu.se
[3] pihong@thep.lu.se
[4] denni@thep.lu.se
[5] bs@thep.lu.se

1 Introduction

This paper concerns methods and results in making predictions for two test problems (A and B) given by "The Great Energy Predictor Shootout - The First Building Data Analysis and Prediction Competition" [1]. The present approach scored # 1 on set B and # 2 on set A. Both problems were given to the contestants in terms of tables of historic data (dependent and independent variables). In approaching the two data sets a few different strategies were explored. The most powerful approach for these applications turned out to be based on the following two key ingredients:

- Prior determination of dependencies in the data by using the δ-test [2].
- Feedforward Artificial Neural Networks (**ANN**) for model building (system identification).

This approach is of "black box" nature. However, it is stressed that prior "expert" knowledge about holidays etc. is essential for good performance.

This paper is organized as follows. In sections 2 and 3 we very briefly review the δ-test approach and ANN method, respectively, with slightly more emphasis on the former, given that it may not be as well known in the community as ANN. The results for data set A and B are discussed in sections 4 and 5. Section 6 contains a brief summary.

2 The δ-test

The behavior of a system is often modeled by analyzing a time series record of certain system variables. Using ANN to model such systems recently has attracted much attention. The success of such models relies heavily upon identifying the underlying structure in the time series – it is advantageous to know in advance the embedding dimension, which inputs are most relevant, noise level, etc. Existing methods for doing this are based either on linear regression, which limits the analysis to linear dependencies, or on trial-and-error procedures. The δ-test [2], to be briefly described below, aims at determining any dependency, be it linear or nonlinear, assuming an underlying continuous function.

Assume that we have sequences of measurements on a *dependent* variable z_0 and a set of *independent* variables z_1, z_2,...z_m. These measurements can correspond to multivariate time series, or to a univariate time series, in which case z_k should be understood as a time lagged variable of z_0: $z_k(t) = z_0(t-k)$. The question is whether there exist functional dependencies of the form

$$z_0 = f(z_1, z_2, ..., z_m) + r \qquad (1)$$

where r represents an indeterminable part, which originates either from insufficient dimensionality of the measurements or from real noise. If the system is completely deterministic in terms of the set of independent variables one has $r = 0$. In the case of a univariate time series, the dimension $d_{min} = m + 1$, which is sufficient to minimize r, is called the minimum *embedding dimension*.

We approach the problem by constructing conditional probabilities in embedding spaces of various dimensions d. The time series can be represented as a series of N points $\mathbf{z}(i)$ in a $(d+1)$-dimensional space $(d = 0, 1, 2, ...)$

$$\mathbf{z}(i) = (z_0(i), z_1(i), .., z_k(i), ., z_d(i)). \tag{2}$$

In terms of distances $l_k(i,j)$ between the k:th components of two vectors $\mathbf{z}(i)$ and $\mathbf{z}(j)$

$$l_k(i,j) = |z_k(i) - z_k(j)|, \quad k = 0, 1, ..., d \tag{3}$$

one can construct the conditional probabilities

$$P_d(\epsilon|\,\vec{\delta}) \equiv P(l_0 \leq \epsilon|\,\vec{l} \leq \vec{\delta}) = \frac{n(l_0 \leq \epsilon, \vec{l} \leq \vec{\delta})}{n(\vec{l} \leq \vec{\delta})}, \tag{4}$$

where ϵ and δ are positive numbers and $n(\vec{l} \leq \vec{\delta})$ and $n(l_0 \leq \epsilon, \vec{l} \leq \vec{\delta})$ are the number of vector pairs satisfying the corresponding distance constraints[6]. How does the probability structure of the dependent variable behave with respect to the independent variables? In ref. [2] the following important observations were made:

1. For a completely random time series there is no dependency and one has

$$P_0(\epsilon) = P_1(\epsilon|\,\vec{\delta}) = ... = P_d(\epsilon|\,\vec{\delta}) = ... \tag{5}$$

   This identity, which should be understood in a statistical sense, holds for any choice of positive ϵ and δ.

2. If a continuous map exists as in eq. (1) with no intrinsic noise, then for any $\epsilon > 0$ there exists a δ_ϵ such that

$$P_d(\epsilon|\,\vec{\delta}) = 1 \quad \text{for} \quad \delta \leq \delta_\epsilon \quad \text{and} \quad d \geq d_0 \tag{6}$$

   where d_0 represents some minimum dimension which covers all the relevant variables. This is a direct consequence of the definition of function uniform continuity,

3. In the presence of noise r, $P_d(\epsilon|\,\vec{\delta})$ will no longer saturate to 1 as ϵ becomes smaller than the width Δr_{max} of the noise.

The behavior of $P_d(\epsilon|\,\vec{\delta})$ as a function of δ and ϵ for various d are shown schematically in fig. 1. The consequences of randomness (eq. (5)) and complete determination (eq. (6)) provides a yardstick with which to measure the degree of dependency between the variables. Interesting quantities to examine are the maxima

$$P_d(\epsilon) \equiv \max_{\delta > 0} P_d(\epsilon|\,\vec{\delta}) = P_d(\epsilon|\,\vec{\delta})|_{\delta \leq \delta_\epsilon}. \tag{7}$$

[6] The notation $\vec{l} \leq \vec{\delta}$ is short for $\{(l_1 \leq \delta), (l_2 \leq \delta), ..., (l_d \leq \delta)\}$.

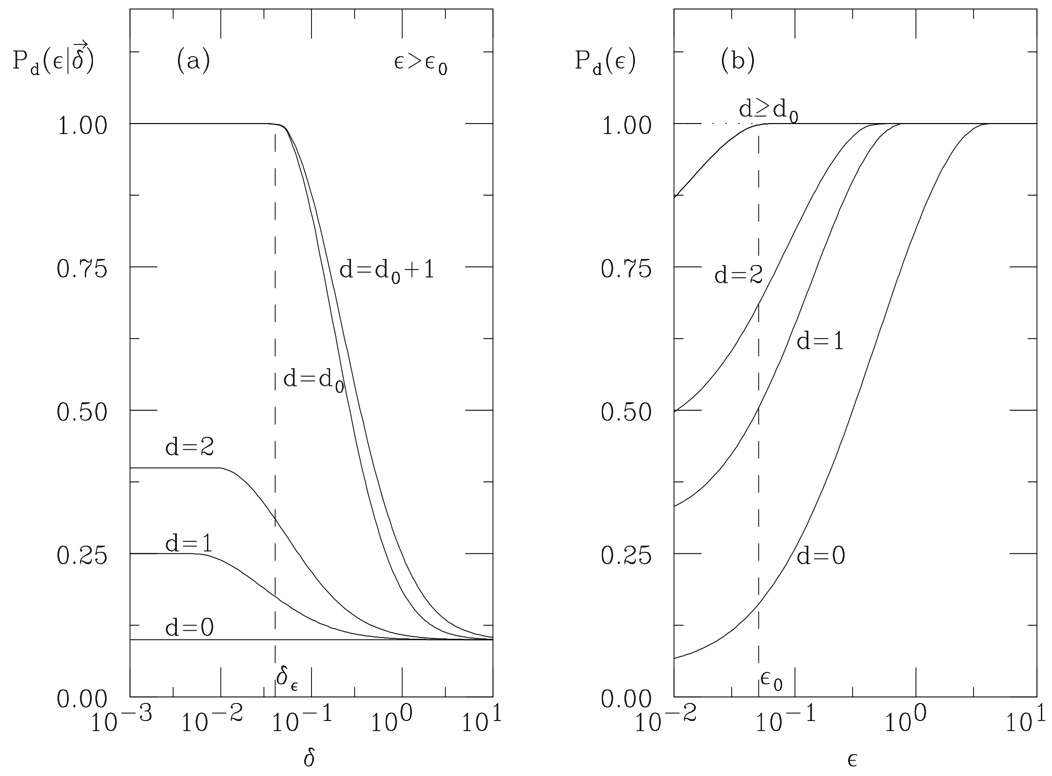

Figure 1: (a). $P_d(\epsilon|\vec{\delta})$ as a function of δ for fixed ϵ. Saturation to 1 would be observed if the width of noise is less than ϵ. (b). Behavior of the maxima $P_d(\epsilon) \equiv \max_{\delta>0} P_d(\epsilon|\vec{\delta})$ as a function of ϵ. The region saturating to 1 would be pushed toward smaller ϵ if the d:th conditional variable is relevant. The point ϵ_0 at which the saturation deviates from 1 can be identified approximately as the width of the noise $\Delta r_{max} \sim \epsilon_0$.

How $P_d(\epsilon)$ changes with d and ϵ provides basically all the information we need (see fig. 1b). To quantify the dependency on each of the variables, it is convenient to define a *dependability index*

$$\overline{\lambda}_d = \frac{\int_0^\infty d\epsilon\,(P_d(\epsilon) - P_{d-1}(\epsilon))}{\int_0^\infty d\epsilon\,(1 - P_0(\epsilon))}, \quad d = 1, 2, \ldots \tag{8}$$

In general $1 \geq \overline{\lambda}_d \geq 0$, while $\overline{\lambda}_d = 1$ (or $\sum \overline{\lambda}_d = 1$) signals a completely deterministic relationship and $\overline{\lambda}_d \approx 0$ singles out irrelevant variables[7].

Constructing statistical quantities out of pairs of points is an efficient utilization of available statistics ($N(N-1)/2$ pairs out of N points). Nevertheless, limited statistics can be problematic, especially if noise levels are high. Statistical errors are estimated

[7]One should keep in mind however that these conditions are only necessary but not sufficient.

as

$$\Delta P_d(\epsilon|\vec{\delta}) = 2\sqrt{\frac{P_d(1-P_d)}{n(\vec{l} \leq \vec{\delta})}}. \tag{9}$$

This expression is not entirely adequate for correlated data, but it serves the purpose to signal the onset of statistically unreliable regions. In case statistics are at a premium an option is often utilized such that a variable k once identified as irrelevant is set *inactive* , which means that the condition $l_k \leq \delta$ is omitted when computing $P_d(\epsilon|\vec{\delta})$ for $d > k$. This option cuts down the loss of statistics due to unnecessarily restrictive conditions.

3 Feedforward ANN for System Identification

Feedforward ANN have turned out to be a very powerful approach for classification problems. A general introduction to the subject can be found in ref. [4]. Recently also system identification problems have been approached with these nonlinear techniques. The aim is to realize a function mapping F_i from the input values x_k to the output values y_i. For the so-called Multilayer Perceptron (**MLP**) [3] a particular form of the function F_i is chosen

$$y_i = F_i(x_1, x_2, ...) = g\left(\sum_j \omega'_{ij} g(\sum_k \omega_{jk} x_k)\right) \tag{10}$$

which corresponds to the feed-forward architecture of fig. 2. The parameters to be fitted are the "weights" ω'_{ij} and ω_{jk}. The "transfer function" g is often chosen as $g(x) \propto \tanh(x)$. The input nodes could be either lagged outputs (e.g. $y(t-1)$,

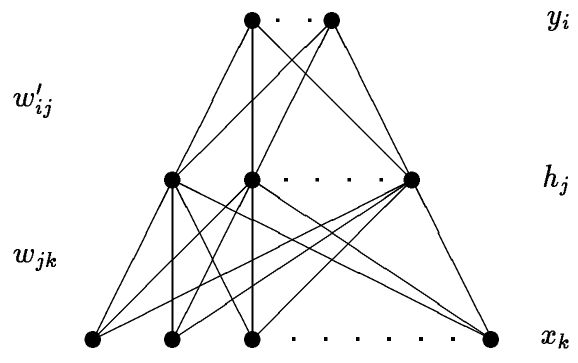

Figure 2: A one hidden layer feed-forward neural network architecture.

$y(t-2)$,..) or other independent variables at time t. The hidden layer enables the network to handle non-linear dependencies with threshold behavior given by $g(x)$. In eq. (10) and fig. 2 one hidden layer is assumed. The architecture can of course be generalized to any number of hidden layers. Fitting to a given data set (or "learning") takes place with gradient descent on e.g. a summed square error

$$E = \frac{1}{2}\sum_i (y_i - t_i)^2 \tag{11}$$

with respect to the weights ω'_{ij} and ω_{jk}, where t_i are the desired (true) output values. This is done by presenting all the **training patterns** repeatedly with successive adjustments of the weights. Once this iterative learning has reached an acceptable level in terms of a low error E, the weights are frozen and the ANN is ready to be used on patterns it has never seen before. The capability of the network to correctly reproduce the mapping of these **test patterns** is called **generalization** performance. In the context of utility prediction the training patterns are historic data and the test patterns represent the independent variables in the prediction part.

This MLP functional expansion contains linear modeling as a special case (linear output and no hidden nodes). It differs, however, from polynomial fittings where each additional power introduced implies a new dimension in an orthonormal space. With few training patterns this might give rise to "overfitting" with degradation in generalization performance. This phenomenon has been observed for the present data. In contrast, adding additional hidden nodes in the MLP sigmoidal expansion does not necessarily "open up" new dimensions – additional hidden nodes may well duplicate the task of existing ones.

The training phase is often terminated before the global minimum of the error (eq. (11)) has been reached, in order to increase the generalization performance. This is most easily done by monitoring the error on a **validation set** (a subset of the training data which is not used in the training) and stop the training when this error stops decreasing.

An alternative method is to use a **recurrent network** [5] that is capable of building an internal memory of time lagged states by using feedback structures. However, the exact nature of these time lagged states is difficult to analyze and there is no evidence that those states always provide optimum time lags for solving the problems at hand.

When comparing MLP with recurrent networks the former requires preprocessing in terms of choosing appropriately time-lagged inputs while the latter approach is supposed to select the relevant time-lags dynamically. With the δ-test in our hands the appropriate time-lags can be efficiently selected for MLP processing. We find that with such cautious choice of input representation, the MLP always outperforms recurrent networks. Hence from now on we will stick to the MLP.

An additional bonus of the δ-test is that the residual errors (eq. (11)) can be analyzed in terms of dependency on the input variables: With appropriate choice of input representations and an efficient learning procedure, there should be no such residual dependencies.

4 Data Set A

4.1 General Properties of Data

This set represents real world data taken hourly during Sept. - Dec. 1989. The task is to predict $\vec{y}(t)$ for the subsequent period of Jan. - Feb. 1990 from the known measurements on $\vec{x}(t)$, where

y_1 = whole building electric (**WBE**) power consumption (kW)
y_2 = whole building cold water (**WBCW**) consumption (10^6 Btu/hr)
y_3 = whole building hot water (**WBHW**) consumption (10^6 Btu/hr)

x_1 = wind speed (miles/hr)
x_2 = solar flux (W/m^2)
x_3 = humidity ratio (water/dry air)
x_4 = temperature (°F)
x_5 = hour

The corresponding dates are also provided. The Christmas holiday is extended and the building seems to have been "shut down" on the 23:rd of December when the power consumption decreases suddenly and sharp transients appear in the water consumptions. This presents a complication when fitting the data since it occurs towards the end of the training set and the transfer to "normal" running (assuming that it is only a temporary state) takes place in January, which is part of the unknown test set. In order to account for this and other possible seasonal behavior we identify the weekdays and holidays (Thanksgiving, Christmas-New Year) and introduce two new variables

x_6 = *weekday*
x_7 = *day-code*

where the *weekday* takes values ranging from 1 to 7 for Monday through Sunday, and the *day-code* is $+1$ if it is a working day and -1 for a weekend day or a holiday. For the WBE power consumption, which shows clear seasonal effects within a week cycle and before and after holidays, we code *weekday* as a "Sunday" if it happens to be a holiday, and the two working days immediately preceding a holiday are coded as "Thursday" and "Friday" respectively. Furthermore, the *day-code* is given a value of -2 for the Christmas recess (Dec. 23 - Jan. 1) and is decreased from its normal value by 0.4 for the first week in September, January and immediately before Christmas. It is also decreased from its normal value by 0.2 for the second week in September, January and before Christmas.

This additional encoding of information (x_6, x_7), constitutes a heuristic departure from the pure "black box" strategy. This encoding is necessary to enable the network to recognize the different patterns on holidays. If the building is a commercial building, with people working inside, one can imagine that the code gives information on the number of people working the first week after a long holiday, on the last working day before a holiday, etc.. Ideally, this information should be available to the modeller in the form of statistics of working hours.

In the test set one finds the *President's Day* (20 Feb. 1990), which causes an ambiguity since even though it is an officially observed holiday it may be ignored by e.g. certain university campus buildings like sport halls. We have chosen not to treat it as a holiday. Making a wrong assumption here can change the CV and MBE values (see section 4.4) by roughly 5 % for the power consumption and 1 % for the water consumptions.

4.2 Variable Dependencies

The functions $\vec{y}(t) = F(\vec{x}(t))$, which are determined from variables at the same time step only, are said to have a **horizontal** dependency only. In cases where there are dependencies upon the history of variables one has **vertical** (time lag) dependencies. Prior to fitting the data with an ANN we used the δ-test extensively to investigate these dependency structures.

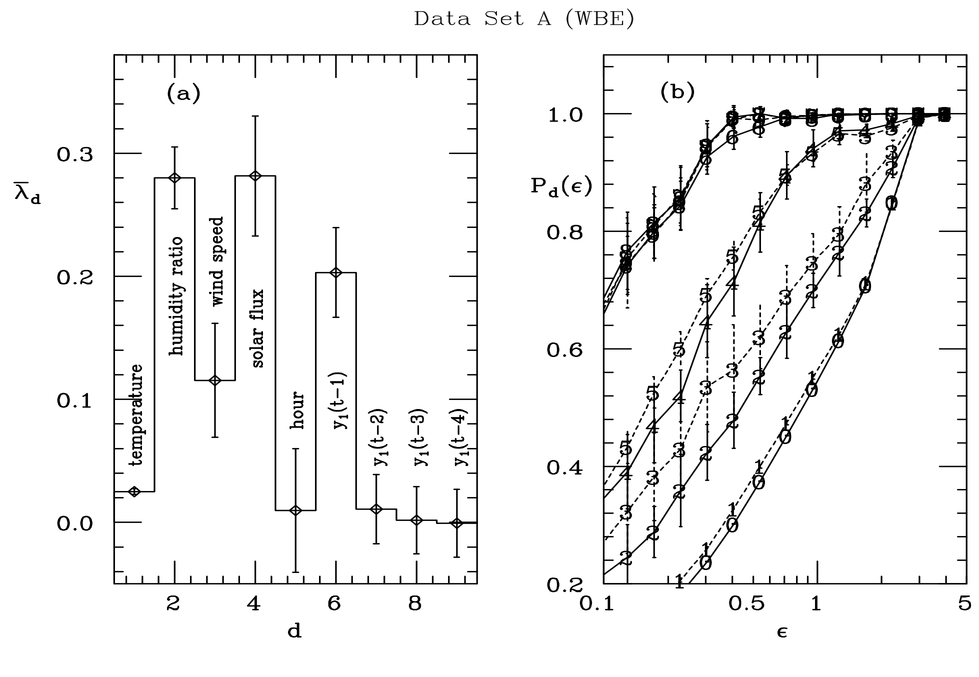

Figure 3: (a): Dependency indices $\overline{\lambda}_d$ for y_1 (*WBE*) on x_4 ($d = 1$), x_3 ($d = 2$), x_1 ($d = 3$), x_2 ($d = 4$), x_5 ($d = 5$), $y_1(t-1)$ ($d = 6$), $y_1(t-2)$ ($d = 7$), $y_1(t-3)$ ($d = 7$) and $y_1(t-4)$ ($d = 8$). (b): The probability $P_d(\epsilon)$ as a function of ϵ for various d as marked on the curves.

All three consumption variables (power, hot and cold water) exhibit strong dependencies on x_2 (solar flux). The water consumptions also show large dependencies on temperature whereas the power consumption appears to depend largely on the combination of temperature and humidity. As an example the dependability indices for the power consumption variable are shown in fig. 3a. Clearly the first seven variables listed in the figure can be regarded as relevant. In particular the 1-lag variable $y_1(t-1)$ is a very important one, while time lags beyond three can be discarded. It is also interesting to estimate the noise level with this set of variables. It is seen on fig. 3b that $P_d(\epsilon)$ starts to deviate from 1 at $\epsilon_0 \sim 0.4$ (c.f. fig. 1b). Assuming a Gaussian distribution for the noise, its standard deviation can be estimated as $\sigma_r \approx \epsilon_0/3.6 = 0.11$. Since the ϵ in fig. 3b is given in unit of the standard deviation σ of the WBE series, this means that the mean error one may achieve is 0.11σ. Given that $\sigma = 149$ and the $mean = 686$ for the WBE data we used, the mean error translates into a $CV = 0.024$ (see eq. (13)), a number comparable with the actual error we get out of an MLP.

4.3 MLP Architecture and Parameters

For the WBE prediction we use an MLP with 7 hidden sigmoidal units, one linear output unit and 13 inputs;

$$\begin{aligned}\vec{x}(t) &= (y_1(t-1), y_1(t-2), x_1(t), x_2(t), x_3(t), x_4(t), \sin(\pi x_5(t)/12), \cos(\pi x_5(t)/12),\\ &\quad \sin(\pi x_6(t)/7), \cos(\pi x_6(t)/7), x_7(t), x_7(t-24), x_7(t+24))\end{aligned}$$

where the last three inputs correspond to the *day-codes* for *today*, *yesterday* and *tomorrow* respectively.

For the WBCW prediction we use an MLP with 11 hidden sigmoidal units, one linear output unit and 20 inputs;

$$\begin{aligned}\vec{x}(t) &= (y_2(t-1), x_1(t), x_2(t), x_3(t), x_4(t), x_1(t-1), x_2(t-1), x_3(t-1), x_4(t-1),\\ &\quad x_1(t-2), x_2(t-2), x_3(t-2), x_4(t-2), \sin(\pi x_5(t)/12), \cos(\pi x_5(t)/12),\\ &\quad \sin(\pi x_6(t)/7), \cos(\pi x_6(t)/7), x_7(t), x_7(t-24), x_7(t+24))\end{aligned}$$

Finally, for the WBHW prediction we use an MLP with 7 hidden sigmoidal units, one linear output unit and 13 inputs;

$$\begin{aligned}\vec{x}(t) &= (y_3(t-1), y_3(t-2), x_1(t), x_2(t), x_3(t), x_4(t), x_2(t-1), \sin(\pi x_5(t)/12),\\ &\quad \cos(\pi x_5(t)/12), 7 - x_6(t), x_7(t), x_7(t-24), x_7(t+24))\end{aligned}$$

All input values are normalized to the interval [0,1]. Standard backpropagation, as implemented in `JETNET 2.0` [6], is used with initial weights randomly distributed in the interval [-0.1,0.1]. The learning rate and momentum term are set to 0.1 and 0.0 respectively. The gradient is sampled over 20 randomly selected patterns for each weight update.

4.4 Results

The data set includes a total of 4208 time steps (hours). The last 1282 time steps make up the test set, in which the independent variables $\vec{x}(t)$ are given whereas the energy consumptions $\vec{y}(t)$ were withheld by the organizers [1]. We thus have at our disposal the data patterns [1,2926] that can be used to train the networks. The accuracy in predicting the unknowns in the test set [2927,4208] is used by the organizers to score the generalization performances.

We use the usual mean squared error (MSE)

$$\frac{1}{N}\sum_{t=1}^{N}(\hat{y}(t) - y(t))^2 \tag{12}$$

to gauge the network performances. Since the previous signal value $y(t-1)$ is used as input in all three cases, two types of errors are to be distinguished: One can make predictions based on true $y(t-1)$ values whenever these are available. The resultant error is then called the single step error (**S-MSE**). Alternatively, one can use the predicted values of y iteratively to make further predictions by feeding the predicted output back as an input. The resultant error is called the multi-step error (**M-MSE**). The iterative approach is used for making predictions in the test set.

The networks are trained in two stages:

1. First a sample of 500 time steps [801,1300] are reserved from the training set to form a validation set. The networks are trained on the remaining 2426 data points. The multi-step errors are monitored and the network giving the best M-MSE on the validation set is picked out. We then make preliminary predictions for the time steps [2927,4208] based on the best network found.

2. In the second stage we train the network on the entire training set (including the 500 time steps previously reserved), with the condition that the new predictions may not drift too far from the preliminary predictions. Specifically we define a mean squared deviation (**MSD**) between the preliminary and the new predictions, and we finally choose the network which minimizes a weighted sum of M-MSE on training set and M-MSD on the testing set.

We find that in the stage 1 training the best M-MSE selected by the validation process does not always correspond to the best M-MSE for the entire training set. The stage 2 "fine tuning" process ensures that we do not get a solution that misrepresents the 500 reserved patterns badly.

The mean squared errors from the best networks achieved are summarized in table 1.

The organizers have defined the coefficient of variation (**CV**) and the mean bias error

Data Set		Stage I Learning (1-800,1301-2926)	Stage I Validation (801-1300)	Stage II Training (1-2926)	Stage II MSD (2927-4208)
WBE	S-MSE	228.3	152.3	203.0	5.43
	M-MSE	881.9	381.3	726.8	61.15
WBCW	S-MSE	0.030	0.034	0.030	0.0009
	M-MSE	0.163	0.092	0.128	0.015
WBHW	S-MSE	0.047	0.038	0.045	0.0002
	M-MSE	0.121	0.056	0.100	0.0044

Table 1: The mean squared errors for single step (S-MSE) and multi-step (M-MSE) predictions for various data sets at the stage I and stage II of the network training. The last column (MSD) is the deviation of the stage II predictions from the preliminary stage I predictions.

(**MBE**) as the following

$$\begin{aligned} \mathrm{CV} &= \frac{1}{\overline{y}}\left[\frac{1}{N}\sum_{t=1}^{N}(\hat{y}(t)-y(t))^2\right]^{1/2}, \\ \mathrm{MBE} &= \frac{1}{\overline{y}}\frac{1}{N}\sum_{t=1}^{N}(\hat{y}(t)-y(t)) \end{aligned} \tag{13}$$

where $y(t)$ = the true value of the signal, $\hat{y}(t)$ = the predicted value, and $\overline{y}$ = the average of the true value. Our results for these two error measures on the training set and the test set are given in table 2.

	Training Set				Test Set	
	CV		MBE		CV	MBE
	S. Step	M. Step	S. Step	M. Step	M. Step	M. Step
WBE	0.0215	0.0407	-3.25×10^{-4}	-1.26×10^{-5}	0.1178	0.105
WBCW	0.0345	0.0705	-7.55×10^{-4}	-4.86×10^{-3}	0.1296	-0.0595
WBHW	0.1010	0.1512	-3.08×10^{-4}	-1.70×10^{-3}	0.3063	-0.2733

Table 2: The coefficient of variations (CV) and the mean bias errors (MBE) for the single step and multi-step predictions, and for the training and test set, respectively.

In figs. 4–6 we show the comparison between the (multi-step) predictions and the true values. The bottom plot in each figure are for the test sets in which the true values became known only after the predictions were made, and demonstrate the true generalization performance of the networks. Fig. 7 shows the predicted energy consumptions as functions of the temperature.

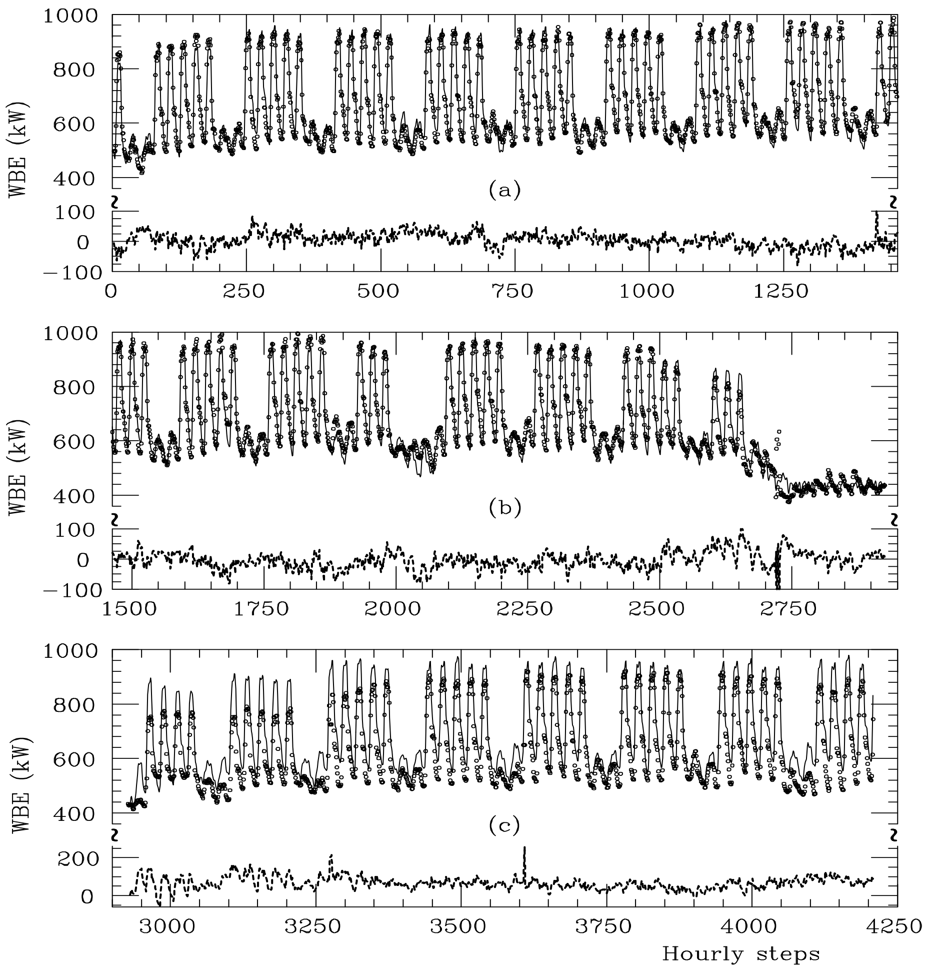

Figure 4: Predicted WBE consumption (the full lines) compared with data (the points). The residues (Prediction − Data) are shown in broken lines. (a) and (b) are for the training set (time steps 1 to 2926), and (c) is for the test set (time steps 2927 to 4208).

4.5 Validation of the Result

As mentioned above, the δ-test can be used to check whether the network has learned its task properly by applying the test on the residual error. If the amount of noise resulting from the δ-test equals the standard deviation of the residue signal, then it is very likely that the prediction is the best possible prediction (apart from minor differences between different networks). We stress however that the δ-test is based on true data values and therefore has direct relevance only to the single step prediction task.

The results from testing the single step residues indicate that for WBE and WBCW consumptions our networks have pretty much reached their limits for learning the

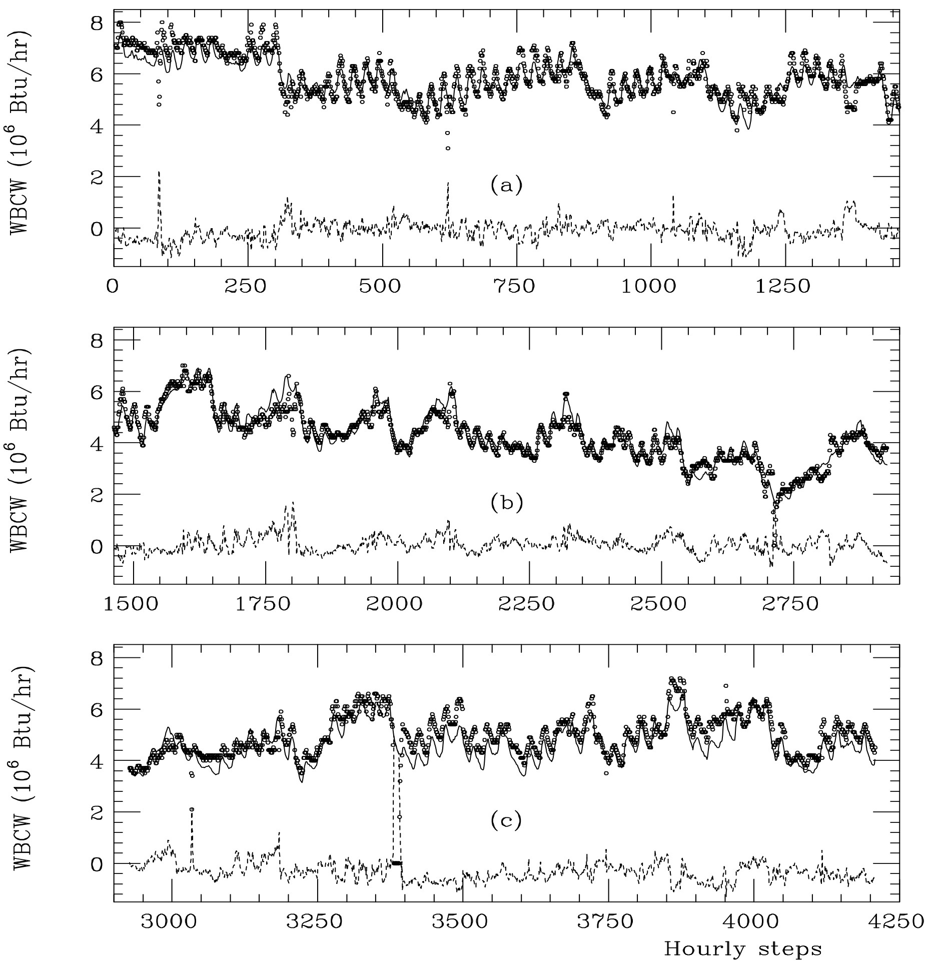

Figure 5: Predicted WBCW consumption plotted together with the data and the residues. The notations are the same as in fig. 4.

tasks, while for WBHW there may still be room for some improvement. However we find that for WBHW the best single step error does not necessarily correspond to the best multi-step error. Since the latter is most relevant for this prediction task, we choose to ignore the signal of the δ-test and accept the network solution.

5 Data Set B

5.1 General Properties of Data

This set consists of 3344 solar radiation measurements during a sixth month period (August - May). The task is to predict $y(\vec{x})$ where

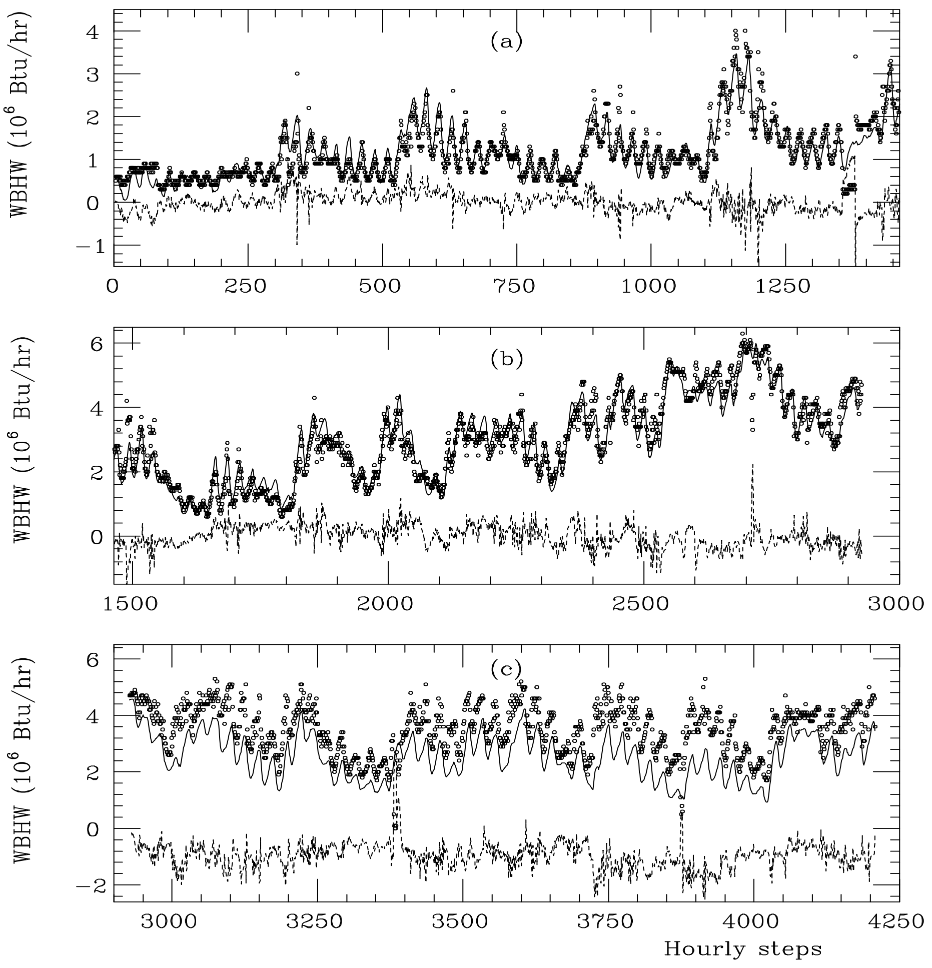

Figure 6: Predicted WBHW consumption plotted together with the data and the residues. The notations are the same as in fig. 4.

y =true beam insolation

x_1 = decimal date (Julian day + hour/24)
x_2 = Horizontal solar flux (W/m^2)
x_3 = South-East solar flux (W/m^2)
x_4 = South solar flux (W/m^2)
x_5 = South-West solar flux (W/m^2)

From a physics point of view it would be natural if the horizontal, south and total solar flux variables were all peeked around noon. However, visual inspection of the data according to the description is not consistent with this. Since the notation of the input values is irrelevant for the results, we stick to the "shootout" labels in our

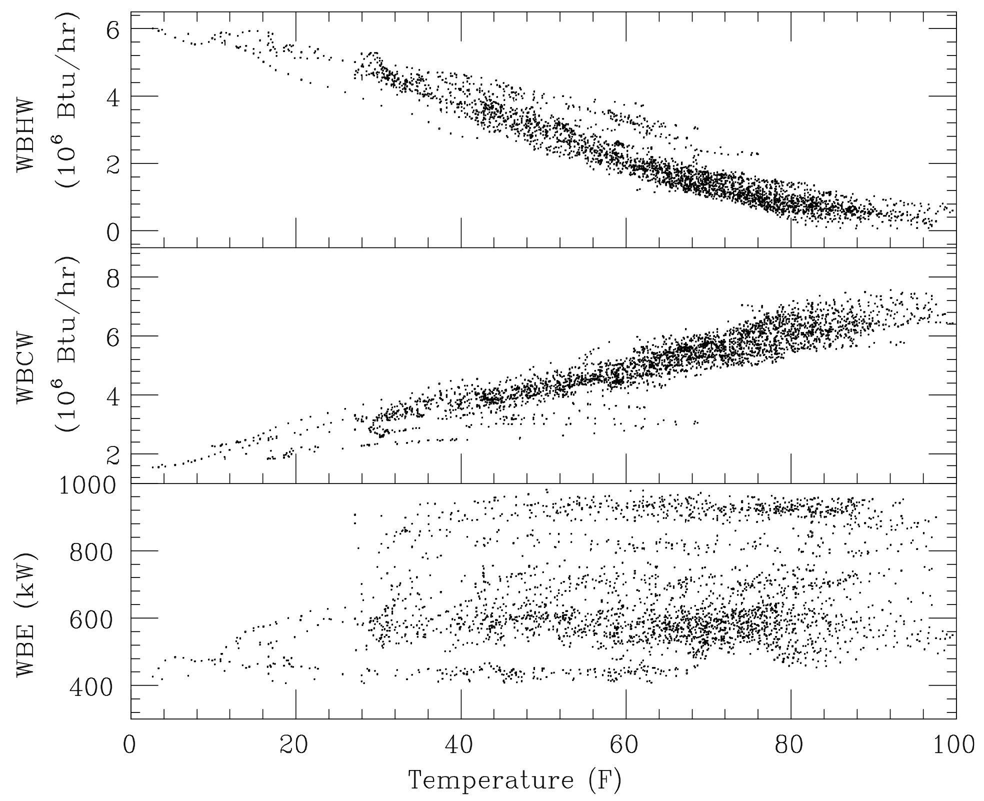

Figure 7: Predicted WBE, WBCW and WBHW consumption versus the D.B. Temperature, respectively.

description and calculations.

5.2 Variable Dependencies

The solar flux has an obviously stronger dependency on the hour than on the (Julian) day and we therefore split the decimal date column into a day and an hour column. Results from applying the δ-test [2] on this data set are shown in fig. 8. The y variable has dependencies on the four angled measures of solar flux as well as on the hour, as seen in fig. 8a. However if the variables are reordered so that the hour is entered the last, as shown in fig. 8c, one sees that the hour does not provide information that is extra to the four solar flux variables. This suggests that if a model is properly built upon the angled solar flux measures there may be no need for an explicit dependency on the hour. There appears to be no dependency on the (Julian) day either. A striking feature is that the sum of indices gives 0.998 in either fig. 8a or fig. 8c, nearly saturating to 1, which suggests a high deterministic relationship. This is in contrast to the data set A (fig. 3) where the sum of the indices yields 0.92 and a relatively large

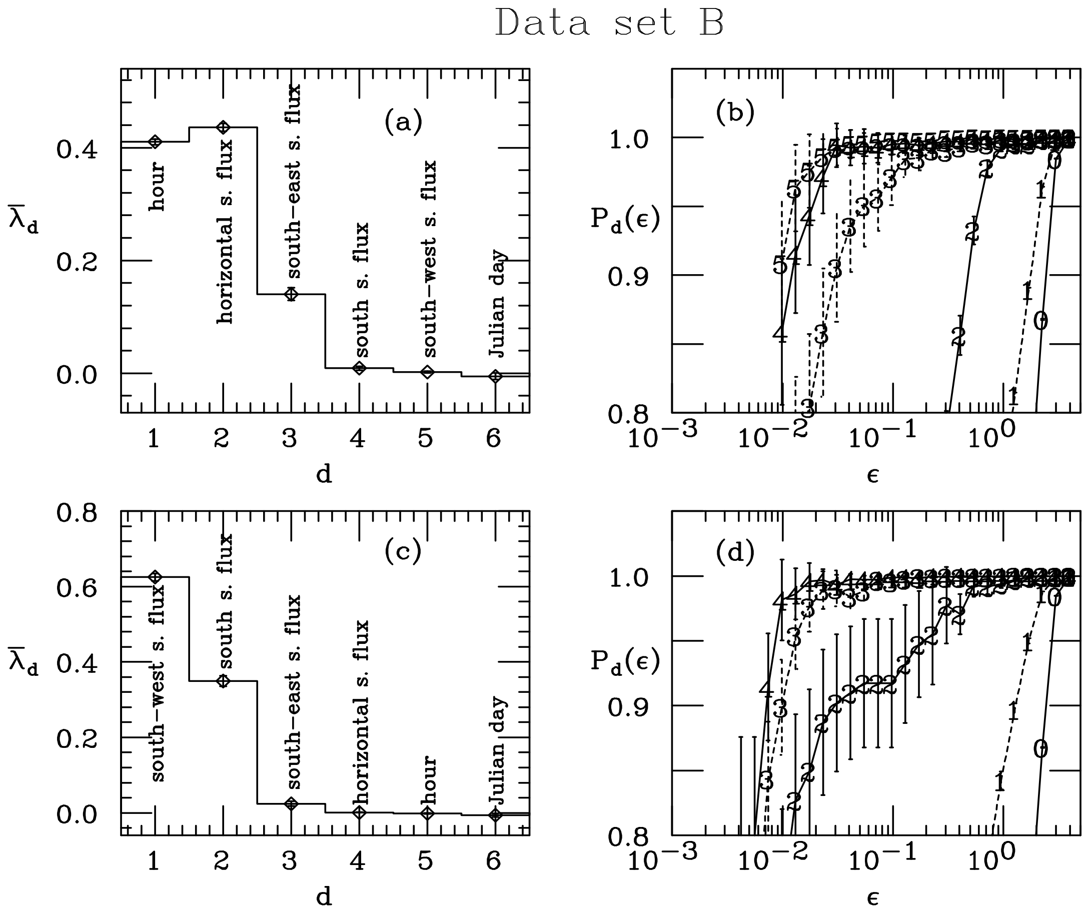

Figure 8: (a): Dependency indices $\overline{\lambda}_d$ for y (*true beam insolation*) on the independent variables. (b): The probability $P_d(\epsilon)$ as a function of ϵ for the same set of variables in (a); d is marked on the curves. (c) and (d): The same as in (a) and (b) but for a different ordering of variables.

noise level is found. From fig. 8b (or fig. 8d) we read $\epsilon_0 \sim 0.03$, and the noise level $\sigma_r \approx \epsilon_0/3.6 = 0.0083$. Multiplying by the standard deviation/mean ratio (316/381) of the y data set, the noise would correspond to an error measure $CV \sim 0.007$.

5.3 ANN Method

Given that there exists a function that fully determines the total solar flux we attempted some different ANN architectures to approximate it. First we used a small subset of the training data as a validation set to monitor the performance. It turned out that the error on the validation set always decreased with the error on the training set and that there is no need to use a validation set. We consequently chose to use the whole training set for training. The ANN architecture that gave the best performance is a standard MLP with 2 hidden layers, a single output unit and 7 inputs, where the inputs are

$$\vec{x} \;=\; (x_2, x_3, x_4, x_5, day, \cos(hour), \sin(hour)).$$

We use no time-lagged inputs since the δ-test suggests that the function is well determined by just using horizontal variables. All input values are normalized to the interval $[0,1]$ and the output value is scaled by a factor of 1300. Initial weights are randomly distributed in the interval $[-0.3, 0.3]$. During learning weights are updated after presentation of every 10 randomly selected patterns. All the calculations have been done using the F77 package `JETNET 2.0` [6]. A Langevin updating scheme (see e.g. ref [7]) was used as it often performs better than backpropagation.

5.4 Results

In fig. (9) a scatter plot is shown of true and predicted beam insolation for the training data, with a variation coefficient CV $= 0.019$ and a mean bias error MBE $= -0.00032$. Note that the CV achieved by the network is reasonably close to the rough estimate of the δ-test. For the test data (a set of data in which the answers were withheld by the organizers), the error measures give CV $= 0.027$ and MBE $= 0.0017$.

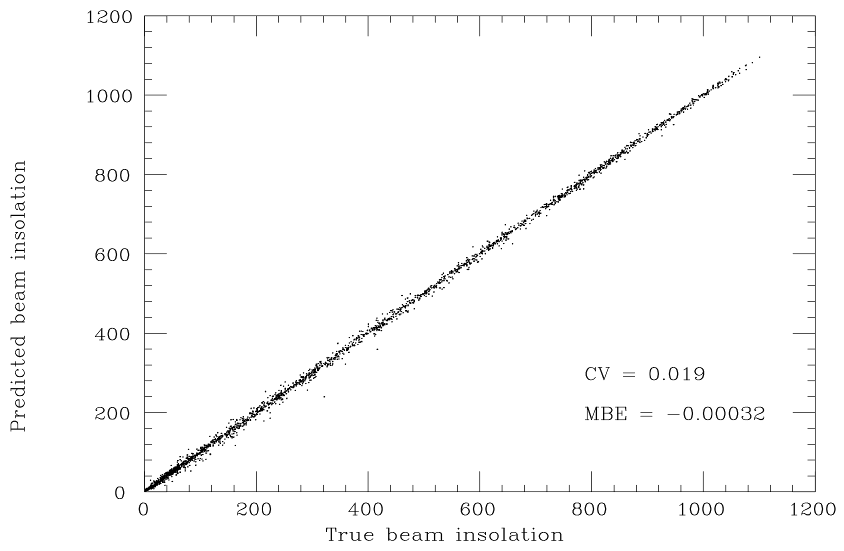

Figure 9: Scatter plot showing predicted beam insolation versus true beam insolation for the training data of set B.

5.5 Validation of the Result

Applying the δ-test to the residue ϵ from the training set gives

$$\epsilon = \hat{G}(day, \cos(hour), \sin(hour), v_2, ..., v_5) + \gamma \quad , \tag{14}$$

where the noise level $\gamma \sim$ stddev(ϵ). This implies that another network trained on the residue signal will only be able to learn within an error of about one standard deviation, which is no improvement. We hence conclude that our prediction is good enough.

6 Summary

We have approached the two data sets provided in "The Great Energy Predictor Shootout - The First Building Data Analysis and Prediction Competition" [1] with an almost "black-box" procedure based upon

- The δ-test for establishing dependencies and gauging network performance.
- A Multilayer Perceptron (MLP) [3] for modeling historic data.

When selecting the appropriate ANN architecture and learning algorithm the choices are a standard MLP and/or a recurrent network [5]. The former requires preprocessing in terms of choosing appropriately time-lagged inputs whereas the latter approach is supposed to select the relevant time-lags dynamically. With the δ-test in our hands the appropriate time-lags can be efficiently selected for MLP processing.

It should be stressed that some "expert" knowledge of holiday structure etc. is needed for peak performance in data set A.

Acknowledgements:

We would like to thank Jeff Sultan, Gordon-Prill, Inc., for sharing his knowledge about utility consumption patterns with us. This research was sponsored in part by the Göran Gustafsson Foundation for Research in Natural Science and Medicine.

References

[1] J.F. Kreider and J.S. Haberl, "The Great Energy Predictor Shootout - The First Building Data Analysis and Prediction Competition", this volume.

[2] H. Pi and C. Peterson, "Finding the Embedding Dimension and Variable Dependencies in Time Series", *LU TP 93-4* (to appear in *Neural Computation*).

[3] D. E. Rumelhart and J. L. McClelland (eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition (Vol. 1)*, MIT Press (1986).

[4] J. Hertz, A. Krogh and R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, Redwood City, Ca. (1991).

[5] R.J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks", *Neural Computation* **1**, 270 (1989).

[6] L. Lönnblad, C. Peterson and T. Rögnvaldsson, "Pattern Recognition in High Energy Physics with Artificial Neural Networks - JETNET 2.0", *Computer Physics Communications* **70**, 167 (1992).

[7] T. Rögnvaldsson, "On Langevin Updating in Multilayer Perceptrons", *LU TP 93-13* (to appear in *Neural Computation*).