Proc. Estonian Acad. Sci. Eng., 2002, **8**, 4, 223–247

Feed-forward neural networks for smooth operation of the high-voltage power transmission network

Taivo Kangilaski

IT Division, Eesti Energia Ltd, Laki 24, 12915 Tallinn, Estonia; Taivo.Kangilaski@energia.ee

Received 13 June 2001, in revised form 21 June 2002

Abstract. The article describes algorithms based on feed-forward neural networks for short-term forecast of high-voltage power consumption and fault prediction, implemented in the software "Event Navigator" of Eesti Energia AS.

Key words: high-voltage power transmission, fault prediction, short-term forecast.

1. INTRODUCTION

Owing to its nature, electricity has to be produced at the time it is being consumed. In the operational mode, planning for a definite time period plays an important role in the control of a power system. Various software is available for calculations of the regimes of power systems. Among them, the Power System Simulator for Engineering (PSS/E) [1,2] seems to be the most popular. The Supervisory Control And Data Acquisition solftware SCADA has also been equipped with regime calculation tools (e.g., Harris SCADA product XA/21 [3], ABB SCADA product S.P.I.D.E.R [4], IVO SCADA system [5], and Siemens SCADA product SINAUT-SPECTRUM [6]).

However, in order to calculate regimes for the time ahead, one needs consumption projections for that time. As a result, electric utilities and system operators face short-term forecasting problems, involving loads during the next hour, the next day, and the next week. In terms of regime, it is also important to know what should be the quantity of facilities in reserve to guarantee smooth operation of the transmission network. This problem is related to fault prediction. The aim of this paper is to consider both of these problems.

The paper is organized as follows. Firstly, the initial data are defined. Secondly, existing models that may solve the problem are analysed. Further, new models for fault projection and short-term forecast of the power consumption are presented. In case of fault prediction, four different learning rules and four initial data combinations are analysed to find out convenient inputs and learning rules. In a short-term forecast, we propose the initialization parameters for multi-layer feed-forward neural networks (FNN). These neural networks are commonly referred to as multi-layer perceptrons that represent a generalization of the single-layer perceptrons.

The study is based on the facilities of 110 kV and more and on the Power Plant facilities. We discuss these issues in the context of Eesti Energia's (EE) management system Event Navigator (ENav) that is web-based and made in the Oracle environment.

2. DATA AND CONSTRAINTS

The input data, used for consumption projection, comprised statistical consumption data, environmental air temperature, and light intensity. As input data, commercial metering data from the CENTRAPULS system was used. In case no metering data was received from CENTRAPULS, the telemetering data from SCADA of the Control Centre was used, which originated from the same metering devices, but were integrated.

Hourly data as well as the collection of breakdown data of the facilities were archived from the beginning of 1993, including all planned and non-planned work and extraordinary repairs.

In data generation, when making a consumption projection for the day $x+1$ on day x, data up to the day $x-1$ are available. Our maximum projection term was 8 days. For the first four days, an average error of $\pm 3\%$ suggests an accurate projection and for the last four days an average error of $\pm 6\%$ indicates a high accuracy of the projection.

Failure prevention implies a significant saving of financial resources. Fault predictions are instrumental when optimal downtime plans and investment plans are drawn up. To predict faults, we should use all the information available at EE. The existing information can be classified into four groups:

1) downtime historical data, including facility name, downtime start and stop times, and downtime type;

2) facility passport data and restrictions;

3) human expert knowledge, including behaviour of the power system;

4) data of the statistical analysis.

Only historical data about downtime was used.

3. CLASSIFICATION OF THE MODELS

For the short-term projection of consumption five broad classes of discrete-time models were considered:

1) time-series and transfer-function models;

2) models based on trigonometric functions;

3) state–space models;

4) models based on orthogonal transformation;

5) hierarchical models including the Group Method of Data Handling (GMDH) and neural networks.

These models are similar in terms of their ability to accommodate a certain degree of uncertainty, and they can adapt to the dynamics of the time-varying process [7].

Time-series and transfer-function models. Time series is a sequence of observations of a process. It may or may not have a periodic component associated with it. Time series may be represented by autoregressive (AR) [8], integrated AR (IAR) [9], AR moving average (ARMA) [10,11], and AR with integrated moving average (ARIMA) [12] models [13–15], which are based on polynomial operators in discrete time. If a time series shows structural features, like a trend and periodicity, structural components may be separately modelled. Transfer-function models are natural extensions of time-series models. It is expected that the process in question is subjected to certain inputs, that influence the output of the process. Transfer-function models have additional terms for exogenous inputs, as ARMAX (that is ARMA with exogenous inputs) and ARIMAX [9,15].

Models based on trigonometric functions. Processes with regular or irregular periodicity can be analysed in the frequency domain and can be modelled in terms of components, expressed as trigonometric functions [16–20]. Besides modelling, frequency-domain characterization provides useful information in the design of filters as well as in assessing the appropriate rate of sampling of continuous-time signals for discrete-time modelling.

State–space models. These models have the unique feature that along with variables that are known or can be measured, the variables that are internal to the process and cannot be measured are also incorporated into the model [21–28]. This is why a state–space model is also called an internal model, whereas a model based on measurable variables is called an external model. Any transfer-function or time-series model can have a state–space representation. State–space models can model processes with or without periodicity.

Models based on orthogonal transformation. These are Singular Value Decomposition (SVD) based models [29]. SVD [30–32] is an optimal orthogonal decomposition, applied in rank determination, matrix inversion, as well as in modelling, prediction, filtering, and information compression of data sequences [33]. In numerical terms, SVD is extremely robust, and singular values in SVD can be computed with greater accuracy than eigenvalues [34]. Models based on SVD are particularly suitable for time series that are nearly periodic or quasi-periodic [35–39]. The principle of modelling for the nearly periodic series is that the consecutive periods are aligned into consecutive matrix rows, which are SV-decomposed. The decomposed components are now modelled, typically as time series.

A quasi-periodic series [36,37] can be decomposed into components which are individually nearly periodic and hence can be modelled as described above.

The two attractive features of the SVD-based modelling are that a prediction of one or multiple periods ahead may be produced, and SVD, which is extremely robust numerically, ascribes robustness to the model.

Hierarchical models and neural networks. These models are primarily suitable for time-series and input-output processes with non-linearity; quasi-periodic processes can also be modelled. Three types of models were considered: models based on GMDH [38–42], neural network models [43–51], and models based on singular-value decomposition with or without non-linear transformation [36,37,42]. All of these models have hierarchical stages or layers where each stage incorporates simple non-linear elements. Since most processes contain a certain degree of non-linearity, these models are applicable for non-linear as well as nearly-linear (and nearly-periodic) processes.

In all the cases, efforts are made to develop parsimonious models, i.e., only essential variables are to be included and the order of the model is kept as low as possible. The degree of accuracy of the data should be duly considered. In the case of applications, the model needs to be protected against irrelevant information influencing it [52–65].

4. SELECTION OF MODELS

In our data analysis, for consumption projection we compared consumption data of the days $K-1$ and $K+1$ (Fig. 1) and consumption data dependence on the environmental air temperature (Fig. 2) and on the light intensity.

Analysing Figs. 1 and 2, one can see that the relationships are non-linear. Thus our task was reduced to the approximation of a function with multiple variables. Since initial data was non-linear and the model was designed to approximate both nation-wide and regional power consumption, with no specific

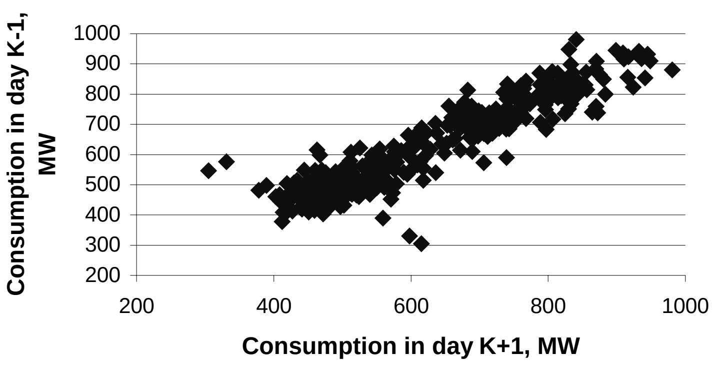


Fig. 1. Day $K-1$ and day $K+1$ consumption data.

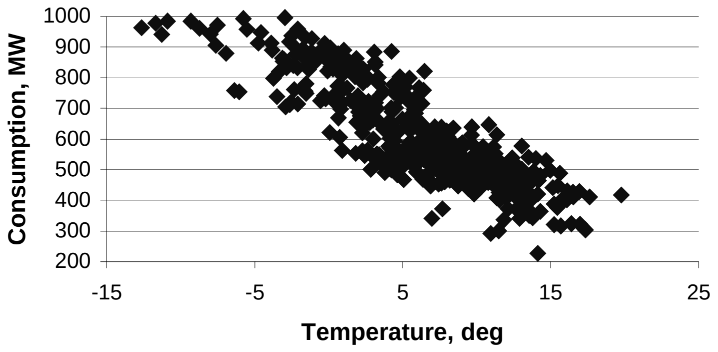


Fig. 2. Consumption dependence on the air temperature.

information available at the time of the network development, a neural network (NN) was chosen [45]. The other reason for selecting NN was that a single hidden layer is sufficient to approximate uniformly any continuous function supported in a unit hypercube [66]. Two other papers on multi-layer perceptrons as universal approximators are [67,68].

The feed-forward neural network was selected since it requires less computation resources [69]. It is also proved that neural network is a very powerful abstraction to learn patterns [42,43,47,48]. In our case it means that we may consider facility downtimes also as a pattern and use the neural network for fault prediction [70–80].

5. FEED-FORWARD NEURAL NETWORK

Artificial neural networks, used in forecasting, are flexible non-linear models that can approximate a wide range of data-generating processes.

In a general form, for a single-variable forecasting problem, an artificial neural network is $Y = F(X, W) + u$. Most functions in this general form, including all functions that are normally used in forecasting, do not qualify as neural networks. Although neural networks can take many forms, the most frequently used form is very specific (two-layered perceptron) and can be written as

$$Y(n) = w_{1,0}^{(2)}(n) + \sum_{j=1}^{N} (w_{1,j}^{(2)}(n) \cdot y_j^{(1)}(n)) + u_0(n), \tag{1}$$

where

$$y_j^{(1)}(n) = \frac{1}{1 - \exp[-v_j^{(1)}(n)]} = \frac{1}{1 - \exp[\,-(w_{j,0}^{(1)}(n) + \sum_{i=1}^{K} w_{j,i}^{(1)}(n) x_i(n))\,]} \tag{2}$$

and

$$u_0(n) = \sum_{l=1}^{M} b_l(n) u_l(n). \tag{3}$$

Here $Y(n)$ represents neural network output, n denotes a time variable, $w_{i,j}^{k}$ is the weight between the ith neuron in layer k and jth neuron in layer $k-1$ (if $j=0,$ then it refers to the threshold), $v_i^{(k)}$ is the net activity of the ith neuron in layer k and $y_i^{(k)}$ is the output of the ith neuron in layer $k,$ N is the number of neurons in the hidden layer, M is the number of external inputs and K is the number of inputs, u_0 is the weighted external input, u_l is external input, and b_l is its weight.

The function in parentheses in Eq. (1) is repeated N times exactly in the same algebraic form. In network jargon, this equation is called a single-output feed-forward neural network with a single hidden layer, with N nodes and logistic activation functions in the hidden layer, and with a linear activation function in the output layer.

This specification is non-linear in the variables and parameters. It is flexible in that it allows for a wide variety of non-linearity and interactions among the explanatory variables, and the repetitive specification is the cornerstone of the flexibility.

It is easy to understand this non-linear function by considering a simple example. Let us take the number of input variables $K=3$ and the number of nodes $N=2$ (Fig. 3). The network function takes the following form:

$$\begin{aligned} Y(n) = w_{1,0}^{(2)}(n) + w_{1,1}^{(2)}(n) \cdot & \frac{1}{1+\exp[-(w_{1,0}^{(1)}(n) + w_{1,1}^{(1)}(n)x_1(n) + w_{1,2}^{(1)}(n)x_2(n) + w_{1,3}^{(1)}(n)x_3(n))]} \\ + w_{1,2}^{(2)}(n) \cdot & \frac{1}{1+\exp[-(w_{2,0}^{(1)}(n) + w_{2,1}^{(1)}(n)x_1(n) + w_{2,2}^{(1)}(n)x_2(n) + w_{2,3}^{(1)}(n)x_3(n))]} + u_0(n). \end{aligned} \tag{4}$$

Given the values of the explanatory variables and a set of parameter w values, one can easily compute the predicted values and residuals for each observation. The estimation problem is to find parameters that make the residuals as small as possible.

Although function (4) is clearly non-linear relative to x and most of the parameters, it has linear components. To show this, let us rewrite the network function as follows:

$$Y(n) = w_{1,0}^{(2)}(n) + w_{1,1}^{(2)}(n) \cdot y_1^{(1)}(n) + w_{1,2}^{(2)}(n) \cdot y_2^{(1)}(n) + u_0(n). \tag{5}$$

In network terms, each $y^{(1)}$ represents a node in the hidden layer. For the particular specification used here, the output function, shown in Eq. (5), is linear in these values. The output function can be considered non-linear in $y^{(1)}$, but for most forecasting problems with continuous dependent variables there will be no advantage of this additional non-linearity. A second linear component is found in

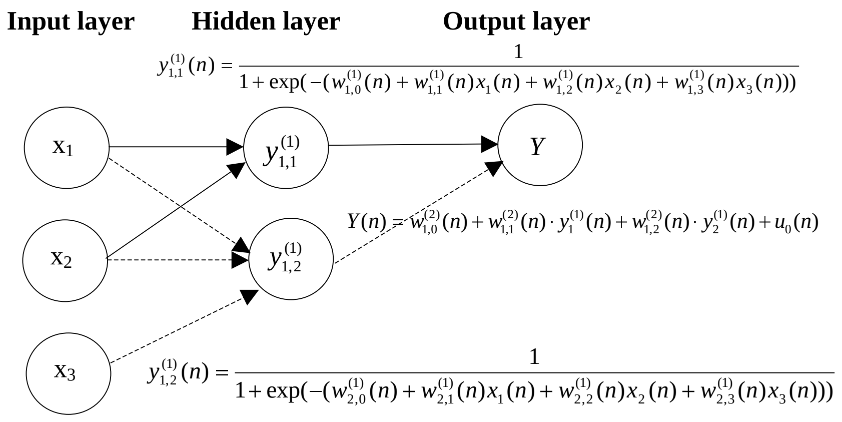

Input layer
Hidden layer
Output layer
$y_{1,1}^{(1)}(n) = \frac{1}{1+\exp(-(w_{1,0}^{(1)}(n)+w_{1,1}^{(1)}(n)x_1(n)+w_{1,2}^{(1)}(n)x_2(n)+w_{1,3}^{(1)}(n)x_3(n)))}$
x_1
$y_{1,1}^{(1)}$
Y
$Y(n) = w_{1,0}^{(2)}(n)+w_{1,1}^{(2)}(n)\cdot y_1^{(1)}(n)+w_{1,2}^{(2)}(n)\cdot y_2^{(1)}(n)+u_0(n)$
x_2
$y_{1,2}^{(1)}$
x_3
$y_{1,2}^{(1)}(n) = \frac{1}{1+\exp(-(w_{2,0}^{(1)}(n)+w_{2,1}^{(1)}(n)x_1(n)+w_{2,2}^{(1)}(n)x_2(n)+w_{2,3}^{(1)}(n)x_3(n)))}$

Fig. 3. Network diagram with three explanatory variables and two nodes.

the denominator of $y^{(1)}$. Specifically, the exponent is a linear weighted sum of the input variables which can be written as

$$v_1^{(1)}(n) = w_{1,0}^{(1)}(n) + w_{1,1}^{(1)}(n)\cdot x_1(n) + w_{1,2}^{(1)}(n)\cdot x_2(n) + w_{1,3}^{(1)}(n)\cdot x_3(n), \tag{6}$$

After some transformations we obtain

$$y_1^{(1)}(n) = \frac{1}{1+\exp[-v_1^{(1)}(n)]} = \frac{\exp[v_1^{(1)}(n)]}{1+\exp[v_1^{(1)}(n)]}. \tag{7}$$

From that, we can recognize a binary logic [81]. The linear weighted sum (v) is the power on e. If v is a large negative number, then $y^{(1)}$ is close to zero. If v is 0, then $y^{(1)}$ is 0.5. And if v is a large positive number, then $y^{(1)}$ is close to 1. In between, it traces out an S-shaped function. This also means that there is an S-shaped relationship between each node value (v) and each explanatory variable (x) in that node. This S curve may be positively or negatively sloped, depending on the sign of the slope coefficient of x ($(w_{j,i}^{(0)}$ in Eq. (2)). Further, the specification is automatically interactive.

Rewriting the exponential, we obtain:

$$\exp[w_{1,0}^{(1)}(n) + w_{1,1}^{(1)}(n)\cdot x_1(n) + w_{1,2}^{(1)}(n)\cdot x_2(n) + w_{1,3}^{(1)}(n)\cdot x_3(n)]$$
$$= \exp[w_{1,0}^{(1)}(n)]\cdot \exp[w_{1,1}^{(1)}(n)\cdot x_1(n)]\cdot \exp[w_{1,2}^{(1)}(n)\cdot x_2(n)]\cdot \exp[w_{1,3}^{(1)}(n)\cdot x_3(n)]. \tag{8}$$

Now each variable x interacts with all other variables that do not have zero slopes in the node. Multiplicative interactions related to the underlying process give evidence of the efficiency of the specification. It is true that each x appears several times and, in the case of Eq. (1), they appear in exactly the same algebraic form. At first glance, econometricians will not be comfortable with that idea. It looks like an extreme form of multi-collinearity. In network language, the repetitive specification is called parallelism or massive parallelism, and it is one of the strengths of this approach. It could raise some serious issues by parameter estimation [45]. With several nodes in the hidden layer, the specification allows for a variety of non-linearities and for a range of variable interactions. For example, two logistic curves, one positively and the other negatively sloped, can be combined to give a U-shaped response over the relevant range of x. Given this flexibility, the estimation problem is to find a set of specific non-linearities and interactions that are useful for explaining the history and for forecasting. In terms of neural networks, Eqs. (1) to (3) describe a feed-forward neural network with a single output which has one hidden layer with one or more nodes, uses logistic activation functions in each node of the hidden layer, and uses a linear activation function in the output layer.

Figure 3 shows a neural network described by the function (1), where the explanatory variables x enter the input layer. The logic transforms appear in the hidden layer, and the result Y appears in the output layer. The idea is that the inputs feed the nodes in the hidden layer, and there is no feedback. Further, the nodes do not feed sideways into each other. Instead, they feed onwards to the output layer. There is no feedback, delayed or otherwise, from the output layer to the hidden nodes. The absence of feedbacks or node-level interactions makes it a feed-forward system.

Activation function refers to the S-shaped nature of the function in each node and to the fact that boundary values (0 and 1) can be interpreted as “on” and “off”. The term “activation function” is taken from the neural sciences and it is related to the requirement that a signal must reach a certain level before a neuron fires to the next level. It is an adequate description of the forecasting. For most forecasting problems, if flexibility is allowed (more than two nodes), some of the nodes will end up specializing and will activate (take on a value close to 1.0) under specific conditions and take on a value close to zero otherwise.

For forecasting purposes, the functions of the hidden layer must be logistic functions. Any other S-shaped function can be used, such as an arctan or a cumulative normal, without a significant change in the model behaviour. It is important to use a smooth differentiable function that is easy to work with for estimation purposes.

Further, it is not necessary to use S-shaped curves at all. For example, bell-shaped functions, like the derivative of a logistic curve, may be used in some nodes. However, moving away from S-shaped curves, the term “activation” becomes less descriptive of functional performance, since the alternative functions would no longer range between zero at one extreme and one at the other. As a result, activation functions of the hidden node are sometimes called neuron-transfer functions [44].

Finally, additional non-linearity may be introduced in the output layer. In fact, for dependent variables that have discrete outcomes, such as a binary variable, a logistic activation function in the output layer would probably be desirable. But for most problems with continuous outcomes, there is no real gain from a further non-linearity at this level.

In the literature on neural networks, the process of parameter estimation is called training. The goal of the training process is to find network parameters that make the model errors small. The estimation process is more complicated than in a regression model because the model is non-linear and because the objective function is relatively complicated. Parameter estimation is a simple process for most common statistical problems such as finding the solution to a non-linear regression model. However, because of the parallelism that reflects the inclusion of multiple nodes in the hidden layer, it can be shown that the least squares objective function for a neural network is extremely complex with a huge number of local optima.

Owing to this complexity it is necessary to explore a wide region of the parameter space to find a relatively good solution. The rule for selecting the final model parameters is based on an average of in-sample and out-of-sample error statistics. Estimation from each random starting point is based on a subset of the sample data. Once a solution is found, these parameters are used to test the power of the estimated parameters, based on the observations withheld from the estimation process. Usually, solutions that perform well in the sample, perform well also out of the sample. Particularly, with a large number of nodes, some solutions will be more specialized in specific cases in the sample, and others will be more stable and more useful out of the sample. There exist large numbers of non-linear specifications that provide similar performance.

As new data become available, it is natural to re-estimate parameters with the extended sample period. In literature, the update process for incorporating new data is called learning.

With linear least squares models, updating the parameters with additional data is a full re-estimation of the model and to the new data typically the same weight as to the earlier data is given. For non-linear least squares, the re-estimation process can begin with the same set of initial guesses that was used to start the original estimation process, or it can begin with the solution obtained from that estimation. In both cases, for problems with a well-behaved object function, the final solution will be a single set of parameters that corresponds to the global optimum based on the expanded data set.

For neural networks that are known to have a large number of local optima, the situation is slightly different. In this case, it seems natural to start with the parameters from the training process and to re-optimize the solution with the new data included. However, starting with the training solution implies starting with the specific set of non-linearities and variable interactions represented by that solution. From this starting point, re-estimation with the expanded data typically leads to minor changes in the estimated parameters. This implies that we are staying at the same local optimum at which we started, and that the location of this solution does not move much because of the new data. In this sense, the role of the new data is smaller than that of the earlier data. Actually, the functional form is determined in the training process which looks at many solutions and selects one, and the new data is used only to refine the parameters of that functional form. To give the new data the same role as has the earlier data, it would be necessary to repeat the entire training process with the expanded data set.

6. MODEL FOR FAULT PREDICTION

As described in Section 4, the neural network is a very powerful abstraction for pattern recognition. We can consider facility downtimes also as a pattern (Fig. 4).

For fault prediction, the developed network uses planned and non-planned facility downtimes to predict faults. Planned downtimes are stops arising from annual or monthly plans. Non-planned downtimes are stops that have not been planned in annual or monthly plans, but are not faults.

Thus, the FNN's objective is to identify a pattern, which is a standard identification problem. To describe the network, initial data is analysed, neural network inputs are defined as well as layers, activation functions and outputs, and the wanted detection probability. The distance forward from the fault is marked with the "+" sign and the distance backward from the fault with "–". In data analysis, the facilities are grouped by the type. We found that on average the downtime count by type is 20 in a quarter. Thus 20 inputs were selected to our FNN to create the input pattern. The idea of selecting the layers is that each next layer should simplify the pattern for grouping the data. For example, in a transformer, the planned work may be a major overhaul (once in 12 years), current maintenance (once in 3 years), transmission checking (once in 2 years), etc. Thus the first layer combines transmission checking and current maintenance, the second layer – major overhaul and combined transmission checking and current maintenance, etc. The neuron selection for hidden layers is based on the introduction of three neurons per each input: "+", "–", and "0". Thus there are 60 neurons in a layer.

To select the activation functions, we should take into account that they must be monotone and bounded and if they are continuous, they must satisfy the Lipschitz condition

$$\| f(v_1) - f(v_2) \| \leq C \| v_1 - v_2 \|, \tag{9}$$

where $f()$ is the activation function, v is net activity, and C is the Lipschitz coefficient. The activation functions are described in Table 1, where $\pm h$ denotes limits of the activation function $f()$, c is the coefficient which determines the rising angle of $f()$, and a is a coefficient that determines the behaviour of FNN when v is about zero.

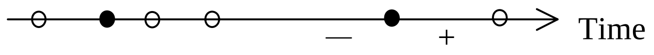


Fig. 4. Illustration of facility stops: ○ planned or non-planned facility downtime (NF), ● emergency downtime fault (FL).

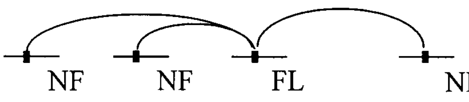
Table 1. Activation functions

Signum function	Identity function	Delta function
1, when a fault exists -1, when no fault exists	1, when a fault exists 0, when no fault exists	h, when $v \geq a$ cx, when $\lvert v \rvert < a$ $-h$, when $v \leq -a$

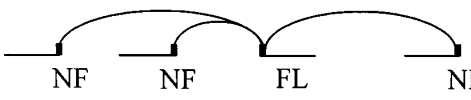
The signum and identity functions were used to predict only fault existence. The delta function was used to obtain the probability of fault existence.

By constructing the FNN, we experimented with input data as follows (E1 to E4 indicate the experiments):

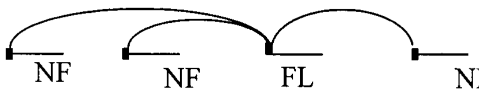
E1. The middle points in the duration of the NF downtimes were found and the distance to the FL start was calculated.



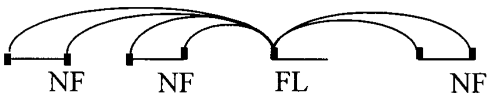
E2. The finishing points of the NF downtimes were found and the distance to the FL start was calculated.



E3. The starting points of the NF downtimes were found and the distance to the fault start was calculated.



E4. The NF downtime beginning and duration were found and the distance to the FL start was calculated.



To find the most suitable learning method, we investigated and built the Hebbian, the Perceptron, the Delta and the Widrow–Hoff Learning Rule in ENav. A general learning rule is based on [43]. The weight vector w_i increases in proportion to the product of the input x and the learning signal r_i. The learning signal is, in general, a function of w_i, x, and sometimes of the reference or desired signal d_i. Hence,

$$r_i = r_i(w_i, x, d_i). \tag{10}$$

The increment of the weight vector is expressed as

$$\Delta w_i(t) = \eta\, r_i\,[w_i(t), x(t), d_i(t)]\,x(t), \tag{11}$$

where η is a positive number called the learning constant that determines the rate of learning. The weight vector adopted at the time t becomes at the next instant (or learning step)

$$w_i(t+1) = w_i(t) + \Delta w_i(t). \tag{12}$$

The Hebbian Learning Rule. This rule requires the weight initialization at small random values around $w_i = 0$ prior learning. The Hebbian Learning Rule represents only feed-forward, unsupervised learning [82]. The rule states that if the cross product of output and input or the correlation term $f_i(s_i)x_j$ is positive, then it results in an increase in the weight w_{ij}; otherwise the weight decreases:

$$r_i \equiv f(w_i^T x) = f_i(s_i), \tag{13}$$

$$\Delta w_{ij} = \eta\, f_i(s_i)\, x_j. \tag{14}$$

Thus, the single weight w_{ij} is adapted, using

$$\Delta w_i = \eta\, f_i(s_i)\, x_j. \tag{15}$$

The Perceptron Learning Rule. In this rule, the learning signal is the difference between the desired and the actual neuron response

$$r_i = d_i - y_i. \tag{16}$$

As mentioned, the zero mean threshold activation function is used and therefore

$$y_i = \operatorname{sgn}(s_i) = \operatorname{sgn}(w_i^T x). \tag{17}$$

In this method, weight adjustments are obtained as

$$\Delta w_i = \eta[d_i - \operatorname{sgn}(w_i^T x)]\,x, \tag{18}$$

$$\Delta w_{ij} = \eta[d_i - \operatorname{sgn}(w_i^T x)]\,x_j. \tag{19}$$

Since the neuron response is only binary, we have

$$\Delta w_{ij} = \pm 2\,\eta\, x. \tag{20}$$

The plus sign is used if $d_i = 1$ and $\operatorname{sgn}(w_i^T x) = -1$.

Using the learning input vectors $x(1)$ to $x(n)$, we calculated constants $\beta = \max_{i=1\ldots n} \| x(i) \|^2$ and $a = \min_{i=1\ldots n} | \omega_0^T x(i) |$, where ω_0 is the solution weight vector in our learning process. The Rosenblat theorem [83,84] is applicable here. It says that if a solution exists in FNN, then in step n_0 the weight ω_0 is attainable:

$$n_0 = \leq \beta \,|\, \omega_0^T \,|\, a^{-2}. \tag{21}$$

The energy E is

$$E = \frac{1}{N} \sum_{n=1}^{N} \left| d(n) - y(n) \right|^2. \tag{22}$$

Since

$$\left| d(n) - y(n) \right|^2 = \begin{cases} 0, \text{ when catch,} \\ 4, \text{ when detects,} \end{cases} \tag{23}$$

then

$$E = \frac{1}{N} 4L, \tag{24}$$

where L is the number of statements in Eq. (23) that are not zero. Therefore E is fault probability (L/N) multiplied by four.

In Eq. (23) the word “catch” is used for describing the following situations:

1) the event is “caught” when it occurs as projected,

2) the event is “caught” when it did not occur as projected.

The word “detect” is used if

1) the event occurred, but not as projected, or

2) the event did not occur, although it was projected.

The Delta Learning Rule. This rule is applicable only if an activation function is differentiable (in our case the function is monotone and continuos) and in the supervised mode [43,84]. The learning signal r_i for this rule is called “delta”, defined as

$$r_i = [d_i - f_i(w_i^T x)] f_i'(w_i^T x). \tag{25}$$

This learning rule can be readily derived from the condition of least squared error between y_i and d_i. The squared error, calculating the gradient vector with respect to w_i, is defined as

$$E = (d_i - y_i)^2 / 2 = (d_i - f_i(w_i^T x))^2 / 2. \tag{26}$$

We obtain the error gradient as

$$\nabla \boldsymbol{E} = -(d_i - y_i) f_i'(w_i^T x) x. \tag{27}$$

Since the minimization of the error requires that the weight changes are in the negative gradient direction, we take

$$\Delta w_i = \eta \nabla \boldsymbol{E} = \eta (d_i - y_i) f_i'(s_i) x. \tag{28}$$

In our case, $y_i' = f_i'(w_i^T x)x$. It means that if $|w_i^T x| < a$, then $y_i' = c$, and if $|w_i^T x| \geq a$, then $y_i = 0$ (Table 1). Thus

$$E_w' = \begin{cases} (d_i - y_i) cx, & \text{if } |w_i^T x| < a, \\ 0, & \text{if } |w_i^T x| \geq a, \end{cases} \tag{29}$$

$$\Delta w = \begin{cases} \eta (d_i - y_i) cx, & \text{if } |w_i^T x| < a, \\ 0, & \text{if } |w_i^T x| \geq a. \end{cases} \tag{30}$$

The Widrow–Hoff Learning Rule. This rule in applicable to the supervised training of the neural network [43,84]. It is independent of the activation functions of neurons used since it minimizes the squared error between the desired output value d_i and the neuron's activation value s_i. The learning signal is defined as

$$r_i = d_i - s_i = d_i - w_i^T x. \tag{31}$$

Thus the weighting vector increment under this learning rule is

$$\Delta w_i = \eta (d_i - s_i) x. \tag{32}$$

This rule can be considered as a special case of delta learning rule if the activation function is simply an identity function. The speed of convergence and the convergence itself of the learning rule depends on the constant η. To make the learning algorithm more reliable and efficient, its adaptive version was proposed in [43]. In our case, the constant η is updated according to the rule

$$\eta(x) = \begin{cases} 1/(x^T x), & \text{if } x^T x \neq 0, \\ 0, & \text{if } x^T x = 0, \end{cases} \tag{33}$$

and the corresponding weight increment is

$$\Delta w_i = \begin{cases} \alpha \eta(x)(d_i - y_i), & \text{if } x^T x \neq 0, \\ 0, & \text{if } x^T x = 0, \end{cases} \tag{34}$$

where α is a constant reduction factor.

7. RESULTS OF FAULT PREDICTION

For teaching we have used data from early 1993 until 1997. Testing was conducted using data from 1998 and 1999. This chapter provides results for transformers, where we used FNN with 20 inputs, 60 neurons in the hidden layer, and one output neuron.

To compare different methods, we analysed the FNN energy (E) because it represents the learning error

$$E = \frac{1}{N}\sum_{i=1}^{N} e_i^2, \tag{35}$$

where e_i represents the difference between the desired output and the real FNN output (Fig. 5). Below, the E indices refer to experiments E1 to E4 described in the previous section.

Tables 2 to 5 show FNN energy convergence or learning results for different learning rules.

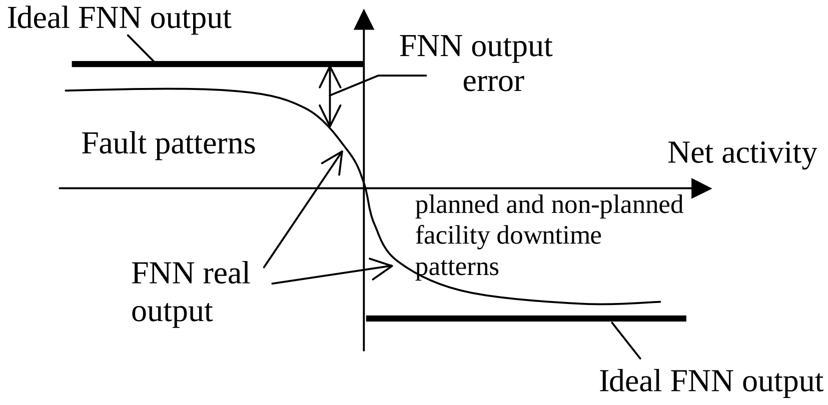


Fig. 5. FNN output and error.

Table 2. The Hebbian Learning Rule, α = 0.47, η = 0.28

Number of experiments	E_{E1}	E_{E2}	E_{E3}	E_{E4}
50	0.515	0.460	0.393	0.499
100	0.447	0.414	0.320	0.421
150	0.443	0.378	0.242	0.380
200	0.380	0.351	0.229	0.310
300	0.343	0.331	0.200	0.288

Table 3. The Perceptron Learning Rule (years 1998/1999)

Number of experiments	E_{E1}	E_{E2}	E_{E3}	E_{E4}
200	0.251	0.294	0.307	0.987

Table 4. The Delta Learning Rule, $\alpha = 0.45$, $\eta = 0.28$

Number of experiments	E_{E1}	E_{E2}	E_{E3}	E_{E4}
50	0.413	0.360	0.344	0.416
100	0.371	0.281	0.229	0.178
150	0.293	0.178	0.111	0.149
200	0.212	0.091	0.107	0.148
300	0.155	0.023	0.103	0.007
400	0.130	0.019	0.100	0.006

Table 5. The Widrow–Hoff Learning Rule, $\alpha = 0.8$

Number of experiments	E_{E1}	E_{E2}	E_{E3}	E_{E4}
50	0.317	0.311	0.400	0.287
100	0.218	0.303	0.220	0.274
150	0.175	0.212	0.153	0.246
200	0.126	0.210	0.065	0.250
300	0.120	0.008	0.100	0.195
400	0.102	0.006	0.070	0.102

From Tables 2 to 5 the following conclusions can be drawn.

1. The convergence speed of the Hebbian Learning Rule is slow.

2. In case of the Perceptron Learning Rule the value of η is not important as long as it is positive. After FNN has reached its minimum, it will not get more accurate (see the Rosenblat Theorem in Section 6). The speed of convergence was not examined.

3. The Delta as well as the Widrow–Hoff learning rules converge as $E[r^2(t)] \to 0$ if $t \to \infty$.

In the majority of cases, the Widrow-Hoff Learning Rule is most effective from the point of view of the convergence speed. An analysis of EE's overhaul data proves that the number of overhauls and of works performed on the facilities grows with an increase of the number of failures. Since 1992, failures account for 4–7% of all works performed. This is a rather stable figure. The analysis showed that failures depend on the type of the network element. There is no correlation between error percentages of different years. This means that depending of the type of the facility, only internal failure causes of facilities play a role. External conditions, such as weather, are not significant. Therefore, based on the downtime data alone, we can predict with high probability whether a certain facility will have a failure at the moment defined by the user. The fact that we can predict no failure is of major value. Depending on the type of the network element, our average detection percentage of emergency events is 71%. With regards to transformers, this figure amounts to 80%, which is the highest detection percentage among outdoor facilities. As for indoor facilities, our average failure detection percentage is 80 and for some facilities even 92. Yet for an individual network element within the facility, our average detection probability has been 57%.

8. A MODEL FOR SHORT-TERM PROJECTION OF THE HIGH-VOLTAGE POWER CONSUMPTION

For short-time projection of power consumption, we divided the day into 6-hour periods. Every hour was described by statistical consumption data, temperature, and light intensity – all in all 18 inputs. This exercise was based also on the assumption that one hidden layer is sufficient besides an input and an output layer to approximate a function with multiple variables, provided the network is completely connected and the number of neurons of the hidden layer exceeds considerably the number of inputs [45]. In the formation of the hidden layer, we used the linear combination of input elements of the input data of one hour as a logic, which gave 7 neurons per hour. Thus the number of neurons of the entire hidden layer was 42.

The output layer has only one neuron, and we took the consumption per hour as its output. We introduced the sigmoid activation function on both the hidden and output layers. Thus every projected hour was supplied with a network of its own, i.e., we developed separate networks of projected consumption for all the respective hours of the 8 days. We chose the back-propagated learning rule as a training rule, because we applied the least squares method to analyse the output error, and we used the sigmoid function as an activation function. Had we chosen identity as an activation function, we would have used the Widrow–Hoff Learning Rule. In practice, a trained neural network is retrained by the software when the difference between the actual consumption and projected consumption exceeds 2.5%.

The following back-propagation learning algorithm [43,84] was applied.

Initialization. All the synaptic weights and threshold levels of the network were set to small random numbers that were uniformly distributed.

Presentation of training examples. We presented the network with an epoch of training examples. For each example in the set ordered in some way, we performed the following sequence of forward and backward computations.

Forward computation. The training example in the epoch was denoted by $[\boldsymbol{x}(n), \boldsymbol{d}(n)]$, with the input vector $\boldsymbol{x}(n)$ applied to the input layer of sensory nodes and the desired response vector $\boldsymbol{d}(n)$ applied to the output layer of computation nodes. We computed the activation potentials and function signals of the network by proceeding forward through the network, layer by layer. The internal activity level $\upsilon_j^{(l)}(n)$ for the neuron j in layer l is

$$\upsilon_j^{(l)}(n) = \sum_{i=0}^{p} w_{ji}^{(l)}(n) y_i^{(l-1)}(n), \tag{36}$$

where $y_i^{(l-1)}(n)$ is the function signal of the neuron i in the previous layer $l-1$ on the nth iteration and $\upsilon_{ji}^{(l)}(n)$ is the synaptic weight of the neuron j in layer l that is fed from neuron i in layer $l-1$. For $i=0$, we have $y_0^{(l-1)}(n) = -1$ and $w_{j0}^{(l)} = \Theta_j^{(l)}(n)$, where $\Theta_j^{(l)}(n)$ is the threshold applied to the neuron j in layer l. Using the sigmoidal non-linearity, the function (output) signal of neuron j in layer l is

$$y_j^{(l)}(n) = [1 + \exp(-\upsilon_j^{(l)}(n))]^{-1}. \tag{37}$$

If neuron j is in the first hidden layer $(l = 1)$, set $y_j^{(0)}(n) = x_j(n)$, where $x_j(n)$ is the jth element of the input vector $\boldsymbol{x}(n)$. If neuron j is the output layer $(l = L)$, set $y_j^{(L)}(n) = o_j(n)$. Hence, we computed the error signal $e_j(n) = d_j(n) - o_j(n)$, where $d_j(n)$ is the jth element of the desired response vector $\boldsymbol{d}(n)$.

Backward computation. We computed the δ's (i.e., local gradients) of the network by proceeding backward, layer by layer, for neuron j in the output layer L:

$$\delta_j^{(L)}(n) = e_j^{(L)}(n) o_j(n)[1 - o_j(n)], \tag{38}$$

and for neuron j in the hidden layer l:

$$\delta_j^{(l)}(n) = y_j^{(l)}(n)[1 - y_j^{(l)}] \sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n). \tag{39}$$

Hence, we adjusted the synaptic weights of the network in layer l according to the generalized delta rule:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha[w_{ji}^{(l)}(n) - w_{ji}^{(l)}(n-1)] + \mu\delta_j^{(l)}(n) y_i^{l-1}(n), \tag{40}$$

where μ is the learning-rate parameter and α is the momentum constant.

Iteration. We iterated the computation by presenting new epochs of training examples to the network until the free parameters of the network were stabilized and the average squared error computed over the entire training set was at a minimum or at a sufficiently small value. By that we were guided by the following considerations. The order of presentation of training examples should be randomized from epoch to epoch. The momentum and the learning-rate parameters are typically adjusted (and usually decreased) as the number of training iterations increases.

For the tuning parameters, the leaning rate μ and the momentum constant α, an algorithm was developed which found the stable-energy case based on the existing data; the constants α and μ were determined for each separate network at every hour. Testing series was started with $\alpha = \mu = 1.0$, and software performed test series with $\alpha = \alpha - 0.001$; around this value the corresponding value of $\mu \pm 0.001$ was looked for. By testing, the lower limit of α was set at 0.

9. RESULTS FOR SHORT-TERM PROJECTION OF HIGH-VOLTAGE POWER CONSUMPTION

The method described above was applied to implement a feed-forward neural network that allowed a successful real-time projection of power consumption for a period of up to 8 days. The tuning parameters found for the networks are shown in Table 6 where the tuning parameters α and μ were applied to every hour's network, and energy was stable in the range $\alpha_1 < \alpha < \alpha_2$ and $\mu_1 < \mu < \mu_2$. Number N, indicating training samples per epoch, was increased proportionally to the projected day.

Based on the initialization parameters given in Table 6, the learning of each network for 600 training samples was performed. Table 7 shows the energies found.

Table 6. Tuning parameters of networks

Day	α_1	α_2	α	μ_1	μ_2	μ	N
1	0.422	0.510	0.466	0.950	1.010	0.980	120
2	0.221	0.320	0.270	0.430	0.442	0.439	125
3	0.471	0.478	0.474	0.021	0.030	0.025	130
4	0.009	0.010	0.009	0.008	0.010	0.009	135
5	0.292	0.233	0.212	0.410	0.456	0.433	140
6	0.443	0.541	0.492	0.021	0.056	0.038	145
7	0.531	0.544	0.537	0.032	0.098	0.065	150
8	0.395	0.400	0.397	0.245	0.299	0.272	155

Table 7. Post-learning energies

Hrs	Days							
	1	2	3	4	5	6	7	8
0	0.246	0.221	0.221	0.212	0.231	0.221	0.212	0.330
1	0.246	0.212	0.232	0.212	0.246	0.200	0.215	0.387
2	0.239	0.212	0.211	0.321	0.253	0.201	0.239	0.355
3	0.222	0.214	0.211	0.213	0.201	0.211	0.299	0.299
4	0.221	0.211	0.245	0.230	0.253	0.231	0.222	0.316
5	0.221	0.210	0.211	0.321	0.243	0.213	0.222	0.337
6	0.260	0.321	0.298	0.242	0.241	0.323	0.220	0.261
7	0.264	0.221	0.244	0.321	0.212	0.321	0.211	0.240
8	0.264	0.212	0.262	0.332	0.221	0.312	0.227	0.251
9	0.260	0.321	0.221	0.252	0.212	0.221	0.223	0.298
10	0.252	0.210	0.219	0.221	0.232	0.222	0.232	0.299
11	0.242	0.211	0.212	0.242	0.213	0.221	0.216	0.241
12	0.344	0.200	0.291	0.310	0.211	0.228	0.210	0.333
13	0.311	0.231	0.277	0.319	0.212	0.221	0.200	0.331
14	0.310	0.242	0.310	0.305	0.211	0.219	0.288	0.349
15	0.323	0.231	0.200	0.331	0.209	0.219	0.276	0.300
16	0.324	0.223	0.299	0.322	0.201	0.244	0.231	0.314
17	0.311	0.222	0.299	0.351	0.200	0.212	0.299	0.344
18	0.241	0.242	0.321	0.281	0.270	0.331	0.252	0.300
19	0.229	0.272	0.400	0.221	0.288	0.399	0.288	0.283
20	0.264	0.261	0.383	0.271	0.203	0.302	0.253	0.299
21	0.214	0.292	0.313	0.223	0.255	0.319	0.219	0.273
22	0.221	0.292	0.330	0.332	0.288	0.371	0.297	0.282
23	0.231	0.214	0.301	0.263	0.271	0.322	0.299	0.293

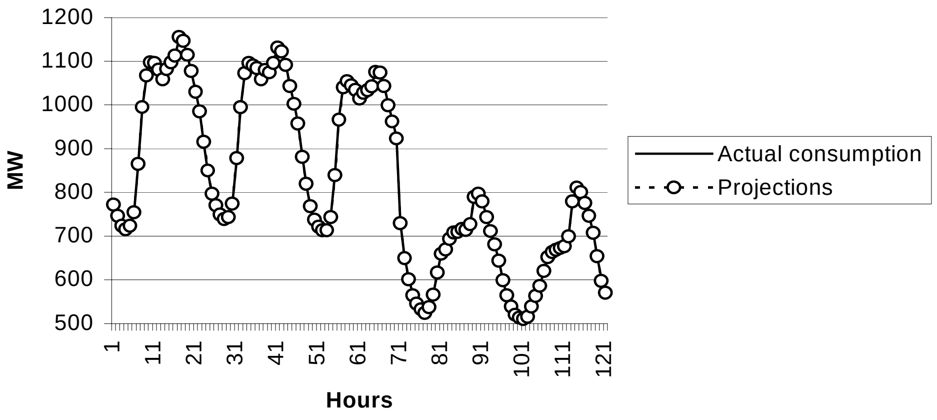


Fig. 6. Actual consumption and consumption projections, where each projected hour has its own FNN.

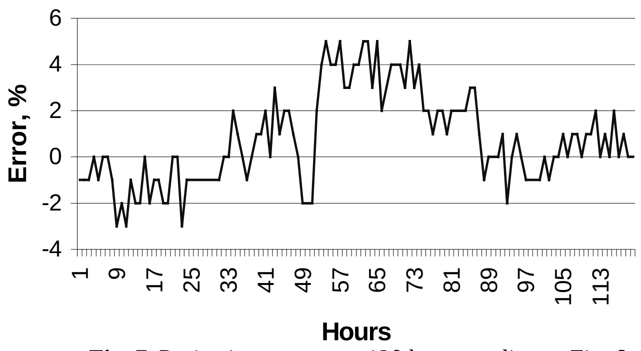


Fig. 7. Projection error over 120 hrs according to Fig. 6.

We applied the method to a feed-forward neural network, which allowed a successful real-time projection of power consumption for a period of up to 8 days. Figure 6 shows a 5-day projection where every hour of the day had its own network. The average error over the one-year testing period was 2.92%. Figure 7 shows the projection error of the graph of Fig. 6.

10. CONCLUSIONS

The article has analysed two important aspects related to regime planning: short-term projection of power consumption at high voltage and fault prediction in a power network based on ENav in the National Grid of Eesti Energia Ltd in the Oracle DB environment.

Various model classes were considered and criteria were proposed for the selection of a suitable model – time-series and transfer-function models, models based on trigonometric functions, state-spaced models, models based on orthogonal transformation, hierarchical models, and neural networks. The last one was selected.

Our models for consumption projection were implemented as a two-layered feed-forward neural network with 18 inputs, reflecting data for a period of six hours, 42 neurons on the hidden layer, and one neuron as an output that provided consumption projection of the respective hour of the day. The projection of every hour had a network of its own, which was applied to all the periods. For the first four days, an average projection error of ±3% and for the last four days an average projection error of ±6% was achieved.

The model for fault prediction was implemented with 20 inputs, 60 neurons on the hidden layer and one in the output layer. The paper explains FNN topology selection and compares four different learning rules in four types of experiments. We found that experiments E1–E3 are quite similar from the point of view of the convergence speed of the learning rule. Experiment E4 was not efficient as for the convergence speed though we used more data than in other experiments. We found that in the majority of cases the Widrow–Hoff Learning Rule was the most effective one in terms of the convergence speed of the learning rule. We found empirically that an average percentage of facility fault prediction is about 70. In the case of a facility located in the building, the detected percentage was about 80. Since the percentage of breakdown detection is more than 70 when as input data only planned, non-planned facility downtimes, and fault downtime was used, we can assume that the non-planned conditions were ignored (weather information, performer qualifications, wrong work planning etc.). Further studies should focus on the integration of facility passport data and restrictions and statistical analysis of data functions to increase the percentage of detection. In the ENav application, the fault prediction functionality is under development.

The software for short-term projection of power consumption has gone through two developments since the requirements to accuracy have increased. When from preliminary version of the software an average projection error of ±3% for the first four days and ±6% for the last four days was required, then the new version gives projections on hourly basis. In order to achieve this goal, the following steps were taken. First, we abandoned the FNN structure since our goal was to obtain greater temperature sensitivity. The model described in this article proved that FNN's first layer became quite insensitive to inputs, i.e., it achieved the local minimum and could not depart from it. In order to avoid this problem, the logistic function, acting as an activation function, was replaced with the hyperbolic tangent. In addition to temperature, light intensity, power consumption, week day (1–7), month (1–12), and threshold were added as inputs. Six neurons were applied to the hidden layer. The first neuron takes into account power consumption of the first day, light intensity, and temperature. Second neuron uses only the current day’s current projected temperature. The third neuron uses only the day before yesterday’s hourly power consumption. The forth neuron uses as its input a specified weekday, and the fifth neuron uses a specified month. The sixth neuron uses the threshold. Output layer neuron is tied with all the neurons on the hidden layer. Rest of the logic remained unchanged. Applying the above logic we were able to achieve the data quality that was required. The next step involved developing of an algorithm that synchronizes the shape of the function that is based on the results, projected by different neural networks, since consumption is projected by different neural networks that are autonomous. Thanks to mentioned synchronization, the accuracy of projected results further improved, but the speed of the algorithm decreased. For the time being an algorithm that increases the levelling engines convergence speed is being elaborated. Currently the short-term projection of power consumption is again in an experimental state due to just described further developments.

REFERENCES

1. *Program Application Guide, PSS/E 25*. **1**, **2**, Power Technologies Inc., Schenectady, New York, 1997.
2. *Power System Simulator for Engineering – PSS/E*. http://www.pti-us.com/pti/software/psse/psse.htm. Accessed 01 March 1999.
3. *Estonian Power System, Tallinn, Estonia, National Dispatch Center for SE Eesti Energia*. Vol. 2: *SCADA/EMS Functionality*, Vol. 3: *Appendixes*. Harris Energy Control Systems. Melbourne, Florida, 1997, Oct. 15, ref: 8020-49.
4. *Estonian Power System, Eesti Energia, Bidding Documents. Procurement of Dispatch Center and Remote Terminal Units. Supply & Installation*. Vol. 2. Prepared by Eesti Energia, Electrotek Concepts and EPIC Engineering under contract with the United States Trade and Development Agency, 1997, Oct. 14, ref: ME 7 3O5 0005.
5. *Proposal for Procurement of Dispatch Center and Remote Terminal Units*. Vol. 2. IVO Power Engineering Ltd, Vantaa, 1997, Oct. 14, ref: PSC/EST-A2-375.
6. *SINAUT Spectrum System Description*. SIEMENS, 1997, Oct. 7, ref: U3452-O-U000-053500.
7. Bunn, D. W. and Farmer, E. D. *Comparative models for electrical load forecasting*. J. Wiley, Belfast, 1985.
8. *AR Model*. http://www-ssc.igpp.ucla.edu/personnel/russell/ESS265/Ch9/autoreg/ node9.html. Accessed 10 October 2000.
9. Draper, N. and Smith, H. *Applied Regression Analysis*, 2nd ed., J. Wiley, New York, 1981.
10. Tiao, G. C. Autoregressive moving average models, intervention problems and outlier detection in time series. In *Handbook of Statistics,* Elsevier Science, Amsterdam, 1985, **5**, 85–118.
11. Berger, J. O., Fienberg, S. E., and Olkin, I. *ARMA Model Identification*. Springer-Verlag, New York, 1992.
12. *ARIMA*. http://www.geocities.com/Colosseum/5585/mprev.html. Accessed 10 October 2000.
13. Box, G. E. P. and Jenkins, G. M. *Time Series Analysis Forcasting and Control.* Holden-Day, San Francisco, 1976.
14. To, F. W. and Tsang, K. M. *Three-dimensional Object Recognition Using an Orthogonal Complex AR Model Approach*. http://ejournals.wspc.com.sg/ijprai/14/preserved-docs/1402/S021800140000009X.pdf. Accessed 10 October 2000.
15. Freund, R. J. and Minton, P. D. *Regression Methods*. Marcel Dekker, New York, 1979.
16. Pentland, A. and Sclaroff, S. Closed-form solutions for physically based modelling and recognition. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 1991, **13**, 715–729.

17. Shibata, R. Various model estimation techniques in time series analyses. In *Handbook of Statistics*, Elsevier Science, Amsterdam, 1985, **5**, 179–187.
18. Lanitis, A., Taylor, C. J., and Cootes, T. F. A Generic system for classifying variable objects using flexible template matching. In *Proc. British Machine Vision Conference*. BMVA Press, 1993, 329–338.
19. Sozou, P. D., Cootes, T. F., and Taylor, C. J. A non-linear generalisation of point distribution models using polynomial regression. In *Proc. British Machine Vision Conference*. BMVA Press, 1994, 397–406.
20. Bodger, P. S., Brooks, D. R. D., and Moutter, S. P. Spectral decomposition of variations in electricity loading using mixed radix fast Fourier transform. *IEE Proc.*, 1987, **134**, 197–202.
21. Harvey, A. C. *Forecasting, Structural Time Series Model and the Kalman Filter*. Cambridge Univ. Press, Cambridge, 1990.
22. Kangilaski, T. The Kalman filter for numerical processing of radar data. In *6th Biennial on Electronics and Microsystem Technology, Baltic Electronics Conference*, BEC 1998, Tallinn, 1998, 123–126.
23. Mendel, J. M. *Lessons in Digital Estimation Theory*. Prentice-Hall, Engelwood Cliffs, N.J., 1987.
24. Szelag, C. R. A short term forecasting algorithm for trunk demand servicing. *Bell System Tech. J.*, 1994, **61**, 67–77.
25. Young, P. Recursive extrapolation, intrapolation and smoothing of non-stationary time-series. In *Identification and System Parameter Estimation* (Chen, H. F., ed.). Pergamon, Oxford, 1988.
26. Mehra, R. K. Kalman filters and their applications to forecasting. *Manage. Sci.*, 1979, **12**, 75–94.
27. Ethier, S. N. and Kurtz, T. G. *Markov Processes: Characterisation and Convergence*. J. Wiley, New York, 1987.
28. Sastri, T. A state-space modelling approach for time series forecasting. *Manage. Sci.*, 1985, **31**, 1451–1470.
29. *Singular Value Decomposition*. http://www.cs.ut.ee/~toomas_l/linalg/lin2/node11.html. Accessed 14 August 2001.
30. Kanjila, P. P. and Palit, S. The singular value decomposition – applied in the modelling and prediction of quasiperiodic processes. *Signal Process.*, 1994, **35**, 257–267.
31. Kanjila, P. P. and Palit, S. On the singular value decomposition, applied in the analysis and prediction of almost periodic signals. *Signal Process.*, 1994, **40**, 269–285.
32. Klema, V. C. and Laub, A. J. The singular value decomposition: its computation and some applications. *IEEE Trans. Autom. Control*, 1980, **AC-25**, 164–176.
33. Golub, G. H. *Matrix Computations*, 2nd ed. The Johns Hopkins Univ. Pr., Baltimore, 1989.
34. *Eigenvalues and Eigenvectors Technique*. http://www.sosmath.com/diffeq/system/linear/eigenvalue/eigenvalue.html. Accessed 14 August 2001.
35. Deprette, F. *SVD and Signal Processing: Algorithms, Applications and Architectures*, North-Holland, Amsterdam, 1988.
36. Kanjilal, P. P., Palit, S., and Rao, G. P. On the modelling and prediction of quasiperiodic signals using SVD and neural networks. In *IFAC Symposium on System Identification, SYSID'94*. Copenhagen, 1994, 1536–1540.
37. Malliopoulos, C., Bakamidis, C., and Carayannis, G. Order determination and optimum harmonic reconstruction of quasi-periodic signals in noise. *Signal Process.*, 1999, **79**, 161–173.
38. Bhattacharya, T. K. and Basu, T. K. Medium range forecasting of hourly power system load by time series analysis using Walsh transform. *Electr. Power Energy Syst.*, 1991, **13**, 193–200.
39. Farlow, S. J. *Self-Organizing Methods in Modelling: GMDH Type Algorithms*. Marcel Dekker, New York, 1988.
40. Ikeda, S., Ochiai, M., and Sawaragi, Y. Sequential GMDH algorithm and its application to river flow prediction. *IEEE Trans. Syst. Man and Cybern.*, 1976, **SMC-6**, 473–479.

41. Kortmann, M. and Unbehauen, H. Two algorithms for model structure determination of nonlinear dynamic systems with applications to industrial process. In *Preprints 8th IFAC Symposium on Identification and System Parameter Estimation*. Beijing, 1988, **2**, 939–946.
42. Kanjilal, P. P. *Adaptive Prediction and Predictive Control*. Peter Peregrinus, London, 1995.
43. Amari, S. Mathematical foundations of neurocomputing. *IEE Proc.*, 1990, **78**, 1443–1463.
44. Azoff, E. *Neural Network Time Series Forecasting of Financial Markets*. J. Wiley, Chichester, 1994.
45. Haykin, S. *Neural Networks. A comprehensive Foundation*. Macmillan, New York, 1994.
46. Isermann, R. and Balle, P. Trends in the application of model based fault detection and diagnosis of technical processes. In *Proc. IFAC World Congress*, San Francisco, 1996.
47. Muresanu, S. *Neural Networks Based Image Recognition*. Technical Report, Katholieke Hogeschool Sint-Lieven, Ghenth, Belgium, May, 2000.
48. Gori, M. and Scarselli, F. Are multilayer preceptrons adequate for pattern recognition and verification? *IEEE Trans. Pattern Anal. Mach. Intelligence*, 1998, **20**, 1121–1132.
49. Ontanu, D.-M. Learning by evolution. A new class of general classifier neural networks and their training algorithm. *Adv. Model. Anal.*, 1993, **26**, 27–30.
50. Weigend, A. S. and Gershenfeld, N. A. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison Wesley, Santa Fe, 1994.
51. Franses, P. *Time Series Models for Business and Economic Forecasting*. Cambridge Univ. Pr., Cambridge, 1998.
52. Holton, W. J. and Keating, B. *Business Forecasting*, 3rd ed., Irvin/Mc Graw-Hill, Boston, 1998.
53. Makridakis, S. *Forecasting, Planning and Strategy for the 21st Century*. Free Press, Collier MacMillan, New York, 1990.
54. Riddington, G. L. Time varying coefficient models and their forecasting performance. *Omega*, 1993, **21**, 573–583.
55. Tashman, L. J. and Leach, M. L. Automatic forecasting software: A survey and evaluation. *Int. J. Forecast.*, 1991, **7**, 209–230.
56. Collopy, F., Adya, M., and Armstrong, J. S. Expert systems for forecasting. In *Principles of Forecasting* (Armstrong, J. S., ed.). Kluwer, Norwell, MA, 2001.
57. MacGregor, D. G. Decomposition in judgmental forecasting and estimation. In *Principles of Forecasting* (Armstrong, J. S., ed.). Kluwer, Norwell, MA, 2001.
58. Mentzer, J. T. and Kahn, K. B. Forecasting technique familiarity, satisfaction, usage, and application. *J. Forecast.*, 1995, **14**, 465–476.
59. Willemain, T. R., Smart, C. N., Schockor, J. H., and DeSautels, P. A. Forecasting intermittent demand in manufacturing: A comparative evaluation of Croston's method. *Int. J. Forecast.*, 1994, **10**, 529–538.
60. Wittink, D. R. and Bergestuen, T. Forecasting with conjoint analysis. In *Principles of Forecasting* (Armstrong, J. S., ed.). Kluwer, Norwell, MA, 2001.
61. Cancelo, J. R. and Espasa, A. *Forecasting Daily Demand for Electricity with Multiple Intput Nonlinear Transfer Function Models: a Case Study*, Departamento de Economía, Universidad Carlos III de Madrid. Working paper 91, 21 July, 1991.
62. Chishti, S. Recursively bootstrapped probability distribution of electricity demand forecast in Pakistan. *J. Energy Develop.*, Spring 1993, 223–231.
63. Carlin, J. B. and Dempster, A. P. Sensitivity analysis of seasonal adjustments: empirical case studies (with discussion). *J. Am. Statist. Assoc.*, 1989, **84**, 6–32.
64. Lamedica, R., Prudenzi, A., Sforna, M., Caciotta, M., and Cencelli, V. O. A neural network based technique for short-term forecasting of anomalous load periods. *IEEE Trans. Power Syst.*, 1996, **11**, 1749–1756.
65. Seppälä, A. *Load Research and Load Estimation in Electricity Distribution*. Dissertation, Technical Research Center of Finland, VTT Publications 289, Espoo, 1996.
66. Cybenco, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst.*, 1989, **2**, 303–314.
67. Funahashi, K. On the approximate realization of continuous mappings by neural networks. *Neural Netw.*, 1989, **2**, 183–192.

68. Hornik, K., Stinchomber, M., and White, H. Multilayer feedforward network are universal approximators. *Neural Netw.*, 1989, **2**, 359–366.
69. Daugherty, E. L. and Bartlett, E. B. Short-term electric load forecasting using neural networks. In *Proc. Iowa State University Electric Power Research Center Power Affiliate Research Program.* Iowa State University, Ames, Iowa, 1993, 291–302.
70. Doraiswami, R. Performance monitoring and fault prediction using a linear predictive coding algorithm. *Automatica*, 1993, **29**, 4.
71. Bartlett, E. B. Analysis of chaotic population dynamics using artifical neural networks. *Chaos, Solitons*, *Fractals*, 1992, **1**, 413–421.
72. Basu, A. and Bartlett, E. B. Detecting faults in nuclear power plants by using dynamic node architecture artificial neural networks. *Nucl. Sci. Eng.*, 1994, **116**, 313–325.
73. Bartlett, E. B. Self determination of input variable importance by neural networks. *Neural Parallel Sci. Comput.*, 1994, **2**, 103–114.
74. Kim, K. and Bartlett, E. B. Error prediction for a nuclear power plant fault-diagnostic advisor using neural networks. *Nucl. Technol.*, 1994, **108**, 283–297.
75. Kim, K. and Bartlett, E. B. Nuclear power plant fault diagnosis using neural networks with error estimation by series association. *IEEE Trans. Nucl. Sci.*, 1996, **43**, 2372–2388.
76. Reinschmidt, K. F. and Ling, B. Neural networks for plant simulation and control. In *Proc. American Power Conference*. Illinois Institute of Technology, Chicago, IL, 1994, **2**, 796–801.
77. Moechtar, M., Farag, A. S., Hu, L., and Cheng, T. C. Combined genetic algorithms and neural-network approach for power-system transient stability evaluation. *Eur. Trans. Electr. Power*, 1999, **9**, 115–122.
78. Rebizant, W. and Szafran, J. Power system fault detection and classification using probabilistic approach. *Eur. Trans. Electr. Power*, 1999, **9**, 183–191.
79. Rebizant, W. and Szafran, J. Power system fault detection and classification using probabilistic approach. *Eur. Trans. Electr. Power*, 1999, **9**, 247–253.
80. Oswald, B. R. Generalized method for fault simulations in power systems. *Eur. Trans. Electr. Power*, 2000, **10**, 59–62.
81. *Log-Linear, Logit, and Probit Models*. http://www2.chass.ncsu.edu/garson/pa765/logit.htm. Accessed 14 August 2001.
82. Moscinski, J. *Advanced Control with Matlab and Simulink.* Ellis Horwood, London, 1995.
83. Priestly, M. B. *Nonlinear and Nonstationary Time Series Analysis.* Academic Press, London, 1992.
84. Pineda, F. J. Generalization of back-propagation to recurrent and higher order neural networks. In *Neural Information Processing Systems*. American Institute of Physics, New York, 1988, 602–611.

Sujuva töö tagamine elektrienergia ülekandevõrgus edasisidestatud neuronvõrkude abil

Taivo Kangilaski

Artiklis on kirjeldatud edasisidestatud neuronvõrkude kasutamist elektrienergia ülekandevõrku haldavas tarkvaras. On vaadeldud elektrienergia tarbimise prognoosi (kõrgepinge osas) ning rikkeliste ja/või avariiliste sündmuste prognoosi algoritme. On põhjendatud algoritmi valikut ning toodud edasisidestatud neuronvõrgu erinevate mudelite katseandmed, leidmaks sobivaimat konfiguratsiooni. Uuringu aluseks on Eesti Energia AS-i tarkvara “Sündmuste registraator”, mille abil püütakse tagada Eesti Vabariigi elektrisüsteemi tõrgeteta tööd põhivõrgus.