Adaptively Trained Neural Networks and Their Application to Electric Load Forecasting

Dong C. Park and Osama Mohammed
Dept. of Electrical and Computer Eng.
Florida International University
Miami, FL 33199

M.A. El-Sharkawi and R.J. Marks II
Dept. of Electrical Engineering
University of Washington
Seattle, WA 98195

Abstract

A training procedure that adapts the weights of a trained layered perceptron type artificial neural network to training data originating from a slowly varying non-stationary process is proposed. The resulting adaptively trained neural network (ATNN), based on nonlinear programming techniques, is shown to adapt to new training data that is in conflict with earlier training data without affecting the neural networks' response minimally to data elsewhere. When the ATNN is applied to the problem of electric load forecasting, it is shown to significantly outperform the conventionally trained layered perceptron.

1 Introduction

In the training of layered perceptron, an assumption of stationarity of the training data is typically made. In a number of cases of interest, however, the training data is a slowly varying nonstationary process. We desire a training procedure that adapts the trained perceptron's response to the current training data, but does not require detailed knowledge of the previous training data. In order for the layered perceptron's weights to adapt to the slowly varying non stationarity, such a procedure should (1) still respond appropriately to previous training data if that data is not in conflict with the new training data and (2) adapt to the new training data, even when it is conflict with portions of the old data.

We propose a procedure for such adaptation which is applicable when the training data's stationarity varies sufficiently slowly. Our procedure for adaptive updating assures proper response to previous training data by seeking to minimize a weight sensitivity cost function while, at the same time, minimizing the mean square error normally ascribed to the layered perceptron. The process is illustrated through application to an interpolation problem and by its use on a electric load forecasting problem with data collected by a power industry.

2 Formulation of Problem

A three layered perceptron type neural network (NN) is used to illustrate the ATNN algorithm. A general derivation of the ATNN algorithm with more than three layers can be found in [1]. Assume a NN with N sets of data,

$$\{(\mathbf{x}(1), d(1)), (\mathbf{x}(2), d(2)), \cdots, (\mathbf{x}(\mathrm{N}), d(\mathrm{N}))\}$$

where we assume that $\mathbf{x}(i)$ is I-dimensional vector and $d(i)$ is scalar. The NN is assumed to have one hidden layer with h hidden neurons. The matrix $\mathbf{W}$ represents the weight matrix between the input and hidden neurons and $\mathbf{v}$ denotes the weight vector which links the hidden and output neurons. The dimensions of $\mathbf{W}$ and $\mathbf{v}$ are $I \times h$ and $h \times 1$, respectively.

For a given input data vector, $\mathbf{x}(i)$, the output of the NN, $y(i)$, is given by

$$y(i) = f[\mathbf{v}^{\mathrm{T}}\mathbf{u}] \text{ and } \mathbf{u} = \mathbf{f}[\mathbf{W}^{\mathrm{T}}\mathbf{x}(i)] \qquad (1)$$

where u_j, $1 \leq j \leq h$, represents the activation of the j^{th} hidden neuron, the superscript T denotes the transpose of a matrix or vector, and $f[\cdot]$ is the sigmoid function, $f[x] = 1/(1+e^{-x})$, $x \in \Re$.

We assume that $\mathbf{W}(\mathrm{N})$ and $\mathbf{v}(\mathrm{N})$ are the weights that minimize the error function with N sets of data [2]:

$$E(\mathrm{N}) = \frac{1}{2}\sum_{i=1}^{N}(d(i) - y(i))^2. \qquad (2)$$

3 Problem Statement

The objective of ATNN is given as follows:

Given $\mathbf{W}(\mathrm{N})$, $\mathbf{v}(\mathrm{N})$, and $(N+1)$ sets of data, determine $\mathbf{W}(\mathrm{N}+1)$ and $\mathbf{v}(\mathrm{N}+1)$ such that

$$E(\mathrm{N}+1) = E(\mathrm{N}) + \frac{1}{2}(d(\mathrm{N}+1) - y(\mathrm{N}+1))^2 \qquad (3)$$

is minimized in such a manner that $y(\mathrm{N}+1) \approx d(\mathrm{N}+1)$.

4 Algorithm Development

In order to accomplish the given objective, the linearization process around the current operating point, is first used. Equations in (1) yield

$$d(\mathrm{N}+1) = f[\mathbf{v}^{\mathrm{T}}(\mathrm{N}+1)\mathbf{u}] = f[(\mathbf{v}(\mathrm{N}) + \Delta\mathbf{v})^{\mathrm{T}}\mathbf{u}] \qquad (4)$$

where

$$\mathbf{u} = \mathbf{f}[(\mathbf{W}(\mathrm{N}) + \Delta\mathbf{W})^{\mathrm{T}}\mathbf{x}(\mathrm{N}+1)]. \qquad (5)$$

The linearization process expands (4) and (5) in a truncated Taylor series about the state of interest, $\{\ \mathbf{x}(\mathrm{N}+1), \mathbf{W}(\mathrm{N}+1)\ \}$. Such a linearization is used in other adaptive signal processing techniques such as the extended Kalman filtering [3] and quasi-linearization [4].

If we define $\mathbf{b} = \mathbf{W}^{\mathrm{T}}(\mathrm{N})\mathbf{x}(\mathrm{N}+1)$ and $\Delta\mathbf{b} = \Delta\mathbf{W}^{\mathrm{T}}\mathbf{x}(\mathrm{N}+1)$ and apply a first order Taylor series expansion to (5), we get

$$\mathbf{u} = \mathbf{f}[\mathbf{b} + \Delta\mathbf{b}] \approx \mathbf{f}[\mathbf{b}] + (\nabla_{\mathbf{b}}\mathbf{f}[\mathbf{b}])\,\Delta\mathbf{b}$$

CH 3006-4/91/0000 – 1125 $1.00 © IEEE

where $\nabla_{\mathbf{b}}\mathbf{f}[\mathbf{b}]$ is the gradient of $\mathbf{f}[\mathbf{b}]$ with respect to $\mathbf{b}$. This approximation limits the perturbation, $\mathbf{f}[\mathbf{b}]$ to be small enough so that $\Delta\mathbf{b} \ll \mathbf{f}[\mathbf{b}]/H$, where H is the Hessian of $\mathbf{f}[\mathbf{b}]$ and $H \ll 1$.

Since $f_i[\cdot] = f[\cdot]$, the i^{th} component of $\mathbf{f}[\mathbf{b}]$ is only a function of b_i (that is, $f_i[\mathbf{b}] = f_i[b_i], 1 \leq i \leq h$) and

$$\frac{\partial f_i[\mathbf{b}]}{\partial b_j} = f[b_i](1 - f[b_i])\, \delta_{i-j} = u_i^*(1 - u_i^*)\, \delta_{i-j},$$

this yields

$$\nabla_{\mathbf{b}}\mathbf{u}^* = \mathrm{diag}[u_1^*(1 - u_1^*), \cdots, u_h^*(1 - u_h^*)]$$

where δ_k, the Kronecker Delta, is one for $k = 0$ and is otherwise zero. Also, $u_i^* = f[b_i]$ is the activation of i^{th} hidden neuron for the new input data with the old weight such as

$$\mathbf{u}^* = [u_1^*, u_2^*, \cdots, u_h^*]^{\mathrm{T}} = \mathbf{f}[\mathbf{W}^{\mathrm{T}}(\mathrm{N})\mathbf{x}(\mathrm{N}+1)].$$

Therefore, the activations of hidden neurons are given by:

$$\mathbf{u} \simeq \mathbf{u}^* + (\nabla_{\mathbf{b}}\mathbf{u}^*)\, \Delta\mathbf{W}^{\mathrm{T}}\, \mathbf{x}(\mathrm{N}+1). \qquad (6)$$

From (4), we get

$$f^{-1}[d(\mathrm{N}+1)] = \mathbf{v}^{\mathrm{T}}(\mathrm{N})\mathbf{u} + \Delta\mathbf{v}^{\mathrm{T}}\mathbf{u} \qquad (7)$$

where $f^{-1}[x] = \ln(x/(1-x))$.

By substituting (6) into (7) and assuming $\Delta\mathbf{v} \ll \mathbf{v}$, we get

$$\begin{aligned} & f^{-1}[d(\mathrm{N}+1)] - \mathbf{v}^{\mathrm{T}}(\mathrm{N})\mathbf{u}^* \\ & \quad \simeq \mathbf{v}^{\mathrm{T}}(\mathrm{N})(\nabla_{\mathbf{b}}\mathbf{u}^*)\Delta\mathbf{W}^{\mathrm{T}}\mathbf{x}(\mathrm{N}+1) + \Delta\mathbf{v}^{\mathrm{T}}\mathbf{u} (8) \end{aligned}$$

The weight perturbation matrix $\Delta\mathbf{W}$ is now rearranged to a vector form as follows:

$$\Delta\mathbf{W}_{vec} = [\Delta W_{vec,1}\Delta W_{vec,2} \cdots \Delta W_{vec,p}]$$

where $p = h \times I$. Then Eq. (8) can be rewritten as

$$c_1 = [\Delta\mathbf{W}_{vec}^{\mathrm{T}} : \Delta\mathbf{v}^{\mathrm{T}}][\mathbf{u}^{\dagger\mathrm{T}} : \mathbf{u}^{*\mathrm{T}}]^{\mathrm{T}} = \mathbf{z}^{\mathrm{T}}\mathbf{a} \qquad (9)$$

where $c_1, \mathbf{a}$, and $\mathbf{z}$ are vectors defined as

$$c_1 \stackrel{\mathrm{def}}{=} f^{-1}[d(\mathrm{N}+1)] - \mathbf{v}^{\mathrm{T}}(\mathrm{N})\mathbf{u}^*, \quad \mathbf{a} \stackrel{\mathrm{def}}{=} [\mathbf{u}^{\dagger\mathrm{T}} : \mathbf{u}^{*\mathrm{T}}]^{\mathrm{T}},$$

$$\mathbf{z} \stackrel{\mathrm{def}}{=} [\Delta\mathbf{W}_{vec}^{\mathrm{T}} : \Delta\mathbf{v}^{\mathrm{T}}]^{\mathrm{T}},$$

and $\mathbf{u}^\dagger$ is a solution of

$$\Delta\mathbf{W}_{vec}^{\mathrm{T}}\mathbf{u}^\dagger = \mathbf{v}^{\mathrm{T}}(\mathrm{N})\mathbf{Q}\Delta\mathbf{W}^{\mathrm{T}}\mathbf{x}(\mathrm{N}+1). \qquad (10)$$

Since Eq. (11) has $h \times (I+1)$ unknowns, there exist many sets of solutions. The most suitable solution among them should have minimum effect to the previous set of data. We are therefore motivated to form a sensitivity matrix of $y(i)$ over a weight change.

If we define

$$E_i = \frac{1}{2}(d(i) - y(i))^2,$$

then the sensitivity measure for $\mathbf{W}$ and $\mathbf{v}$ are:

$$\begin{aligned} \frac{\partial E_i}{\partial w_{jk}} &= -[d(i) - y(i)](\frac{\partial y(i)}{\partial w_{jk}}) \\ \frac{\partial E_i}{\partial v_k} &= -[d(i) - y(i)](\frac{\partial y(i)}{\partial v_k}) \qquad (11) \end{aligned}$$

where w_{jk}, the weight of interconnection between input neuron j and hidden neuron k, is the jk^{th} element of $\mathbf{W}$.

Now, define $SW_{i,jk}$ to be the sensitivity of $y(i)$ due to small changes in W_{jk} and $SV_{i,k}$ to be the sensitivity of $y(i)$ to the v_k's:

$$\begin{aligned} SW_{i,jk} &= y(i)(1 - y(i))\, v_k\, u_k(1 - u_k)\, x_j \\ SV_{i,k} &= y(i)(1 - y(i))\, u_k \qquad (12) \end{aligned}$$

As a consequence, the change in E_i, ΔE_i, due to the perturbations in W_{jk} and V_k yields

$$\begin{aligned} \Delta E_i &= \sum_{j,k}(\frac{\partial E_i}{\partial w_{jk}})\, \Delta\mathbf{W}_{jk} + \sum_k(\frac{\partial E_i}{\partial v_k})\, \Delta\mathbf{v}_k \\ &= (\varepsilon_i)[\Delta W_{vec,1} \cdots \Delta W_{vec,p} : \Delta v_1 \cdots \Delta v_h] \\ & \quad \times[SW_{i,1} \cdots SW_{i,p} : SV_{i,1} \cdots SV_{i,h}]^{\mathrm{T}} \qquad (13) \end{aligned}$$

where $\varepsilon_i = -(d(i) - y(i))$, $1 \leq i \leq N$ and $p = I \times h$.

Equivalently,

$$\Delta\mathbf{E} = \mathbf{A}\,\mathbf{S}\,\mathbf{z} \qquad (14)$$

where

$$\mathbf{A} = \mathrm{diag}[\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_N]$$

$$\mathbf{S} = \begin{bmatrix} SW_{1,1} & \cdots & SW_{1,p} & SV_{1,1} & \cdots & SV_{1,h} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ SW_{N,1} & \cdots & SW_{N,p} & SV_{N,1} & \cdots & SV_{N,h} \end{bmatrix},$$

where $\mathbf{z}$ is a $q \times 1$ vector defined in (9) and $q = p + h = (I+1) \times h$.

With the presense of the $(N+1)^{st}$ data which requires $d(\mathrm{N}+1) = y(\mathrm{N}+1)$, the objective function of (3) can be changed to

$$J = \frac{1}{2}\sum_{i=1}^{N}(E_{i,W(N)} - E_{i,W(N+1)})^2 = \frac{1}{2}\sum_{i=1}^{N}\Delta E_i^2$$

where $E_{i,W(N)}$ and $E_{i,W(N+1)}$ are the errors for the i^{th} data coupling with $\{\mathbf{W}(\mathrm{N}), \mathbf{v}(\mathrm{N})\}$ and $\{\mathbf{W}(\mathrm{N}+1), \mathbf{v}(\mathrm{N}+1)\}$, respectively. Or in matrix form by Eq. (14),

$$J = \frac{1}{2}(\Delta\mathbf{E})^{\mathrm{T}}\,(\Delta\mathbf{E}) = \frac{1}{2}\mathbf{z}^{\mathrm{T}}\,\mathbf{K}\,\mathbf{z} \qquad (15)$$

where $\mathbf{K} = \mathbf{S}^{\mathrm{T}}\,(\mathbf{A}^{\mathrm{T}}\mathbf{A})\,\mathbf{S}$. Note that $\mathbf{K} = \mathbf{K}^{\mathrm{T}}$.

The problem we have turns out to be a standard nonlinear programming problem. Specifically,

$$\text{minimize} \quad J(\mathbf{z}) = \frac{1}{2}\mathbf{z}^{\mathrm{T}}\,\mathbf{K}\,\mathbf{z}, \quad \text{s.t.} \quad \mathbf{z}^{\mathrm{T}}\,\mathbf{a} = c_1.$$

Geometrically, an equi-cost line of the cost function $J(\mathbf{z})$ represents a q-dimensional hyperellipse centered at the origin of the $\mathbf{z}$ plane and the constraint $\mathbf{z}^{\mathrm{T}}\,\mathbf{a} = c_1$ is a $(q-1)$ dimensional hyperplane on the $\mathbf{z}$ plane. Note that the the eigenvalues and eigenvectors of $\mathbf{K}$ determine the shape of $J(\mathbf{z})$. Since $\mathbf{z}$ represents the perturbations in weights, $\mathbf{z}$ should be limited to meet linearization assumption of the truncated Taylor series. At the same time, it is necessary to allow $\mathbf{z}$ to be large enough so that we can have at least one solution on the $\mathbf{z}^{\mathrm{T}}\,\mathbf{a} = c_1$ plane.

One way to find the boundary constraint in $\mathbf{z}$ is the use of the projection method; to find the projection point, $\hat{\mathbf{z}}$, on the $\mathbf{z}^{\mathrm{T}}\,\mathbf{a} = c_1$ plane from the origin of $\mathbf{z}$ and set the boundary such as $-m\hat{\mathbf{z}} \leq \mathbf{z} \leq m\hat{\mathbf{z}}$ with $m \geq 1$. The projection point is given by

$$\hat{\mathbf{z}} = c_1\,\mathbf{a}/\mathbf{a}^{\mathrm{T}}\,\mathbf{a}. \qquad (16)$$

A boundary, $-m\hat{\mathbf{z}} \leq \mathbf{z} \leq m\hat{\mathbf{z}}$, assures having an intersection with $\mathbf{z}^{\mathrm{T}}\,\mathbf{a} = c_1$. The larger the m, the smaller the $J(\mathbf{z})$ can be. The larger m, however, may allow the resulting solution to violate the Taylor series expansion used in (6).

By combining the selected boundary constraint with our problem, we can rewrite our problem as

$$\begin{aligned} &\text{minimize} \qquad J(\mathbf{z}) = \frac{1}{2}\mathbf{z}^{\mathrm{T}}\,\mathbf{K}\,\mathbf{z} \\ &\text{s.t.} \qquad \mathbf{z}^{\mathrm{T}}\,\mathbf{a} = c_1 \;\text{ and }\; -m\hat{\mathbf{z}} \leq \mathbf{z} \leq m\hat{\mathbf{z}} \quad (17) \end{aligned}$$

There exists a variety of methods to solve this type of problem in the field of nonlinear programming including the gradient projection method, the reduced gradient method, and the convex simplex method. Among these method, is the *reduced gradient method* (RGM) with a linear constraint chosen for our problem since it fits best our problem.

A brief description of the RGM will be given in this paper because of the limitation on the number of pages of paper. More detailed explanation about the RGM can be found in [5].

The RGM starts with partitioning of variables into two groups: independent and dependent. The cardinality of independent group is the number of linear constraints. Since there is one linear constraint in our problem, we will have one dependent variable. The other $(q-1)$ variables are assigned as independent variables.

The basic idea of the RGM is to minimize the cost function iteratively by using the steepest descent method. The negative gradient of each independent variable determines the direction of movement of the independent variable. The movement of a dependent variable is dictated by the linear constraint. If we denote dependent and independent variables as z_α and $\mathbf{z}_\beta$ respectively, the problem in (17) can be written as:

$$\begin{aligned} &\text{minimize} \quad J(\mathrm{z}_\alpha, \mathbf{z}_\beta) = \frac{1}{2}(\mathrm{z}_\alpha, \mathbf{z}_\beta)^{\mathrm{T}}\,\mathbf{K}\,(\mathrm{z}_\alpha, \mathbf{z}_\beta) \qquad (18) \\ &\text{s.t.} \quad \mathrm{z}_\alpha \mathrm{a}_\alpha + {\mathbf{a}_\beta}^{\mathrm{T}}\,\mathbf{z}_\beta = c_1 \;\text{ and }\; -m\hat{\mathbf{z}} \leq \mathbf{z} \leq m\hat{\mathbf{z}} \quad (19) \end{aligned}$$

where $\mathbf{a} = [\mathrm{a}_\alpha, {\mathbf{a}_\beta}^{\mathrm{T}}]^{\mathrm{T}}$.

The gradient of $J(\mathbf{z})$ is:

$$\nabla_{\mathbf{z}} J(\mathbf{z}) = (\nabla_{z_\alpha}\, J(\mathbf{z}),\; \nabla_{\mathbf{z}_\beta}\, J(\mathbf{z})) = \mathbf{K}\,\mathbf{z} = \mathbf{K}(z_\alpha, z_\beta).$$

From (19), the relationship between the movements of z_α and $\mathbf{z}_\beta$ is:

$$\Delta \mathrm{z}_\alpha = -\frac{1}{\mathrm{a}_\alpha}{\mathbf{a}_\beta}^{\mathrm{T}} \Delta \mathbf{z}_\beta. \qquad (20)$$

This movements in the iterations for minimum $J(\mathbf{z})$ guarentees that every solution is on the $(q-1)$ dimensional hyperplane given in (19). By using the relationship (20), the amount of movement for z_α, $\nabla_{z_\alpha}\, J(\mathbf{z})$, can be decomposed into the amount of movement for $\mathbf{z}_\beta$. The gradient with respect to $\mathbf{z}_\beta$, (the *reduced gradient* of $\mathbf{z}$), instead of $\mathbf{z}$, is found as:

$$\mathbf{r}^{\mathrm{T}} = \nabla_{\mathbf{z}_\beta}\, J(\mathrm{z}_\alpha, \mathbf{z}_\beta) - \frac{1}{\mathrm{a}_\alpha}\nabla_{z_\alpha}\, J(\mathrm{z}_\alpha, \mathbf{z}_\beta)\,\mathbf{a}_\beta \qquad (21)$$

where $\mathbf{r} = [r_1, r_2, \cdots, r_{q-1}]^{\mathrm{T}}$.

Since the solutions by the RGM should satisfy the inequality constraint given in (19), the vector $\mathbf{z}_\beta$ should not move to the direction of its gradient unless the resultant vector is inside of the boundary during its iteration.

We desire that the RGM provides us the global minimum solution of $J(\mathbf{z})$ within the boundary constraint. The globalness can be achieved if the cost function is a convex function and the boundary set formed by the constraint is a convex set. The convexity of the cost function $J(\mathbf{z})$ and the constraint set $\mathcal{B}$ is proven in [1] and [6]. Therefore, the solution by the RGM is global minimum with the given constraint.

5 Experiments and Results

The NN approach has been proposed for several power system applications including security assessment and electric load forecasting [7-9]. When NN approach is compared to classification method such as the Classification and Regression Trees, NN approach shows superior performance in terms of accuracy [10].

Recently proposed neural network approach for the electric load forecasting has several key features that make it highly suitable for the electric load forecasting. For example, it does not require any preassumed functional relationship between electric load and other weather variables. One can view the NN as a nonlinear mapping tool between weather variables and electric load without the need for predetermined model.

Although the NN is very promising tool in load forecasting, several key issues should be addressed. The nonstationarity of the electric loads and weather variables is one of them. Since most of the existing training algorithms including Error Back Propagation algorithm assume the stationarity of the training data, those algorithms have their own limitation of performance when they are applied to the electric load forecasting problem. In this sense, the proposed ATNN algorithm will be suitable for the electric load forecasting problem where the load profile is dynamic in nature with temporal, seasonal, and annual variations.

In order to demonstrate use of ATNN, we applied it to the electric load forecasting problem. Short-term load forecasting (several hours to a few days lead time) is a very useful tool for several applications such as economic allocation of generation, energy transaction, and system security analysis. Hourly temperature and load data for the Seattle/Tacoma area in the interval of Nov. 1, 1988 - Feb. 28, 1989 and Nov. 1, 1989 - Feb. 25, 1990 were collected and provided for us by the Puget Sound Power and Light Company.

For given data set, NN's were trained and tested for the hourly load forecasting with lead time of 48 hours. First, a standard Error Back Propagation algorithm is used for the initial weights of the NN using the first data set, Nov. 1,1988 - Feb. 28, 1989. The other data set is, then, continuously applied to the initially trained NN with 24 data each time. The ATNN adapts the trained NN by sequentially using the latest 24 data points (last 24 hours of data) while the regular NN simply augments training data set with 24 new entries. One of the topologies of NN's used for our experiment was as follows:

Input:$(k, T_{k-168}, T_{k-50}, T_{k-49}, T_{k-48}, T_k,$
$L_{k-168}, L_{k-50}, L_{k-49}, L_{k-48}),$

Output:L_k, and 4 hidden neurons in one hidden layer, where k is the hour of the day and T_i and L_j represent the temperature and load at the hour i and j, respectively.

To compare the performances of different training algorithms, the following error measure for daily average error was used:

$$\text{percent error} = \frac{1}{24}\sum_{k=1}^{24}\frac{\mid L_k - L'_k \mid}{L_k} \times 100\%$$

where L_k and L'_k denote actual and forecasted loads at hour k.

The mean(m) and standard deviation (σ) in percent errors of forecasting over the period of Nov. 1, 1989 through Feb. 25, 1990 were measured for the both of the Error Back Propagation (EBP) and ATNN algorithms. ATNN shows $m = 2.78\%$ and $\sigma = 0.47\%$ while EBP shows $m = 3.69\%$ and $\sigma = 0.77\%$. The improvement of ATNN over EBP was especially significant when the weather condition changes abruptly.

6 Conclusion

In this paper, an adaptive training algorithm for a layered perceptron has been proposed. The algorithm uses previousely trained network to adapt the weights for the new data. Conventional learning algorithms, such as error backpropagation, can give averaged result for inconsistent data which can occur often in a nonstationary environment.

This adaptively trained neural network utilizes linearization process around the current operating point and a nonlinear programming technique.

We have demonstrated use of the ATNN by its application to the nonstationary problem of electric load forecasting where the seasonal or annual load profile can change severely.

Acknowedgements

This work was supported by the Puget Sound Power and Light Co. (PSPL) and the Washington Technology Center at the University of Washington. The authors thank Mr. Milan L. Bruce of the PSPL, Dr. Seho Oh, and Professor Les E. Atals for their valuable comments.

References

[1] Dong Chul Park, *Identification of Stationary/ Nonstationary Systems Using Artificial Neural Networks*, Ph.D. Dissertation, University of Washington, Seattle, 1990

[2] D. Rumelhart, G.E. Hinton, R.J. Williams, " Learning internal representation by error propagation." In D. Rumelhart, J. McAlelland, and the PDP Research Group. *Parallel Distributed Processing : Explorations in the Microstructure of Cognition*, MIT Press, Cambridge, MA, 1986

[3] A. Gelb, *Applied Optimal Estimation*, The M.I.T. Press, Cambridge, Massachusetts, 1974

[4] K.S.P. Kumar and R. Shridar, " On the identification of control systems by the quasi-linearization method," IEEE Tr. on Automatic Control, Vol. AC-9, pp.151-154, 1964

[5] D. Luenberger,*Linear and Nonlinear Programming*, Addison-Wesley Publishing Co., Reading, MA, pp.295-321, 1984

[6] Dong C. Park, M. El-Sharkawi, R.J. Marks II, "An Adaptively Trained Neural Network," IEEE Tr. on Neural Networks (in press)

[7] Dong C. Park, M. El-Sharkawi, R. Marks II, and L. Atlas, "Electric load forecasting using an artificial neural network," IEEE Trans. Power Sys., May 1991

[8] M. Aggoune, M. El-Sharkawi, D. Park, M. Damborg, and R. Marks II, "Preliminary results on using artificial neural networks for security assessment," IEEE Trans. Power Sys., May 1991

[9] M. El-Sharkawi, R. Marks II, M. Aggoune, D. Park, M. Damborg, and L. Atlas, "Dynamic security assessment of power systems using back error propagation artificial neural networks," Proc. of the 2nd Annual Sym. on Expert Systems Appl. to Power Systems, Seattle, WA, July 1989 (to appear in IEEE Trans. Power Sys.)

[10] L. Atlas, R. Cole, Y. Muthusamy, A. Lippman, G. Connor, D. Park, M. El-Sharkawi, R. Marks II, " A performance comparison of trained multi-layer perceptrons and trained classification trees," Proc. of IEEE Special Issue on Neural Networks, Vol. 78, No. 10, pp.1614-1619, October 1990