PREDICTING SYSTEM LOADS WITH ARTIFICIAL NEURAL NETWORKS—Methods and Results from "The Great Energy Predictor Shootout"

Mattias B.O. Ohlsson **Carsten O. Peterson, Ph.D.** **Hong Pi, Ph.D.**

Thorsteinn S. Rögnvaldsson **Bo P.W. Söderberg, Ph.D.**

ABSTRACT

A feed-forward artificial neural network (ANN) procedure has been devised for predicting utility loads; the resulting predictions are presented for two test problems given by "The Great Energy Predictor Shootout—The First Building Data Analysis and Prediction Competition" (Kreider and Haberl 1994). Key ingredients in this approach are the multilayer perceptron and a method (δ-test) for determining relevant inputs. These methods are briefly reviewed, together with comments on alternative schemes such as fitting to polynomials and the use of recurrent networks.

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" (Kreider and Haberl 1994). The present approach scored first on set B and second 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 (Pi and Peterson 1993) and
- feedforward artificial neural networks (ANN) for model building (system identification).

This approach is of a "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 the next two sections, the δ-test approach and the ANN method, respectively, are reviewed, 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 sets A and B are discussed in the following two sections. The last section contains a brief summary.

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, whose inputs, such as the noise level, etc., are most relevant. 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 (Pi and Peterson 1993), to be briefly described later, aims to determine 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 ($z_1, z_2, \ldots, z_m$). 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

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

where r represents an indeterminable part that 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.*

Mattias B.O. Ohlsson and **Thorsteinn S. Rögnvaldsson** are doctoral students, **Carsten O. Peterson** is a professor, **Hong Pi** is a research fellow, and **Bo P.W. Söderberg** is an associate professor in the Department of Theoretical Physics at Lund University, Lund, Sweden.

The problem is approached 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 ($d = 0, 1, 2, \ldots$):

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

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

$$l_k(i,j) = |z_k(i) - z_k(j)|, \; k = 0, 1, \ldots, d, \quad (3)$$

one can construct the conditional probabilities

$$P_d(\varepsilon \mid \vec{\delta}) \equiv P(l_0 \le \varepsilon \mid \vec{l} \le \vec{\delta}) = \frac{n(l_0 \le \varepsilon, \vec{l} \le \vec{\delta})}{n(\vec{l} \le \vec{\delta})}, \quad (4)$$

where ε and δ are positive numbers and $n(\vec{l} \le \vec{\delta})$ and $n(l_0 \le \varepsilon, \vec{l} \le \vec{\delta})$ are the numbers of vector pairs satisfying the corresponding distance constraints.[1] How does the probability structure of the dependent variable behave with respect to the independent variables? In Pi and Peterson (1993), the following important observations were made.

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

$$P_0(\varepsilon) = P_1(\varepsilon \mid \vec{\delta}) = \ldots = P_d(\varepsilon \mid \vec{\delta}) = \ldots. \quad (5)$$

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 Equation 1) with no intrinsic noise, then for any $\varepsilon > 0$ there exists a δ_ε such that

$$P_d(\varepsilon \mid \vec{\delta}) = 1 \text{ for } \delta \le \delta_\varepsilon \text{ and } d \ge d_0 \quad (6)$$

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

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

The behavior of $P_d(\varepsilon \mid \vec{\delta})$ as a function of δ and ε for various d is shown schematically in Figure 1. The consequences of randomness (Equation 5) and complete determination (Equation 6) provide a yardstick with which to measure the degree of dependency between the variables. Interesting quantities to examine are the maxima:

$$P_d(\varepsilon) \equiv \max_{\delta > 0} P_d(\varepsilon \mid \vec{\delta}) = P_d(\varepsilon \mid \vec{\delta})|_{\delta \le \delta_\varepsilon}. \quad (7)$$

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

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

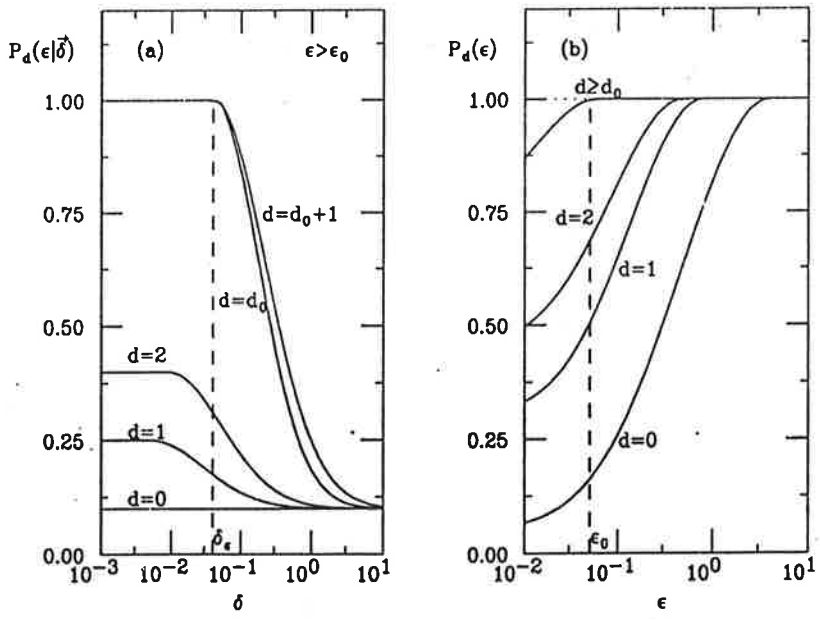

Figure 1 *(a) $P_d(\varepsilon \mid \vec{\delta})$ as a function of δ for fixed ε. Saturation to 1 would be observed if the width of noise is less than ε. (b) Behavior of the maxima $P_d(\varepsilon) \equiv \max_{\delta > 0} P_d(\varepsilon \mid \vec{\delta})$ as a function of ε. The region saturating to 1 would be pushed toward smaller ε if the dth 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 $\Delta r_{max} \sim \varepsilon_0$.*

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

In general, $1 \geq \vec{\lambda}_d \geq 0$, while $\vec{\lambda}_d = 1$ (or $\Sigma \vec{\lambda}_d = 1$) signals a completely deterministic relationship, and $\vec{\lambda}_d \approx 0$ singles out irrelevant variables.[2]

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 as

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

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 $P_d(\varepsilon \mid \vec{\delta})$ for $d > k$. This option cuts down the loss of statistics due to unnecessarily restrictive conditions.

FEEDFORWARD ANN FOR SYSTEM IDENTIFICATION

Feedforward ANNs have turned out to be a powerful approach for classification problems. A general introduction to the subject can be found in Hertz et al. (1991). Recently, system identification problems also 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) (Rumelhart and McClelland 1986), a particular form of the function F_i is chosen:

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

which corresponds to the feed-forward architecture of Figure 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., $y(t-1)$, $y(t-2)$, . . .) or other independent variables at time t. The hidden layer enables the network to handle nonlinear dependencies, with threshold behavior given by $g(x)$. In Equation 10 and Figure 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, for example, 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 a 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 (Equation 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 that is not used in the training) and stopping the training when this error stops decreasing.

An alternative method is to use a recurrent network (Williams and Zipser 1989) 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

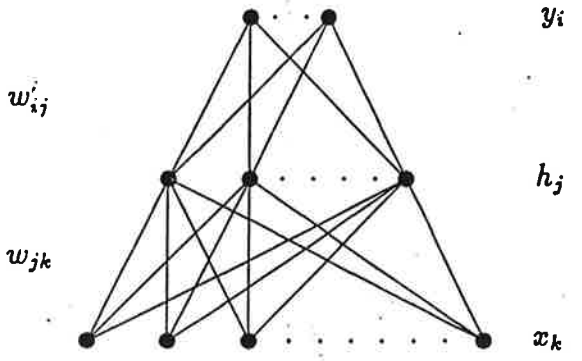

Figure 2 *A feed-forward neural network architecture with one hidden layer.*

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

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, the appropriate time lags can be efficiently selected for MLP processing. With such a cautious choice of input representation, the MLP always outperforms recurrent networks. Hence, from now on we will use the MLP.

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

DATA SET A

General Properties of Data

This set represents real-world data taken hourly from September to December 1989. The task is to predict $\vec{y}(t)$ for the subsequent period of January and February 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/h),
y_3 = whole-building hot water (WBHW) consumption (10^6 Btu/h),
x_1 = wind speed (mph),
x_2 = solar flux (W/m^2),
x_3 = humidity ratio (water/dry air),
x_4 = temperature (°F), and
x_5 = hour.

The corresponding dates are also provided. The Christmas holiday is extended and the building seems to have been shut down on December 23, when the power consumption decreases suddenly and sharp transients appear in the water consumption. This presents a complication when fitting the data because it occurs toward 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 and x_7 = day-code, where the weekday takes values ranging from 1 to 7 for Monday through Sunday, and the day-code is +1 for a working day and −1 for a weekend day or a holiday. For the WBE power consumption, which shows clear seasonal effects within a weekly cycle and before and after holidays, we code a 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 (December 23-January 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 modeler in the form of statistics of working hours.

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

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.

All three consumption variables (power and hot and cold water) exhibit strong dependencies on x_2 (solar flux). Water consumption also shows 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 Figure 3a. Clearly the first seven variables listed in the figure can be regarded as relevant. In particular, the one-lag variable $y_1(t-1)$ is an 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 in Figure 3b that $P_d(\varepsilon)$ starts to deviate from 1 at $\varepsilon_0 \sim 0.4$ (c.f. Figure 1b). Assuming a Gaussian distribution for the noise, its standard deviation can be estimated as $\sigma_0 \approx \varepsilon_0/3.6 = 0.11$. Since the ε in Figure 3b is given in units 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 used, the mean error translates into a CV

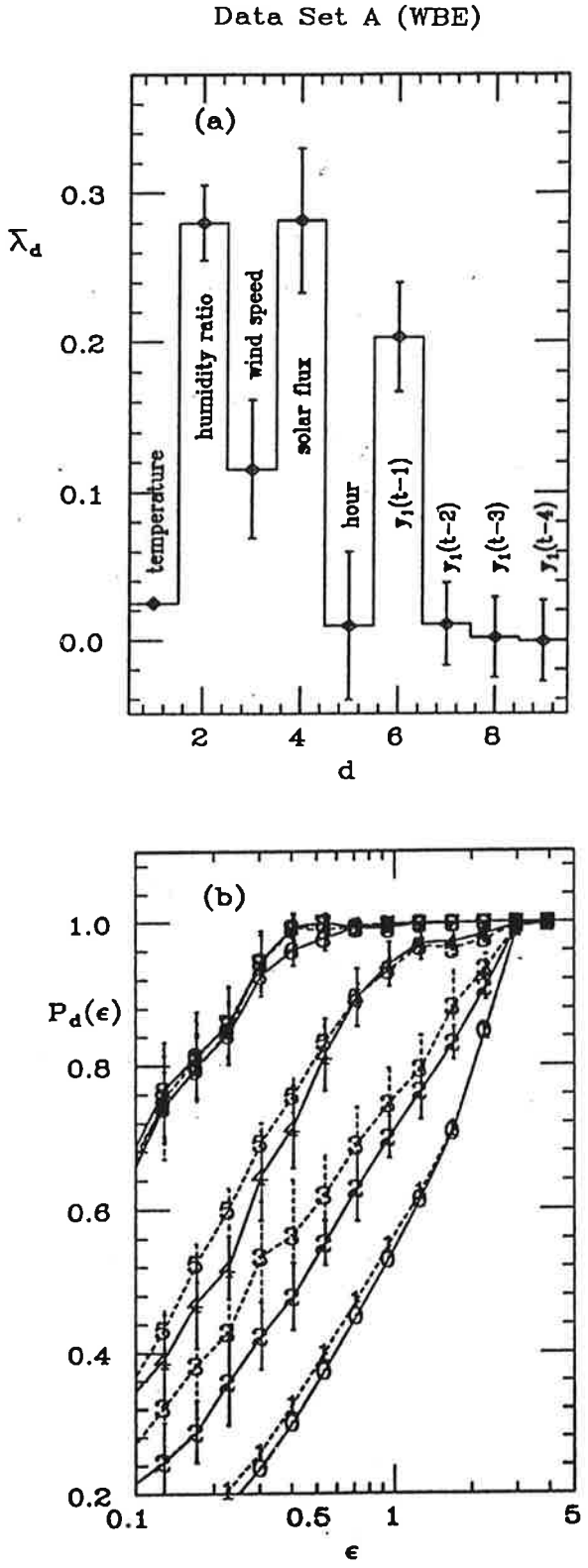

Figure 3 *(a) Dependency indices $\bar{\lambda}_d$ for y_1 (WBE) on x_4 (d = 1), x_3 (d = 2), x_1 (d = 3), x_2 (d = 4), x_5 (d = 5), $y_1(t-1)$ (d = 6), $y_1(t-2)$ (d = 7), $y_1(t-3)$ (d = 7), and $y_1(t-4)$ (d = 8). (b) The probability $P_d(\epsilon)$ as a function of ϵ for various d as marked on the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

= 0.024 (see Equation 13), a number comparable with the actual error obtained from an MLP.

MLP Architecture and Parameters

For the WBE prediction, we use an MLP with 7 hidden sigmoidal units, 1 linear output unit, and 13 inputs:

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

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, 1 linear output unit, and 20 inputs:

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

Finally, for the WBHW prediction, we use an MLP with 7 hidden sigmoidal units, 1 linear output unit, and 13 inputs:

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

All input values are normalized to the interval [0,1]. Standard backpropagation, as implemented in Lönnblad et al. (1992), 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.

Results

The data set includes a total of 4,208 time steps (hours). The last 1,282 time steps make up the test set in which the independent variables $\vec{x}(t)$ are given, whereas the energy consumption $\vec{y}(t)$ was withheld by the organizers (Kreider and Haberl 1994). 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] is reserved from the training set to form a validation set. The networks are trained on the remaining 2,426 data points. The multi-step errors are monitored and the network giving the best M-MSE on the validation set is picked out. Then, preliminary predictions are made 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 finally choose the network that minimizes a weighted sum of M-MSE on the 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 badly misrepresents the 500 reserved patterns.

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 (MBE) as the following:

$$\mathrm{CV} = \frac{1}{\bar{y}}\left[\frac{1}{N}\sum_{t=1}^{N}(\hat{y}(t)-y(t))^2\right]^{1/2}$$

and

$$\mathrm{MBE} = \frac{1}{\bar{y}}\frac{1}{N}\sum_{t=1}^{N}(\hat{y}(t)-y(t)) \tag{13}$$

where

$y(t)$ = true value of the signal,
$\hat{y}(t)$ = predicted value, and
$\bar{y}$ = average of the true value.

The results for these two error measures on the training set and the test set are given in Table 2.

Figures 4 through 6 show the comparison between the (multi-step) predictions and the true values. The bottom plots in the figure are for the test sets in which the true values became known only after the predictions were made; they demonstrate the true generalization performance of the networks. Figure 7 shows the predicted energy consumption as functions of the temperature.

TABLE 1
Mean-Squared Errors for Single-Step (S-MSE) and Multi-Step (M-MSE) Predictions for Various Data Sets at 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.)

Data Set		Stage I		Stage II	
		Learning (1-800,1301-2926)	Validation (801-1300)	Training (1-2926)	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 2
Coefficient of Variations (CV) and Mean Bias Errors (MBE) for the Single-Step and Multi-Step Predictions and for the Training and Test Sets, Respectively

	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

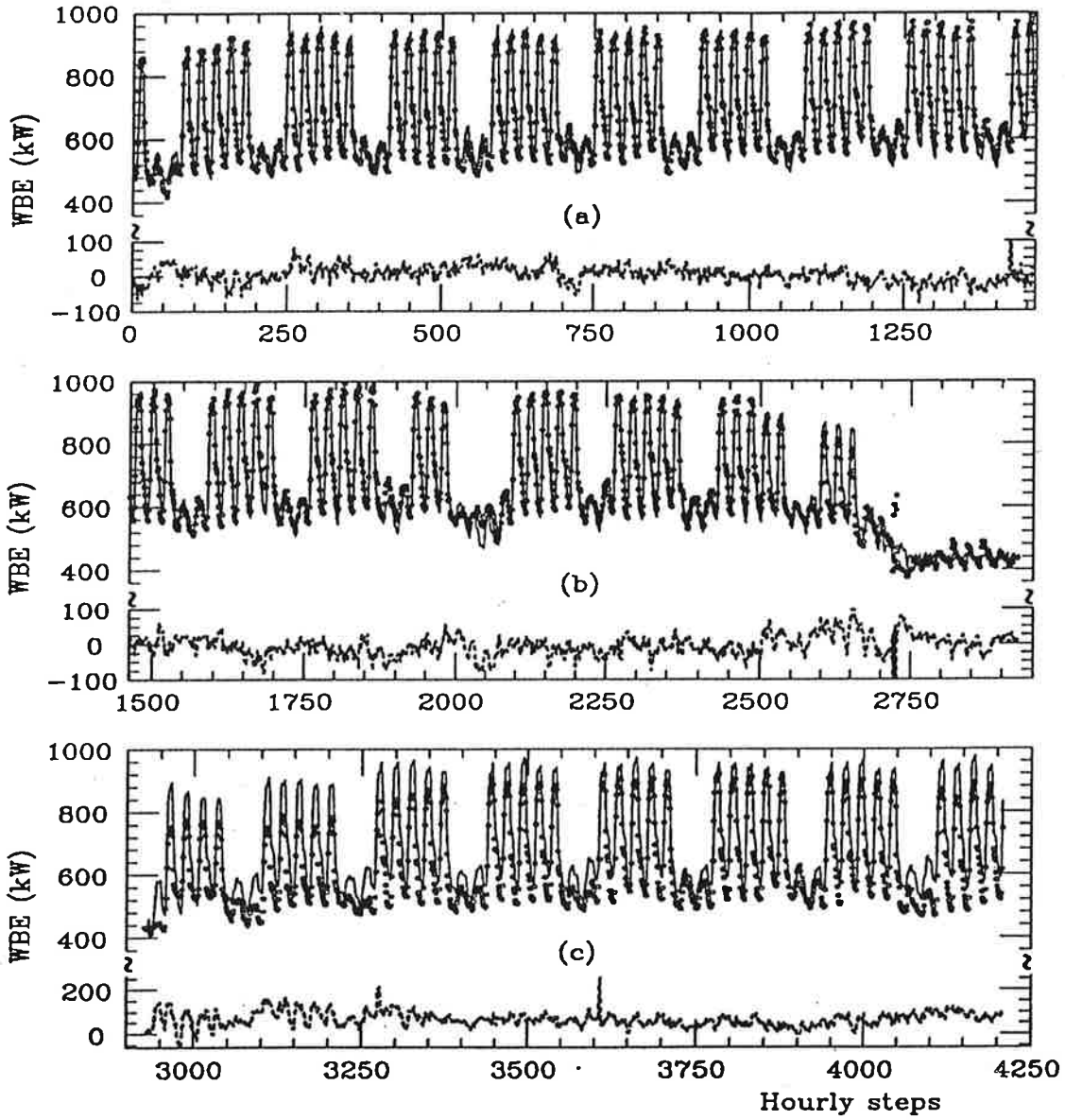

Figure 4 *Predicted WBE consumption (solid lines) compared with data (points). The residues (Prediction—Data) are shown as broken lines. (a) and (b) are for the training set (time steps 1 to 2926) and (c) is for the test set (time steps 2927 to 4208).*

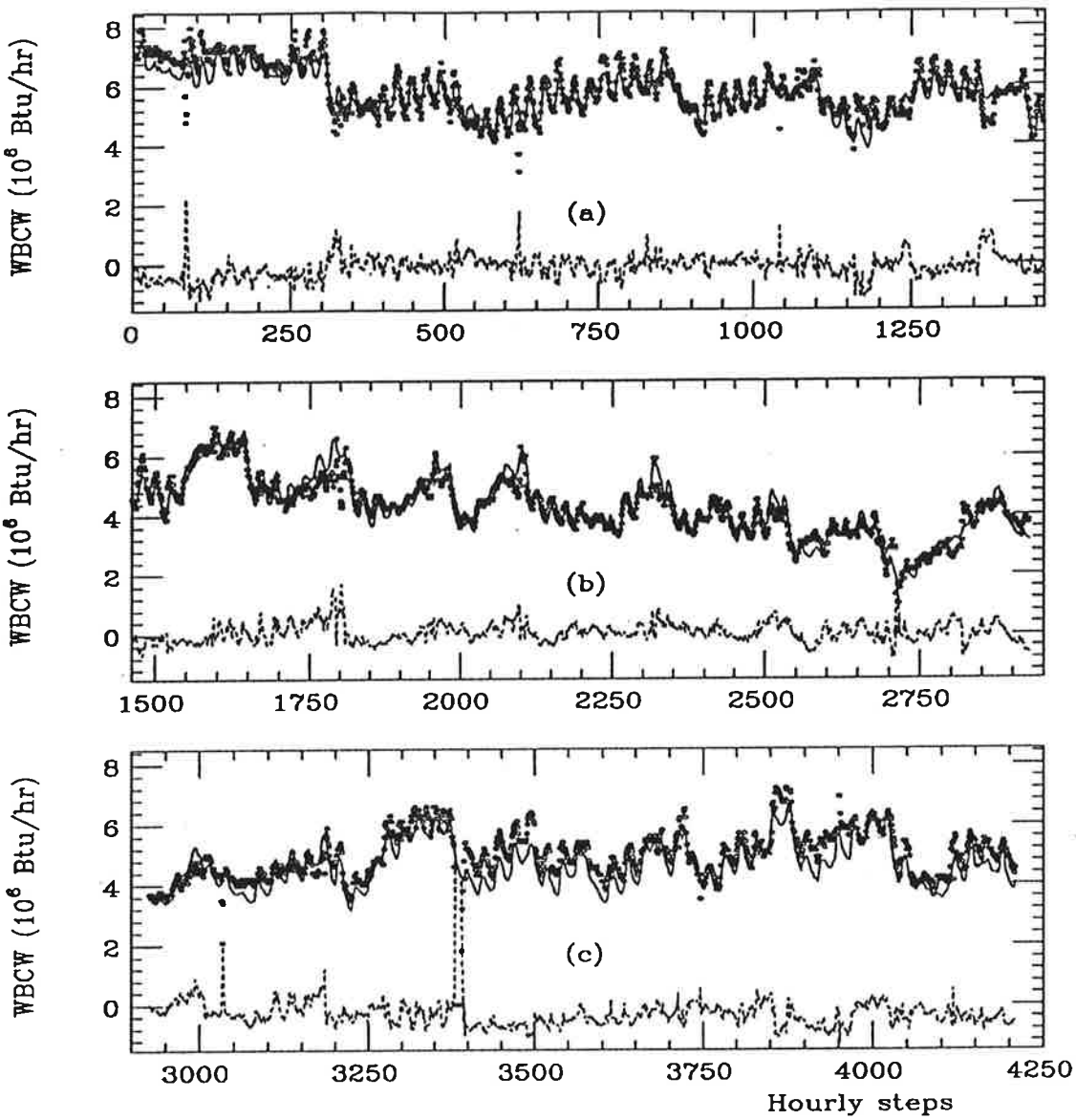

Figure 5 *Predicted WBCW consumption plotted together with the data and the residues. The notations are the same as in Figure 4.*

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 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, these networks have almost reached their limits for learning the tasks, while for WBHW there may still be room for some improvement. However, 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.

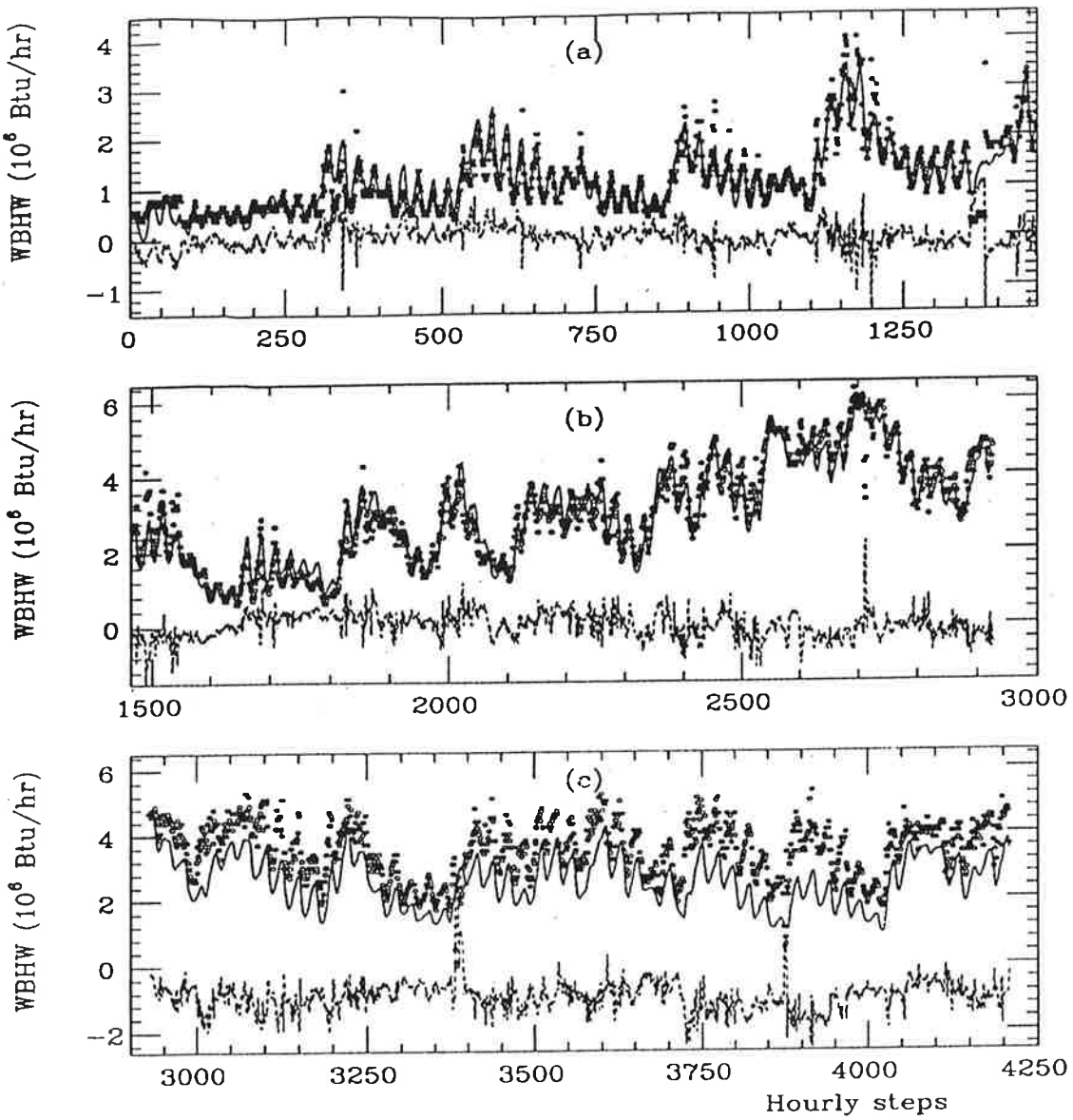

Figure 6 *Predicted WBHW consumption plotted together with the data and the residues. The notations are the same as in Figure 4.*

DATA SET B

General Properties of Data

This set consists of 3,344 solar radiation measurements during a nine-month period (August-May). The task is to predict $y(\vec{x})$, where

y = true beam insolation,
x_1 = decimal date (Julian day + hour/24),
x_2 = horizontal solar flux (W/m^2),
x_3 = southeast solar flux (W/m^2),
x_4 = south solar flux (W/m^2), and
x_5 = southwest 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 peaked 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 use the "shootout" labels in our description and calculations.

Variable Dependencies

The solar flux has a stronger dependency on the hour than on the (Julian) day; therefore, the decimal date column is split into day and hour columns. Results from applying the δ-test (Pi and Peterson 1993) on this data set are shown

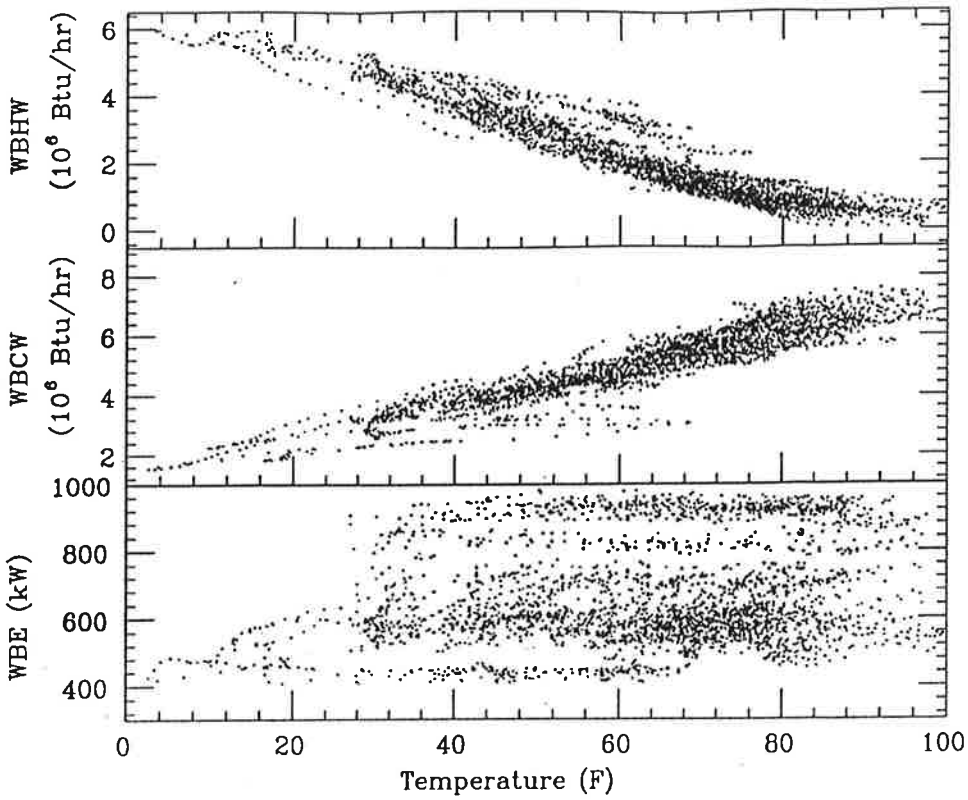

Figure 7 Predicted WBE, WBCW, and WBHW consumption vs. the dry-bulb temperature.

in Figure 8. The y variable has dependencies on the four angled measures of solar flux as well as on the hour, as seen in Figure 8a. However, if the variables are reordered so that the hour is entered last, as shown in Figure 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 Figure 8a or 8c, nearly saturating to 1, which suggests a high deterministic relationship. This is in contrast to data set A (Figure 3), where the sum of the indices yields 0.92 and a relatively large noise level is found. From Figure 8b (or Figure 8d), we read $\varepsilon_0 \sim 0.03$, and the noise level $\sigma_r \approx \varepsilon_0/3.6 = 0.0083$. Multiplying by the standard deviation/mean ratio (316/381) of the y data set, the noise would correspond to an error measure of CV ~ 0.007.

ANN Method

Given that there is 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 two hidden layers, a single-output unit, and seven inputs, where the inputs are

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

No time-lagged inputs were used 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 1,300. 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 package described in Lönnblad et al. (1992). A Langevin updating scheme (see, for example, Rögnvaldsson [1994]) was used, as it often performs better than backpropagation.

Results

In Figure 9, a scatter plot is shown of true and predicted beam insolation for the training data, with a variation

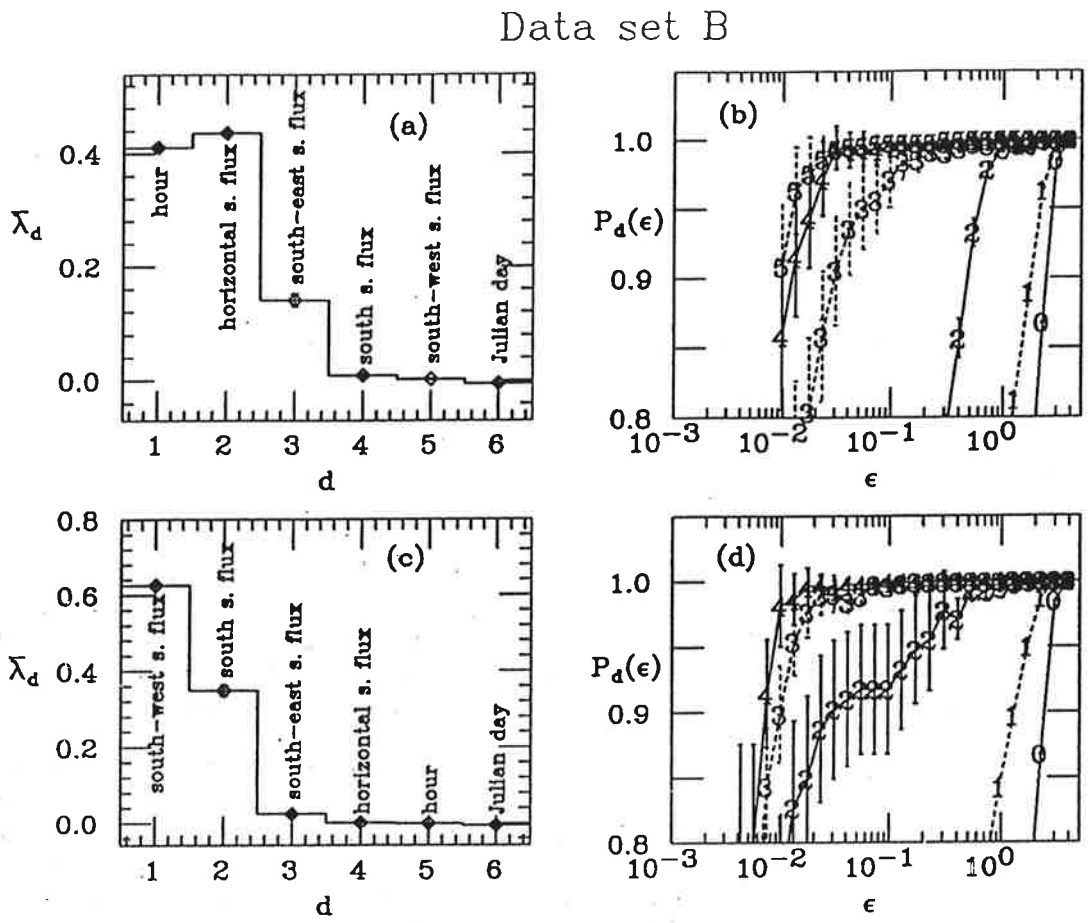

Figure 8 : *(a) Dependency indices* $\bar{\lambda}$ *for* y *(true beam insolation) on the independent variables. (b) The probability* $P_d(\epsilon)$ *as a function of* ϵ *for the same set of variables as in (a);* d *is marked on the curves. (c) and (d) The same as in (a) and (b) but for a different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

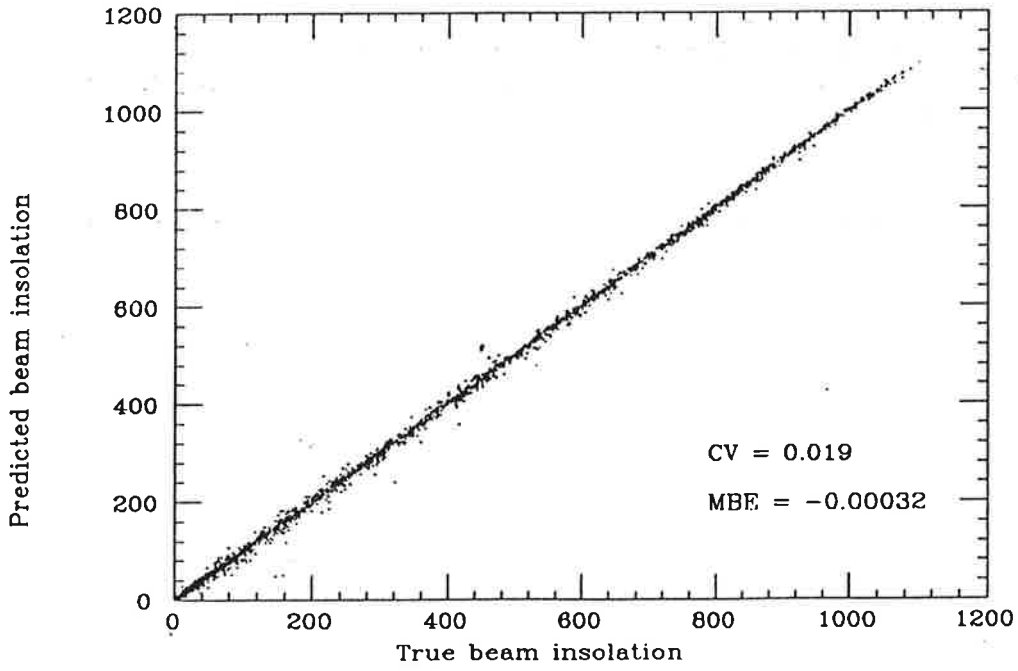

Figure 9 *Scatter plot showing predicted beam insolation vs. true beam insolation for the training data of set B.*

coefficient of CV = 0.019 and a mean bias error of 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.

Validation of the Result

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

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

where the noise level $\gamma \sim \text{stddev}(\varepsilon)$. 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. Hence, we conclude that our prediction is good enough.

SUMMARY

We have approached the two data sets provided in "The Great Energy Predictor Shootout—The First Building Data Analysis and Prediction Competition" (Kreider and Haberl 1994) with an almost "black-box" procedure based on

- the δ-test for establishing dependencies and gauging network performance and
- a multilayer perceptron (MLP) (Rumelhart and McClelland 1986) for modeling historic data.

When selecting the appropriate ANN architecture and learning algorithm, the choices are a standard MLP and/or a recurrent network (Williams and Zipser 1989). 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, 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.

ACKNOWLEDGMENTS

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

REFERENCES

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

Kreider, J.F., and J.S. Haberl. 1994. The great energy predictor shootout—The first building data analysis and prediction competition. *ASHRAE Transactions* 100(2).

Lönnblad, L., C. Peterson, and T. Rögnvaldsson. 1992. Pattern recognition in high energy physics with artificial neural networks—JETNET 2.0. *Computer Physics Communications* 70: 167.

Pi, H., and C. Peterson. 1993. Finding the embedding dimension and variable dependencies in time series. *Neural Computation.* 6:509.

Rögnvaldsson, T. 1994. On Langevin updating in multilayer perceptrons. *Neural Computation.*

Rumelhart, D.E., and J.L. McClelland, eds. 1986. *Parallel distributed processing: Explorations in the microstructure of cognition*, vol. 1. Cambridge, MA: MIT Press.

Williams, R.J., and D. Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation* 1: 270.