Linear-Least-Squares Initialization of Multilayer Perceptrons Through Backpropagation of the Desired Response

Deniz Erdogmus, *Member, IEEE*, Oscar Fontenla-Romero, Jose C. Principe, *Fellow, IEEE*, Amparo Alonso-Betanzos, *Member, IEEE*, and Enrique Castillo

Abstract—**Training multilayer neural networks is typically carried out using descent techniques such as the gradient-based *backpropagation (BP) of error* or the quasi-Newton approaches including the Levenberg–Marquardt algorithm. This is basically due to the fact that there are no analytical methods to find the optimal weights, so iterative local or global optimization techniques are necessary. The success of iterative optimization procedures is strictly dependent on the initial conditions, therefore, in this paper, we devise a principled novel method of *backpropagating the desired response* through the layers of a multilayer perceptron (MLP), which enables us to accurately initialize these neural networks in the minimum mean-square-error sense, using the analytic linear least squares solution. The generated solution can be used as an initial condition to standard iterative optimization algorithms. However, simulations demonstrate that in most cases, the performance achieved through the proposed initialization scheme leaves little room for further improvement in the mean-square-error (MSE) over the training set. In addition, the performance of the network optimized with the proposed approach also generalizes well to testing data. A rigorous derivation of the initialization algorithm is presented and its high performance is verified with a number of benchmark training problems including chaotic time-series prediction, classification, and nonlinear system identification with MLPs.**

Index Terms—**Approximate least-squares training of multilayer perceptrons (MLPs), backpropagation (BP) of desired response, neural network initialization.**

I. Introduction

Although examples of neural networks have appeared in the literature as possible function approximation and adaptive filtering tools as early as 1950s [1]–[5], due to the extremely high computational loads required for optimally training these structures, the interest in them seemed to decline for about two decades. In the 1980s, Rumelhart *et al.* proposed the well-known and widely appreciated error backpropagation (BP) algorithm for training multilayer perceptrons (MLPs) and other neural networks belonging to Grossberg's additive model [6]. For the last two decades, while appreciating this brilliant algorithm, researchers have focused their efforts on improving the convergence properties of BP, the main concern being the slow convergence speed due to its gradient-descent nature. Among these, we mention the momentum term [6], [7], adaptive stepsizes [8], Amari's natural gradient [9], and more advanced search methods based on line search (conjugate gradient) [10], [11], and pseudo-Newton methods [10], [12]–[16], and even exact evaluation of the Hessian [17], [18]. A clear problem with all of these iterative optimization techniques is their susceptibility to local minima. In order to conquer this difficulty global search procedures have to be introduced. From simple methods using random perturbations [19] to more principled global search algorithms based on genetic algorithms [20] or simulated annealing [21], such proposals have also appeared in the literature; most without a striking success.

A less researched area deals with the search for good initialization algorithms. An interesting approach by Husken and Goerick was to utilize evolutionary algorithms to select a good set of initialization weights for a neural network from a set of optimal weight solutions obtained *a priori* for similar problems [22]. After all, if we could approximately choose the weights *close* to the global optimum, then BP or any other descent algorithm (e.g., Levenberg–Marquardt) could take the network weights toward the optimum fast and reliably. This initialization problem, however, is not trivial. Since typically random weight initialization is utilized, Thimm and Fiesler investigated the effects of the distribution type and its variance on the speed and performance of training [23], while Colla *et al.* discussed using the orthogonal least squares method for selecting weights for the processing elements (PEs) from a predetermined library according to the output mean-square-error (MSE) [24]. Nguyen and Widrow proposed assigning each hidden PE to an approximate portion of the range of the desired response [25]. Drago and Ridella, on the other hand, proposed the statistically controlled activation weight initialization, which aimed to prevent PEs from saturating during adaptation by estimating the maximum values that the weights should take initially [26]. There

Manuscript received June 10, 2003; revised March 19, 2004. This work was supported in part by the National Science Foundation under Grant ECS-0300340. The work of D. Erdogmus and O. Fontenla-Romero was supported by the Predoctoral Grants Program of the X. de Galicia, Spain.

D. Erdogmus is with the Department of Computer Science and Engineering, Oregon Graduate Institute, Oregon Health Sciences University, Portland, OR 97006 USA (e-mail: derdogmus@ieee.org).

O. Fontenla-Romero and A. Alonso-Betanzos are with the Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA), Department of Computer Science, University of A Coruna, 15071 A Coruna, Spain (e-mail: oscarfon@mail2.udc.es; ciamparo@udc.es).

J. C. Principe is with the Computational NeuroEngineering Laboratory (CNEL), Electrical and Computer Engineering Department, University of Florida, Gainesville, FL 32611 USA (e-mail: principe@cnel.ufl.edu).

E. Castillo is with the Department of Applied Mathematics and Computational Sciences, University of Cantabria, 39005 Santander, Spain (e-mail: castie@unican.es).

Digital Object Identifier 10.1109/TNN.2004.841777

have been heuristic proposals of least-squares initialization and training approaches [27]–[29], which did not rigorously consider the transformation of desired output through the layers.[1] Castillo *et al.* also proposed a linear least squares training approach for single-layer nonlinear neural networks; the nonlinearity of the neurons were also adapted besides the weights in order to optimize performance [30]. Along these lines, Cho and Chow proposed optimizing the output layer using linear least squares (assuming a linear output layer) and optimizing the preceding layers using gradient descent [31].

In this paper, we propose a new initialization algorithm that is based on a simple linear approximation to the nonlinear solution that can be computed analytically with linear least squares. We reason in the following way for the single hidden layer MLP (this reasoning can be extended to two hidden layer MLPs also). The single-hidden-layer MLP is approximating the solution for the regression or classification problem by computing at the same time the best adaptive basis functions and the best projection on them. In this MLP, the outputs of the hidden layer neurons could be regarded as the adaptive basis functions $\varphi_i(\mathbf{x})$, where i is the index running over the hidden PEs and $\mathbf{x}$ is the input vector to the MLP. The first layer weights are then the parameters of the adaptive basis functions. The second layer weights then can be regarded as the projection coefficients of the desired response to the nonlinear manifold spanned by these basis functions in a multidimensional space. The difficulty arises because both the basis functions and the projection must be optimized at the same time, and the projection manifold is nonlinear, which means that local optimum solutions exist. Motivated by a recent paper by Castillo *et al.* [30], we studied a linear approximation methodology to approximate both the basis functions and the best projection. Castillo *et al.* considered the approximate least squares optimization of a postnonlinear network of single layer weights (no-hidden layer), assuming the MSE before the nonlinearity as the criterion rather than the MSE after the nonlinearity. The desired response at the output of the nonlinearities is simply converted to a desired response at the input of the nonlinearities by utilizing the inverse of the sigmoidal nonlinearities. In this paper, we extend the approach to networks consisting of multiple layers of weights (MLPs). In addition, we improve the optimization accuracy by further considering the local scaling effects of the nonlinearities at the operating points dictated by the current value of the desired outputs.

The idea is to reflect the desired response at the output of the nonlinearity of the PEs back to the input of the nonlinearity of the PEs [30], but also considering the slope of the nonlinearity at the current value of the desired signal), and then solve analytically a linear regression problem for each layer of weights. The proposed algorithm sweeps the layers from the first to the last (optimizing first the basis functions and then the projection coefficients), however, the opposite sweep direction could also be utilized. Hence, the name of this methodology is BP of the desired response.

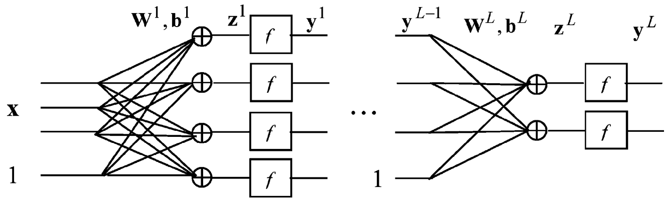


Fig. 1. Assumed MLP structure and the designated variable names.

The organization of this paper is as follows. First, we establish the rules for backpropagating the desired response through nonlinearities and linear layers of weights, since these are the two fundamental building blocks of MLPs. Once the desired response is backpropagated through the layers, it is necessary to optimize each layer using its corresponding input and desired output samples. The linear least squares problem that is required to be solved is mathematically expressed and the solution, which is in the form of a linear system of equations, is derived. Then the algorithm for initializing a two-layer MLP is presented. A section discussing various key issues on the successful application of the algorithm to practical problems of different types follows the algorithm description. Finally, we investigate the performance of the proposed BP of the desired response approach on benchmark problems including a classification problem (this represents discrete-valued desired outputs), two real-world chaotic time-series prediction examples and a realistic nonlinear system identification problem (these represent continuous-valued desired outputs).

II. Backpropagating the Desired Signal Throughthe Layers

Considering the MLP architecture shown in Fig. 1, we notice that there are two parts to backpropagating the desired signal through the layers. Each layer consists of a linear weight matrix and an additive bias term followed by a preselected nonlinear mapping, usually chosen to be a sigmoid-type function.[2] For the successful operation of the proposed algorithm, it is crucial that the nonlinearity is monotonically increasing, but the range does not have to be the whole real number set (the sigmoid-type nonlinearities satisfy this condition and will be used in the numerical examples). In the following, we designate the output of the lth layer of an MLP by $\mathbf{z}^l$ before the nonlinearity and $\mathbf{y}^l$ after the nonlinearity. The weight matrix of each layer is $\mathbf{W}^l$ and the bias vector is $\mathbf{b}^l$. The input vector to the MLP is $\mathbf{x}$. For future convenience, n_l is the number of neurons in the corresponding layer and n_0 is the number of inputs (including the bias inputs). Also, we let N be the number of training samples given in the form $(\mathbf{x}_t, \mathbf{d}_t^L)$, where t is the sample vector index and L is the number of layers in the MLP. In addition, we will denote by $\mathbf{d}^l$ the desired response for the output of the lth layer after the nonlinearity and by $\overline{\mathbf{d}}^l$ the desired response for the output of this layer before the nonlinearity. The overall goal of this process is to generate weight matrices $\mathbf{W}^l$ and $\mathbf{b}^l$ that lead to small

[1]Of special interest is [29], where three least-squares initialization schemes were compared for speed and performance. However, all three approaches ignored the scaling effects of the sigmoidal nonlinearities' slopes in the least squares problem, which is important as we will discuss later.

[2]The bias vector of each layer can be incorporated into the weight matrix as an additional column since it is equivalently the weight of a constant unit input from the previous layer. However, for explanation convenience in the following sections, we will continue to treat it as a separate parameter.

output MSE over the training data. The MSE, which is possibly weighted across outputs, is given by $E[(\mathbf{d}^L-\mathbf{y}^L)^T\mathbf{H}(\mathbf{d}^L-\mathbf{y}^L)]$ and the expectation is approximated by sample mean over the training samples.[3] Two points are important to mention at this point are.

1) In most cases, the optimality criterion is unweighted MSE, thus, the weighting factor $\mathbf{H}$ is identity. However, for the sake of completeness we will include this weighting factor in the formulations.
2) The goal of the initialization algorithm is to provide weights with as small MSE as possible on the training set, so that a possible further training with iterative algorithms (like BP and Levenberg–Marquardt) require minimal number of updates to converge.

The latter means generalization and overfitting issues are not explicitly considered in the presented algorithm due to the following reasons: this issue can be handled in the subsequent iterative updates via a cross-validation set, and given a sufficiently large training set (compared to the size of the network being trained) overfitting will not become a problem. Nevertheless, in order to demonstrate successful generalization of the initialization solution to novel test data, numerical results will also be presented.

A. Backpropagating the Desired Response Through the Nonlinearity

Consider a single-layer nonlinear network for which the equations $\mathbf{z}=\mathbf{W}\mathbf{x}+\mathbf{b}$ and $\mathbf{y}=\mathbf{f}(\mathbf{z})$ define the forward flow of signals, where $\mathbf{W}$, $\mathbf{b}$, $\mathbf{f}(.)$ denote the weight matrix, the bias vector, and the output nonlinearity, respectively. Suppose the weighted MSE for a vector output $\mathbf{y}$ is the chosen optimization criterion and $\mathbf{d}$ is the desired response. Let $\mathbf{H}$ be the weighting factor as described previously. Then Rule 1 describes the BP of the desired response through the output nonlinearity $\mathbf{f}(.)$. The output nonlinearity $\mathbf{f}:\Re^n\rightarrow\Re^n$ consists of elementwise application of the same sigmoidal nonlinear function $f(.)$ as shown in Fig. 1.

Rule 1: Let $\mathbf{d},\mathbf{y},\mathbf{z},\overline{\mathbf{d}}\in\Re^n$ be the desired and actual outputs after the nonlinearity and before the nonlinearity, $\mathbf{W}\in\Re^{n\times m},\mathbf{b}\in\Re^{n\times 1}$ be the weight matrix and the bias vector, and $f,f^{-1},f':\Re\rightarrow\Re$ be the (sigmoid) nonlinearity, its inverse, and its derivative. Also assume that $\mathbf{d}$ is in the range of $\mathbf{f}$. Minimization of the weighted MSE between $\mathbf{d}$ and $\mathbf{y}$ at the output of the nonlinearity is equivalent (up to first order) to minimizing a weighted MSE between $\mathbf{z}$ and $\overline{\mathbf{d}}=\mathbf{f}^{-1}(\mathbf{d})$, where the inverse function is evaluated at each entry separately.[4] In the latter, each error sample is also weighted according to the value of the derivative of the nonlinearity at the corresponding operating point. Mathematically, this is given by

$$\min_{\mathbf{W},\mathbf{b}} E[(\mathbf{d}-\mathbf{y})^T\mathbf{H}(\mathbf{d}-\mathbf{y})]\approx\min_{\mathbf{W},\mathbf{b}} E[(\mathbf{f}'(\overline{\mathbf{d}})\bullet\overline{\boldsymbol{\varepsilon}})^T\mathbf{H}(\mathbf{f}'(\overline{\mathbf{d}})\bullet\overline{\boldsymbol{\varepsilon}})] \tag{1}$$

where $\bullet$ denotes the element-wise product of the vectors $\mathbf{f}'(\overline{\mathbf{d}})$ and $\overline{\boldsymbol{\varepsilon}}=\overline{\mathbf{d}}-\mathbf{z}$.

Derivation: In the Appendix.

Thus, we conclude that, to backpropagate the desired signal through the nonlinearity, we evaluate the inverse of the nonlinear function at the value of the desired signal after the nonlinearity. The weights then must be optimized according to the criterion given in (1). The scaling term $\mathbf{f}'(\overline{\mathbf{d}})$ ensures that each error sample corresponding to an input–output pair of the training set is magnified appropriately according to the operating point of the nonlinearity at the corresponding value of the desired signal. This result also agrees with commonsense. For example, when the PE is near its saturation level, the variance caused by an error sample before the nonlinearity corresponds to a small variance after the nonlinearity. On the other hand, the variance of the error corresponding to a desired signal operating at a large-slope region of the nonlinearity will be magnified by that slope while passing through the nonlinearity. Note that if $\mathrm{var}\left(\|\overline{\mathbf{d}}\|\right)$ is also small, then since the operating point of the nonlinearity will almost be fixed for all samples, this scaling term becomes unnecessary. All the previous applications of least squares to initialize the weights in MLPs failed to take the variance of $\mathbf{d}$ into account, because they simply reflected the desired response to the input of the nonlinearity. This is a poor approximation and as (1) shows, and the scaling effect of the nonlinearity on the variance of the error before the nonlinearity should be considered in minimizing the MSE at the output of the nonlinearity. In general, for more accurate results one may want to use more terms in the Taylor series expansion, however, this brings in the higher order moments of the error, which prevents us from using the linear least squares fitting algorithms for training.

B. Backpropagating the Desired Signal Through the Linear Weight Layer

Consider a linear layer whose output is given by $\mathbf{z}=\mathbf{W}\mathbf{x}+\mathbf{b}$ and the desired signal $\overline{\mathbf{d}}$ is given for $\mathbf{z}$. In this scheme, we assume the weights $\mathbf{W}$ and $\mathbf{b}$ are fixed, but the input vector $\mathbf{x}$ is the free optimization variable. In the MLP context, $\mathbf{x}$ will correspond to the output (after the nonlinearity) of the previous layer. The previous layers will produce output vectors only in a bounded range due to the limited range of the sigmoidal nonlinearities, however, for now this limitation will be ignored. Consequently, during the BP of the desired signal of $\mathbf{z}$ to a desired signal for $\mathbf{x}$, only the weighting factor of the MSE criterion will change. The result is summarized in Rule 2.

Rule 2: Let $\overline{\mathbf{d}},\mathbf{z}\in\Re^n$ be the desired output and the actual output. Let $\mathbf{d},\mathbf{x}\in\Re^m$ be the desired input and the actual input. Let $\mathbf{W}\in\Re^{n\times m},\mathbf{b}\in\Re^{n\times 1}$ be the fixed weight matrix and the bias vector. Then, the optimal input $\mathbf{x}$ that minimizes the weighted MSE between $\overline{\mathbf{d}}$ and $\mathbf{z}$ is the input $\mathbf{x}$ that minimizes

[3]For notational simplicity, the expectation operator $E[.]$ is sometimes used to denote averaging over samples. In this particular case

$$E[(\mathbf{d}^L-\mathbf{y}^L)^T\mathbf{H}(\mathbf{d}^L-\mathbf{y}^L)]\equiv\left(\frac{1}{N}\right)\Sigma_{k=1}^N(\mathbf{d}_k^L-\mathbf{y}_k^L)^T\mathbf{H}(\mathbf{d}_k^L-\mathbf{y}_k^L).$$

[4]Note that without loss of generality, we assume that the desired output at the output of the nonlinearity is in the range of the sigmoid nonlinearity so that the inverse exists. This could be achieved easily by a linear shift-and-scale operation on the desired output signal. In an MLP, this shift-and-scale operation can be accomplished by modifying the weight matrix and the bias vector of the following layer accordingly. We will discuss this issue later in the light of the complete algorithm for multiple layers.

a modified weighted MSE criterion between itself and the best solution to $\mathbf{Wd}+\mathbf{b}=\overline{\mathbf{d}}$ the least squares sense. That is

$$\min_{\mathbf{x}} E[(\overline{\mathbf{d}}-\mathbf{z})^T\mathbf{H}(\overline{\mathbf{d}}-\mathbf{z})] = \min_{\mathbf{x}} E[(\mathbf{d}-\mathbf{x})^T\mathbf{W}^T\mathbf{HW}(\mathbf{d}-\mathbf{x})] \tag{2}$$

where $\mathbf{d}$ is the least square solution to the equality $\mathbf{Wd}=(\overline{\mathbf{d}}-\mathbf{b})$ taking $\mathbf{H}$ into account if necessary.

Derivation: In the Appendix.

There are two situations that require special attention.

1) If $n \geq m$ (more outputs than inputs), then $\mathbf{d}=(\mathbf{W}^T\mathbf{HW})^{-1}\mathbf{W}^T\mathbf{H}(\overline{\mathbf{d}}-\mathbf{b})$ is the unique least squares solution for the over-determined system of linear equations ($\mathbf{Wd}=(\overline{\mathbf{d}}-\mathbf{b})$ with weighting factor $\mathbf{H}$ also taken into account). In an MLP, this corresponds to the situation where the layer has more hidden PEs at the output side than at the input side.
2) If $n < m$, then the QR factorization may be used to determine the minimum norm least squares solution for this underdetermined system of linear equations ($\mathbf{Wd}=(\overline{\mathbf{d}}-\mathbf{b})$) [32]. Since in this underdetermined case there are infinitely many solutions that yield zero error to the equation, the factor $\mathbf{H}$ becomes redundant and need not be considered. In this case the procedure is to obtain $\mathbf{Q}$ and $\mathbf{R}$ by a QR decomposition of $\mathbf{W}$ and then calculate $\mathbf{d}=\mathbf{Q}[\mathbf{R}^{-T}(\overline{\mathbf{d}}-\mathbf{b})\quad \mathbf{0}^T]^T$, where ${}^{-T}$ denotes the matrix inverse-transpose operation and $\mathbf{0}$ is a vector of zeros with length $m-n$.

In both cases, this result tells us that, given a desired signal $\overline{\mathbf{d}}^l$ for the linear output $\mathbf{z}^l$ of the lth layer, we can translate this signal as a desired signal $\mathbf{d}^{l-1}$ for the nonlinear output of the previous layer in the MLP structure. The latter value can then be backpropagated through the nonlinearity as described in Rule 1.

C. Scaling the Desired Output Down to the Range of the Nonlinearity

An important issue that we have ignored until now is the bounded range of the sigmoidal nonlinearities used in MLPs. Consequently, the domain of the inverses of these nonlinearities are also bounded, thus, they are not defined everywhere on the real axis. When the desired response is backpropagated through a linear layer of weights in accordance with Rule 2, the resulting values may be amplified to values that exceed the range of the nonlinearity of the previous layer (due to the matrix inversions involved in computing $\mathbf{d}$). Hence, translating these desired response values through the nonlinearity according to Rule 1 is impossible. However, to overcome this problem, the desired value could be linearly shifted and scaled to fit the range of these nonlinearities.[5] Considering the lth layer of an MLP, this means, when the desired output of the lth layer ($\mathbf{d}^l$) is backpropagated to the output of the $(l-1)$th layer, the desired values for the $(l-1)$th layer's nonlinear outputs ($\mathbf{d}^{l-1}$) could end up outside the range of the specific nonlinear function. In this case, we suggest utilizing a linear shift-scale operation to obtain $\mathbf{d}^{l-1} \leftarrow \mathbf{Td}^{l-1}+\mathbf{s}$ (where $\mathbf{T}$ is an invertible matrix) such that each entry of the new scaled value of $\mathbf{d}^{l-1}$ is inside the range of the nonlinearity. Then to get the same output from the network, should the first layer be optimized according to the new value of $\mathbf{d}^{l-1}$, the lth layer weights need to be modified as follows: $\mathbf{W}^l \leftarrow \mathbf{W}^l\mathbf{T}^{-1}$ and $\mathbf{b}^l \leftarrow -\mathbf{W}^l\mathbf{T}^{-1}\mathbf{s}+\mathbf{b}^l$. For simplicity, $\mathbf{T}$ can be selected to be a diagonal matrix with entries adjusted such that the maximum value of the unscaled value of $\mathbf{d}^{l-1}$ is mapped to a value less than the maximum value of the range of the nonlinearity.

A simple transform that works is found by setting $\mathbf{T}$ to a diagonal matrix with ones on all diagonal entries except for those that correspond to rows of $\mathbf{d}^1$ that have values exceeding the range of the nonlinearities. These entries of $\mathbf{T}$ can be set to the inverse of the dynamic range of the corresponding row of $\mathbf{d}^1$. The shift vector $\mathbf{s}$ can be set to all zeros most of the time, except when a row of $\mathbf{d}^1$ is constant throughout the training set. In this case, the corresponding entry of $\mathbf{s}$ is set to the negative of this constant value that the mentioned row of $\mathbf{d}^1$ has.

III. Solving for the Optimal Weights Using Least Squares

Once the desired signal at the output layer is backpropagated according to the procedures outlined in the previous section, the remaining task is to solve for the optimal weights of each layer using linear least squares.[6] The optimization of the weight matrix of a certain layer, however, must be done immediately after each least square, since the BP of the desired signal through the layers introduce couplings between individual outputs of layer through the weighted MSE problem that arises due to the weight matrices of the following layers (notice in Rule 2 that the weighting factor for MSE changes from $\mathbf{H}$ to $\mathbf{W}^T\mathbf{HW}$). In addition, the BP through nonlinearities introduces a scale factor for each error sample that depends on the slope of the nonlinearity of the layer for that sample (notice in Rule 1 that $\mathbf{f}'(\overline{\mathbf{d}})$ is introduced into the least squares problem). These considerations can be summarized in the following optimization problem.

A. Problem 1

We are given a linear layer of the form $\mathbf{z}=\mathbf{Wx}+\mathbf{b}, \mathbf{W}\in\Re^{n\times m}, \mathbf{b}\in\Re^{n\times 1}$ with the training samples $(\mathbf{x}_s,\overline{\mathbf{d}}_s)$ $s=1,\ldots,N$, where s is the sample index, and also a matrix $\mathbf{G}=[\gamma_{ij}]$ as the weighting factor of MSE (here $\mathbf{G}$ is used to represent a generic weighting factor, which could be $\mathbf{H}$ or $\mathbf{W}^T\mathbf{HW}$ according to the previous notation). We define the error for every sample of the training data and for every entry of the output vector as

$$\overline{\boldsymbol{\varepsilon}}_s=\overline{\mathbf{d}}_s-\mathbf{z}_s \quad s=1,\ldots,N \tag{3}$$

where each output entry is evaluated using

$$\mathbf{z}_s=\mathbf{b}+\mathbf{Wx}_s \quad s=1,\ldots,N. \tag{4}$$

[5]An alternative solution that is observed to work in all simulations we tried is to extend the inverse of the nonlinear function periodically. In particular, for the arctangent nonlinearity, the periodic tangent function becomes a suitable periodic inverse automatically. However, the mechanism behind this solution is not fully understood at this time. Therefore, we will not discuss this possibility further.

[6]Although these weights are referred to as optimal, they are only optimal in the least square sense after all the mentioned approximations. They are not necessarily the true optimal weights of the MLP in the grand scheme.

The optimal weights for this layer according to the arguments presented in Rule 1 and Rule 2 are the solutions to the following minimization problem:

$$\min_{\mathbf{W},\mathbf{b}} J = \frac{1}{N}\sum_{s=1}^{N}(\mathbf{f}'(\overline{\mathbf{d}}_s)\bullet\bar{\boldsymbol{\varepsilon}}_s)^T\mathbf{G}(\mathbf{f}'(\overline{\mathbf{d}}_s)\bullet\bar{\boldsymbol{\varepsilon}}_s). \tag{5}$$

Solution: In order to solve for the optimal weights for Problem 1, we take the derivatives of the cost function with respect to the weights of the system and equate to zero. Since this is a quadratic cost function in terms of the optimization parameters, there is a unique global optimum. The solution for the case of an arbitrary symmetric positive definite weighting factor $\mathbf{G}$ is quite cumbersome and its derivation is provided in the Appendix for the interested readers. As will be seen in the experimental section, generally assuming that $\mathbf{G} = \mathbf{I}$ provides extremely satisfactory results. Therefore, here we will only consider this special case, for which the solution is also simple to obtain and express. If $\mathbf{G} = \mathbf{I}$, then the overall least squares problem decomposes to multiple least squares problems (one for each output). Specifically, the cost function in (5) becomes

$$\min_{\mathbf{W},\mathbf{b}} J = \sum_{j=1}^{n}\frac{1}{N}\sum_{s=1}^{N}(f'(\bar{d}_{js})\bar{\varepsilon}_{js})^2 \tag{6}$$

where the subscript j runs over the outputs ($j = 1,\ldots,n$) and the elementwise product reduces to standard multiplication of scalars. Defining the column vector $\mathbf{w}_j = [b_j\mathbf{W}_{j:}]^T$ using the jth entry of $\mathbf{b}$ and the jth row of $\mathbf{W}$ and $\mathbf{u} = [1\ \mathbf{x}^T]^T$, the modified input vector, we have the following expression for the error at the jth output channel: $\bar{\varepsilon}_{js} = d_{js} - \mathbf{u}_s^T\mathbf{w}_j$. The gradient of (6) with respect to a particular $\mathbf{w}_k$ is

$$\frac{\partial J}{\partial \mathbf{w}_k} = \frac{-2}{N}\sum_{s=1}^{N}[f'^2(\bar{d}_{ks})(d_{ks} - \mathbf{u}_s^T\mathbf{w}_k)\mathbf{u}_s]\quad k = 1,\ldots,n. \tag{7}$$

Each $\mathbf{w}_k$ can be solved independently by equating the corresponding gradient given in (7) to zero. The solution is a slightly modified version of the Wiener-Hopf equation, one that takes the local scaling effects of the nonlinearity into account

$$\left[\sum_{s=1}^{N} f'^2(\bar{d}_{js})d_{js}\mathbf{u}_s\right] = \left[\sum_{s=1}^{N} f'^2(\bar{d}_{js})\mathbf{u}_s\mathbf{u}_s^T\right]\mathbf{w}_k\quad k = 1,\ldots,n. \tag{8}$$

IV. Training Algorithm for a Single Hidden Layer MLP

In this section, we will describe how to apply the procedures devised in the preceding sections to the training of a two-layer MLP (one hidden layer). Although we restrict this algorithm to only two layers here, the idea can be generalized easily to MLPs with more layers. However, it is known that an MLP with only a single hidden layer that contains a sufficiently large number of hidden PEs can approximate any continuous-differentiable function [33], [34], thus, this topology is sufficient, in general.

Consider a two-layer MLP with n_0 inputs, n_1 hidden PEs, and n_2 output PEs. For the sake of generality, we also assume that the output layer contains nonlinearities. We denote the weight matrix and the bias vector of layer l with $\mathbf{W}^l$ and $\mathbf{b}^l$, respectively, where $l = 1,2$. The output vectors of this layer before and after the nonlinearity are denoted by $\mathbf{z}^l$ and $\mathbf{y}^l$, and $\overline{\mathbf{d}}^l$ and $\mathbf{d}^l$ denote the desired signals for these outputs, in the same order. Assuming that unweighted MSE is the optimization criterion ($\mathbf{H} = \mathbf{I}$), the proposed initialization algorithm for this MLP follows the following procedure:

Initialization: Obtain training data in the form $\mathbf{x}_s, \mathbf{d}_s^2$, $s = 1,\ldots,N$. Select random initial values for the weights $\mathbf{W}^1$, $\mathbf{b}^1$, $\mathbf{W}^2$, $\mathbf{b}^2$. Evaluate $\mathbf{z}_s^1, \mathbf{y}_s^1, \mathbf{z}_s^2, \mathbf{y}_s^2$ corresponding to the input sample $\mathbf{x}_s$ using these randomly selected weights. Set $J_{\text{opt}} = E[(\mathbf{d}^2 - \mathbf{y}^2)^T(\mathbf{d}^2 - \mathbf{y}^2)]$. Also assign current weights as $\mathbf{W}_{\text{opt}}^1 = \mathbf{W}^1, \mathbf{b}_{\text{opt}}^1 = \mathbf{b}^1, \mathbf{W}_{\text{opt}}^2 = \mathbf{W}^2, \mathbf{b}_{\text{opt}}^2 = \mathbf{b}^2$.

Step 1) Compute $\overline{\mathbf{d}}_s^2 = f^{-1}(\mathbf{d}_s^2)$, $\forall s$ as the desired signal for $\mathbf{z}_s^2$.

Step 2) If $n_2 \geq n_1$, then assign $\mathbf{d}_s^1 = (\mathbf{W}^{2T}\mathbf{W}^2)^{-1}\mathbf{W}^{2T}(\overline{\mathbf{d}}_s^2 - \mathbf{b}^2)$ as the desired signal for $\mathbf{y}_s^1$.

If $n_2 < n_1$, then obtain $\mathbf{Q}$, $\mathbf{R}$ by a QR decomposition of $\mathbf{W}^2$ and assign $\mathbf{d}_s^1 = \mathbf{Q}[\mathbf{R}^{-T}(\overline{\mathbf{d}}_s^2 - \mathbf{b}^2)\quad \mathbf{0}^T]^T$, where $\mathbf{0}$ is a vector of zeros of length $n_1 - n_2$, as the desired signal for $\mathbf{y}_s^1$.

In any case, if there exist $\mathbf{d}_s^1$ values that are outside the range of the nonlinearity of PEs, employ the scale-shift technique to bring these values into the range. (In this forward sweep approach modifying $\mathbf{W}^2$ and $\mathbf{b}^2$ accordingly is not necessary since they will be optimized later in the procedure anyway.)

Step 3) Compute $\overline{\mathbf{d}}_s^1 = f^{-1}(\mathbf{d}_s^1)$, $\forall s$ as the desired signal for $\mathbf{z}_s^1$.

Step 4) Optimize $\mathbf{W}^1$ and $\mathbf{b}^1$ using the linear least squares equations in (8), using $\mathbf{x}_s$ as input samples and $\overline{\mathbf{d}}_s^1$ as desired output samples. Here, we assume $\mathbf{G} = \mathbf{I}$ for simplicity.[7]

Step 5) Evaluate $\mathbf{z}_s^1$, $\mathbf{y}_s^1$ using the new values of first layer weights.

Step 6) Optimize $\mathbf{W}^2$ and $\mathbf{b}^2$ using the linear least squares equations in (8), using $\mathbf{y}_s^1$ as input samples and $\overline{\mathbf{d}}_s^2$ as desired output samples.

Step 7) Evaluate $\mathbf{z}_s^2$, $\mathbf{y}_s^2$ using the new values of second-layer weights.

Step 8) Evaluate the value of $J = E[(\mathbf{d}^2 - \mathbf{y}^2)^T(\mathbf{d}^2 - \mathbf{y}^2)]$, with the new output samples. If $J < J_{\text{opt}}$, set $J_{\text{opt}} = J$, and $W_{\text{opt}}^1 = W^1, b_{\text{opt}}^1 = b^1, W_{\text{opt}}^{21} = W^2, b_{\text{opt}}^2 = b^2$.

Step 9) Go back to Step 2).

The algorithm presented here prefers backpropagating the desired signal all the way to the first layer and then optimizes the weights of the layers sweeping them from the first to the last. Alternatively, first the last layer weights may be optimized, then the desired signal can be backpropagated through that layer using the optimized values of the weights, and so on. Thus, in this alternative algorithm, the layers are optimized sweeping them from the last to the first. In this alternative approach, in the steps where the backpropagated desired output is scaled down

[7]According to Rule 2, we should have used $\mathbf{G} = \mathbf{W}^{2T}\mathbf{W}^2$. However, extensive experiments not presented here showed that this choice is not critical to the performance of the algorithm. The simpler choice of $\mathbf{G} = \mathbf{I}$ gived satisfactory results in all experiments.

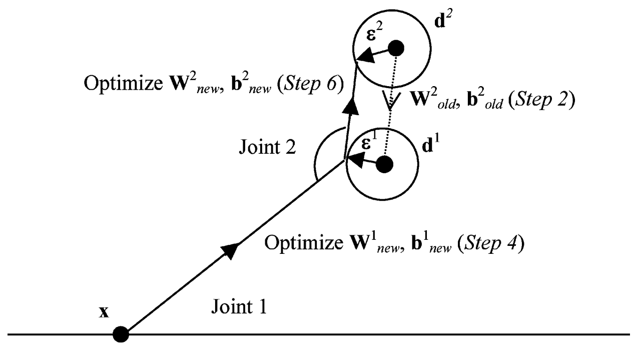

ε^2
$\mathbf{d}^2$
Optimize $\mathbf{W}^2_{new}$, $\mathbf{b}^2_{new}$ (*Step 6*)
$\mathbf{W}^2_{old}$, $\mathbf{b}^2_{old}$ (*Step 2*)
Joint 2
ε^1
$\mathbf{d}^1$
Optimize $\mathbf{W}^1_{new}$, $\mathbf{b}^1_{new}$ (*Step 4*)
x
Joint 1

Fig. 2. Illustration of the robot arm analogy for the operation of the algorithm.

to the range of the nonlinearities (such as Step 2) the weights of the following layer should be modified accordingly, as opposed to the previous algorithm where this modification is not necessary. Extensive experiments with both approaches yielded similar results.

Finally, once this algorithm is iterated a number of times (two to five is usually sufficient), the weight values that correspond to the smallest MSE error that is encountered are stored and can be assigned as initial conditions to a standard iterative descent algorithm. Since the algorithm does not explicitly minimize the MSE, in applications where the designer aims to obtain the optimal weights as accurately as possible, the postapplication of a standard descent algorithm may be necessary. In many applications, however, the aim is to obtain a sufficiently accurate solution in a short amount of time even if this solution is not optimal. For example, in an online control scenario that employs adaptive MLPs for various functions (controller, plant model etc.), it is not always necessary to obtain the actual optimal solution; rather, it is desired to achieve a preset level of modeling accuracy as quickly as possible. The algorithm outlined previously can achieve this task in a few iterations with little computational requirements.

In order to understand the operation and the behavior of the algorithm better (for the two-layer MLP case), consider an analogy to a robotic arm with two joints (depicted in Fig. 2). This analogy is a mechanical equivalent of adapting the bases and the projections at the same time in a two-layer MLP. In this analogy, the weights of the two layers become the angles of the two joints of the robot arm, whose one end is fixed at a given point (a given sample of the input vector) and the other end is trying to reach a desired position in space (the corresponding desired response for the network). The fixed arm lengths signify the limited choice of basis functions and the bounded span of these bases. At each iteration, the robot arm first evaluates the desired position for Joint 1, depending on the current value of Joint 2 (backpropagates the desired response through layers). Then it moves Joint 1 (first-layer weights) to bring the first arm as close as possible to its desired location (backpropagated desired response for the first layer). Finally, it moves Joint 2 (second-layer weights) to bring the second arm as close as possible to the desired position (actual desired network response).

V. Discussion of Algorithmic Issues

There are a few issues about this initialization algorithm to be noted. The most important one is why this algorithm is referred to as an initialization procedure and not as a self-contained training methodology. It is clear from the derivation and from the algorithm itself that the individual iterations of the procedure do not explicitly impose the reduction of MSE compared to the preceding iteration. In fact, it was observed in numerous simulations that although the algorithm achieves a very small MSE value (very close to global optimum) at certain iterations, it does not necessarily converge to the global minimum (or any other local minimum for that matter). Therefore, this approach should be utilized as an exceptionally fast way of obtaining a nearly optimal operating point. If the weights at each training stage are stored, then we can use the ones that correspond to the smallest error as the ones to keep (either for the initialization of BP or as the nearly optimal weights).

A second issue related to the previous observation is that the algorithm does not offer a smoothly decreasing value of training MSE through the iterations. In fact, it has been observed that at every sweep, the algorithm jumps from a region in the weight space to a completely different region. This should not be regarded as a weakness, however. On the contrary, this stochastic search is the strength of this method as an effective search algorithm to determine a good set of weights. In fact, from the robot arm analogy we have presented in the previous section, it is possible to see how this algorithm produces many possible projection spaces; only limited by the vast set of possible functions spanned by the MLP.

Therefore the selection of weighting factors in the individual least squares solutions of layers becomes an important concern. Although Rule 2 shows that one should consider the amplifying effect of the following output layer in selecting the MSE weighting factor $\mathbf{G}$, in most cases, our experience showed that the algorithm performs equally well with an identity weighting matrix as suggested in Step 4. The reason for this is, since at a given iteration the second-layer weights are, in general, far from being optimal, considering their effect on the optimization of the first layer is not necessary. In addition, once the first layer is optimized with respect to the unweighted MSE, the second layer can be optimized accordingly (as done by the algorithm in the forward sweep scheme).

Another important aspect is the computational complexity of the proposed initialization algorithm. To assess its complexity, let us focus on the two-layer MLP. Recall from the previous section that the main source of complexity is the solution of the least-squares problems at Step 4 and Step 6. Each requires solving the square linear system of equations in (8). In general, for the lth layer, there are $n_l(1+n_{l-1})$ equations and unknowns. Since the inversion of a size-m square matrix requires $O(m^3)$ calculations, The computational complexity of the algorithm is $\max_l O(n_l^3 n_{l-1}^3)$. In the two-layer MLP case, if $n_0 > n_2$ this is $O(n_1^3 n_0^3)$.[8]

[8]To give an idea, the code running in Matlab 5.3 on a Pentium 4 desktop computer with CPU speed 2.40 GHz, completes one iteration in less than 1 s. The authors did not attempt to optimize the code for speed using advanced numerical algorithms. Typically, the best solution is achieved in the second iteration.

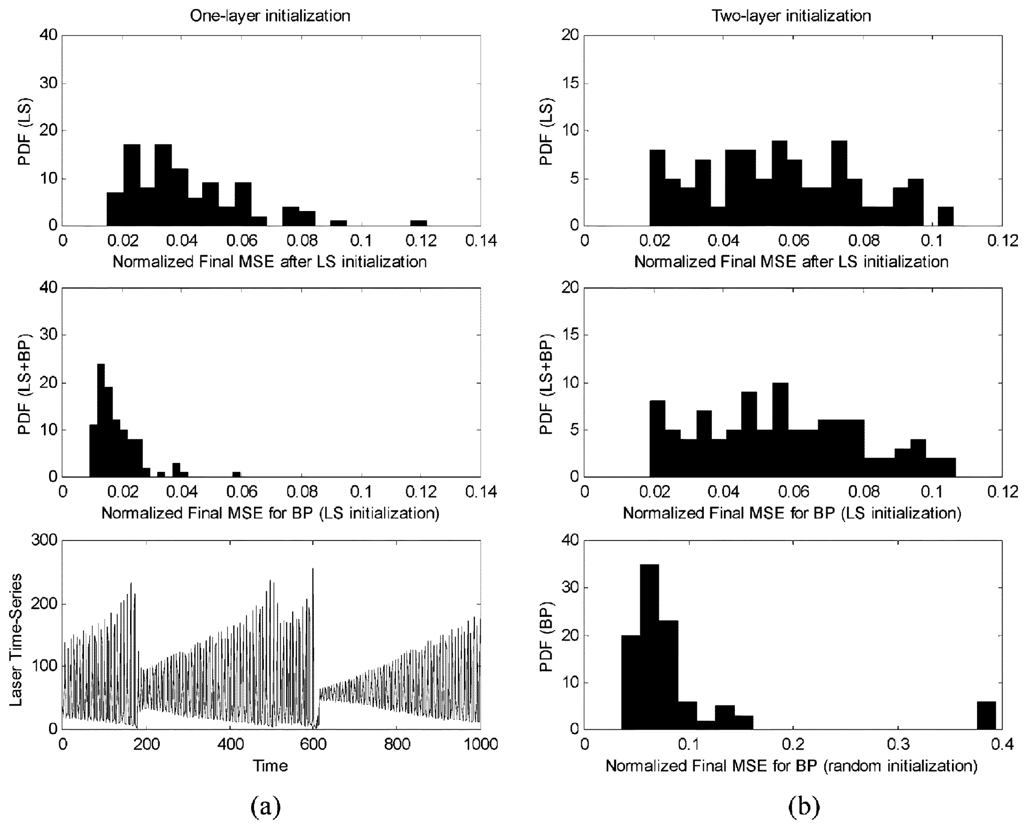


Fig. 3. Histograms of final MSE values for the laser-series starting from 100 random initial weights. (a1) Only the output layer is trained with the proposed least squares approach. (a2) Solutions in (a1) given as initial conditions to the standard BP algorithm. (a3) The training data from the laser-series. (b1) Both layers trained with the proposed least squares approach. (b2) Solutions in (b1) given as initial conditions to the standard BP algorithm. (b3) MLP trained with the standard BP algorithm only.

Finally, after a desired number of iterations are performed, the algorithm might generate initial weight estimates where some parameters are larger than desirable.[9] This is especially important in hardware implementations of neural networks since dynamic range of variables and parameters might be restricted by hardware design. In such cases, the scale factor for the first layer weights can be traded with the slope of the nonlinearity and the scale factor of the second layer (assuming a linear output layer) can be traded with the range of the nonlinearity. In particular, for the single hidden layer MLP under consideration, suppose that the nonlinear functions in the hidden layer PEs are parameterized as $\alpha_1 f(\alpha_2 x)$ where $f(x)$ is a unit-range, unit-slope (at the inflection point) function, such as $f(x) = (2/\pi)\arctan(\pi x/2)$. Then, we can reduce the dynamic range of the first and second-layer weights by dividing them by some values α_1 and α_2 respectively. By modifying the nonlinear function in the hidden PEs according to the prescription given previously, we can make the network output remain unchanged. This weight normalization procedure can be employed after the iterations of the initialization algorithm are completed.

[9]Note that large weight values of an MLP do not necessarily imply any statistical approximation inability of the network. What is important is not to have saturated nonlinearities in response to certain inputs. In all the experiments we performed with this algorithm, we checked the nonlinearity activation values for saturation and the trained networks do not result in saturating nonlinearities.

VI. Monte Carlo Case Studies

In this section, we will present training and testing performances of the proposed BP of the desired signal. Time-series prediction, nonlinear system identification, and pattern classification are benchmark problem types that are usually considered in the literature. Therefore, our case studies include the single-step prediction of two real-world chaotic time-series (the laser time series [35], [36], and the Dow Jones closing index series [36]). In addition, the identification of the realistic nonlinear engine manifold dynamics (based on a car engine [37], [38]) is carried out using an MLP. The engine manifold model assumes the manifold pressure and manifold temperature as the states, and the pressure as the system output. The input is the angle of the throttle that controls the amount of air flowing into the manifold. Finally, to as an illustrative example, the spiral classification problem is also considered. In the following, we summarize the settings used for each data set. For the first three examples, the number of PEs are selected to be the smallest that yield similar training and cross-validation errors. For the spiral example, one PE per exemplar is utilized to guarantee perfect classification on the training set.

- **Laser Time Series:** MLP size 3-11-1, predict next value from most recent three values.
- **Dow Jones Index:** MLP size 5-7-1, predict next value from most recent five values.
- **Engine Identification:** MLP size 4-5-1, estimate output from most recent two inputs and two outputs.

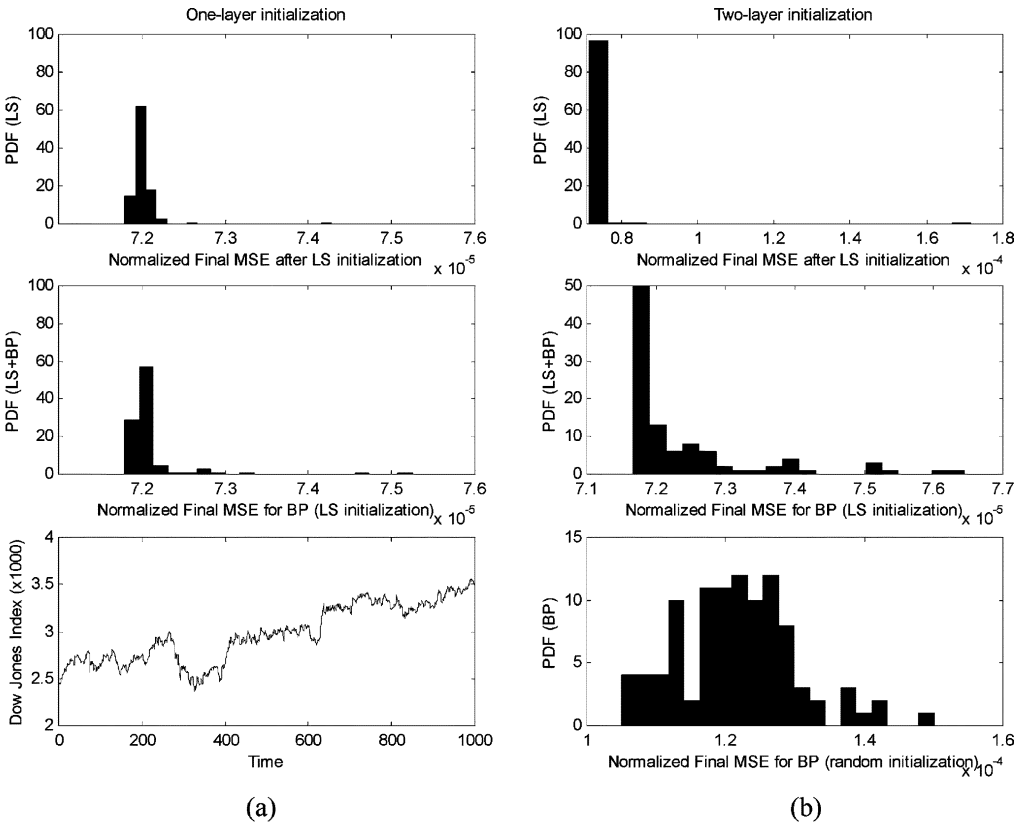


Fig. 4. Histograms of final MSE values for the Dow Jones series starting from 100 random initial weights. (a1) Only the output layer is trained with the proposed least squares approach. (a2) Solutions in (a1) given as initial conditions to the standard BP algorithm. (a3) The training data from the Dow Jones series. (b1) Both layers trained with the proposed least squares approach. (b2) Solutions in (b1) given as initial conditions to the standard BP algorithm. (b3) MLP trained with the standard BP algorithm only.

- **Spiral Classification:** MLP size 2-50-1, determine which of the two spirals the input pattern is from.

All networks have linear output PEs, $\arctan$ nonlinearities, and are trained using 1000 input-desired pairs. Training is repeated starting from 100 random sets of initial weights. The following algorithms are utilized, all using the same data, settings, and initial weights: BP,[10] least-squares initialization of only the second (output) layer according to Steps 5–7 of the proposed algorithm for only one iteration (LS1), initialization of both the first and second layers using the LS-procedure proposed in this paper using Steps 1–9 for three iterations (LS2), and least-squares-initialized-BP algorithms using both results of LS1 and LS2 for initializing the weights (LS1+BP and LS2+BP).

Since the LS-initialization procedure readily brings the weights to the vicinity of the optimal solution, further iterations using BP in the LS+BP approaches improve MSE on the training sets marginally. For the three data sets (Laser, Dow Jones, and Engine), we have allowed BP to train for 1000, 2000, and 200 iterations, respectively. In contrast, we have allowed the LS+BP algorithms to train for 250, 500, and 50 epochs using BP after the least-squares initialization. For all BP phases, MATLAB's Neural Network Toolbox was utilized and these numbers of iterations were determined experimentally to guarantee convergence of the BP algorithm from numerous random initial conditions, prior to the actual Monte Carlo runs that are utilized in the results presented here. The training performance is measured using the normalized MSE measure, which is defined as the ratio of the error power to the power of the desired signal.

[10]In the simulations we provide the solutions obtained with the simple BP rule. There exist more efficient iterative algorithms in the literature that use second-order optimization techniques such as Levenberg–Marquardt and other quasi-Newton methods. However, the main point of these simulations is to demonstrate that the proposed approximate least squares algorithm provides a useful solution (in terms of training and test set MSE levels). Our goal is not to compare the performance of this algorithm with BP or any other more advanced training algorithm, since BP of the desired response algorithm is proposed as a potential initialization for these iterative approaches rather than replacing them.

The results of the laser time series prediction simulations are summarized in Fig. 3. Starting from the preselected 100 random initial weights, the LS1 and LS2 initialization schemes both achieved MSE values in the range [0.02,0.12] [shown in Fig. 3(a1) and (b1)]. Further training with BP [shown in Fig. 3(a2) and (b2)] did not improve MSE significantly. BP approach [shown in Fig. 3(b3)], on the other hand, achieved similar MSE values to those of LS1 and LS2, however, a significant percentage of the solutions were trapped at MSE values larger than 0.1, and some at 0.4. It is noteworthy that the least squares initializations are computationally much simpler compared to BP, yet they still achieve almost optimal MSE values with a few iterations.

The results of the Dow Jones series prediction simulations are summarized in Fig. 4. Similarly, the LS1 and LS2 initialization schemes achieve almost optimal MSE values [shown in Fig. 4(a1) and (b1)]. Once again, further training with BP did not improve MSE performance significantly. Notice that, in

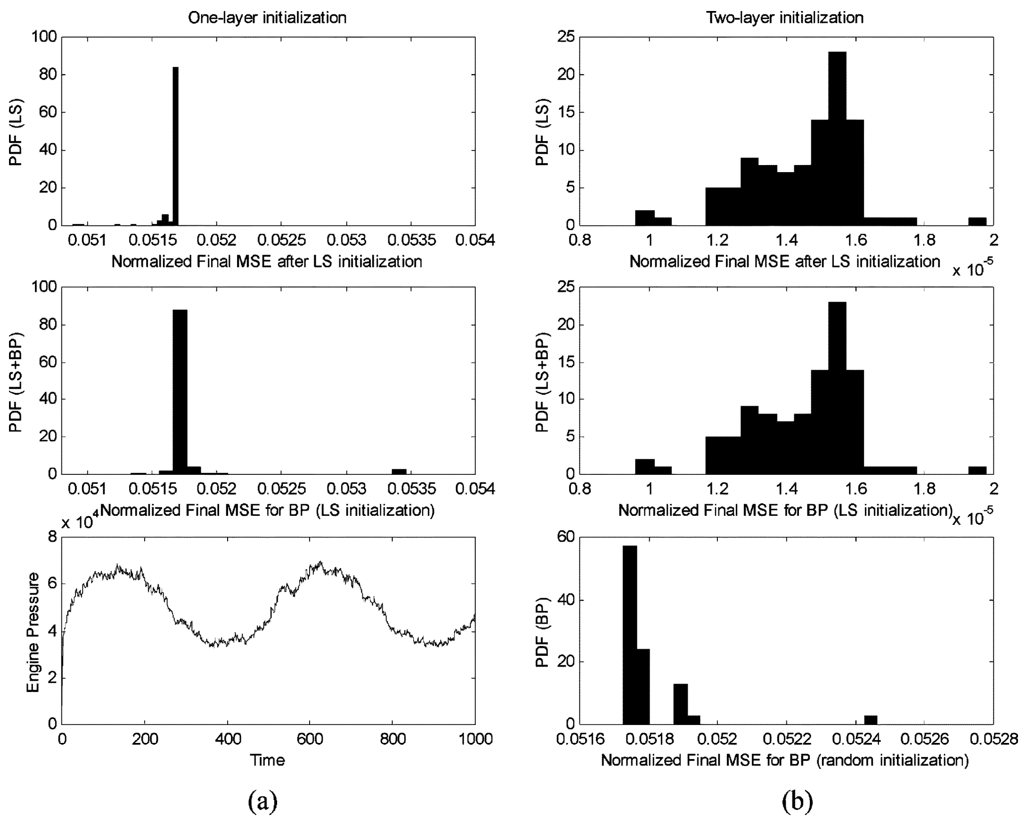


Fig. 5. Histograms of final MSE values for the engine-dynamics-identification starting from 100 random initial weights. (a1) Only the output layer is trained with the proposed least squares approach. (a2) Solutions in (a1) given as initial conditions to the standard BP algorithm. (a3) The desired output for the engine-dynamics-identification problem. (b1) Both layers trained with the proposed least squares approach. (b2) Solutions in (b1) given as initial conditions to the standard BP algorithm. (b3) MLP trained with the standard BP algorithm only.

TABLE I
AVERAGE AND STANDARD DEVIATIONS OF NORMALIZED MSE VALUES ON TRAINING AND TEST DATA SETS OVER 100 MONTE CARLO RUNS WITH RANDOM INITIAL WEIGHTS

Data Set	Training MSE	Testing MSE
Laser	$5.5\text{x}10^{-2} \pm 2.7\ \text{x}10^{-2}$	$5.9\ \text{x}10^{-2} \pm 2.7\text{x}10^{-2}$
Engine	$1.5\text{x}10^{-5} \pm 2.9\text{x}10^{-6}$	$1.6\text{x}10^{-5} \pm 4.1\text{x}10^{-6}$
Dow Jones	$7.3\text{x}10^{-5} \pm 9.4\text{x}10^{-6}$	$2.0\text{x}10^{-1} \pm 3.5\text{x}10^{-1}$

this case, BP-only approach, BP, did not achieve as small MSE values as the least squares initialization schemes. BP was possibly always trapped at nearby local minima.

The first two examples demonstrated the advantages of the proposed initialization scheme over using BP-only for training neural networks. They did not however, reveal any advantages of initializing both layers (LS2) over initializing only the output layer (LS1). This final example demonstrates this advantage. The results of the engine-dynamics-identification simulations are summarized in Fig. 5. Notice that while LS1 achieves an MSE around 5×10^{-2} [shown in Fig. 5(a1)], LS2 brings the MLP weights to a region where the MSE is on the order of 10^{-5} [shown in Fig. 5(b1)]. Once again, further training using BP does not improve significantly over the MSE values attained by the LS1 and LS2 initialization schemes. The BP approach was also trapped at the local minimum, which had an MSE of 5×10^{-2}. These results demonstrate clearly that initializing both layers using the proposed BP of desired response approach is advantageous compared to both using BP alone and initializing only the output layer.

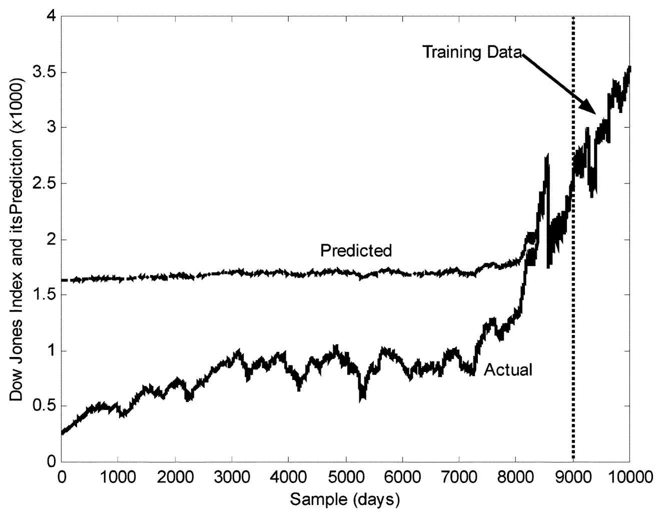


Fig. 6. The prediction performance of a trained network on the test set of Dow Jones data. Normalized training MSE is 7.2×10^{-5} and testing MSE is 3.6×10^{-1}.

Finally, we present results that demonstrate the generalization capability of the solution provided by the proposed algorithm. For this purpose, we trained the MLPs described in the previous experiments using 1000 samples for each of the three data sets:

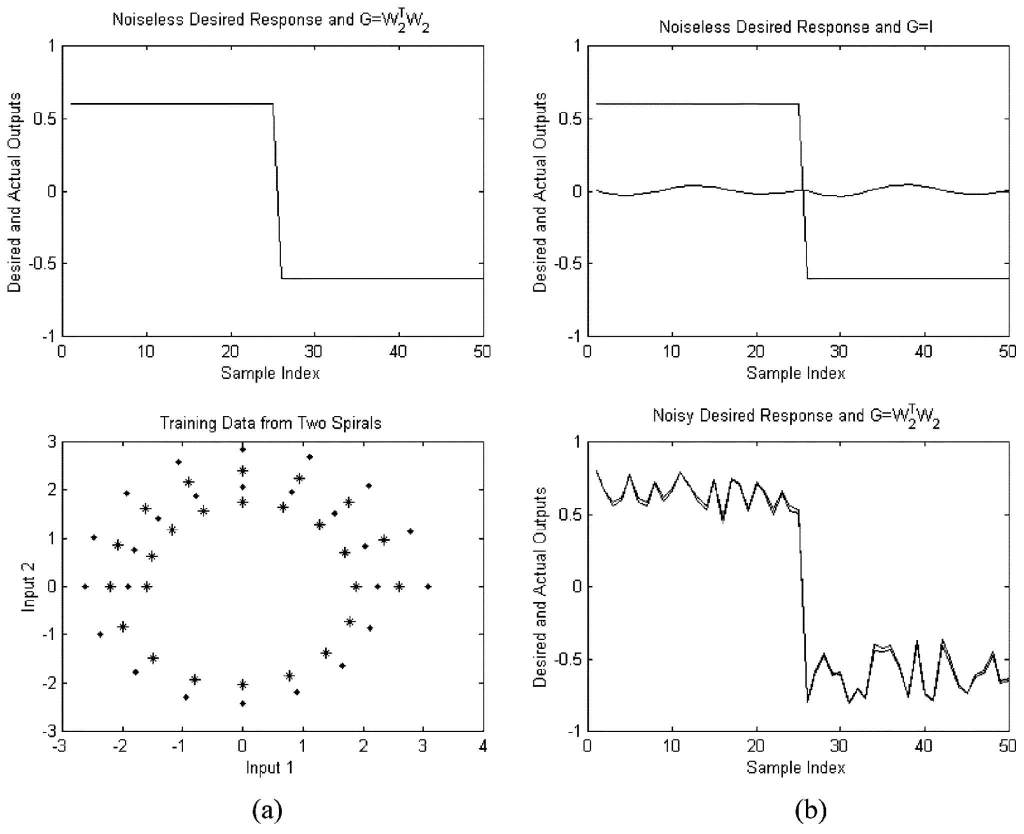


Fig. 7. Spiral classification results (best results obtained in 20 iterations for each method). (a1) Desired and actual classification results for noiseless desired response and theoretically offered $\mathbf{G} = \mathbf{W}_2^T\mathbf{W}_2$ weighted least squares training for the first layer b1) Desired and actual classification results for noiseless desired response and $\mathbf{G} = \mathbf{I}$ unweighted least squares training for the first layer. (b2) Desired and actual classification results for noisy desired response and theoretically offered $\mathbf{G} = \mathbf{W}_2^T\mathbf{W}_2$ weighted least squares training for the first layer.

Laser, Engine, and Dow Jones. Then, the trained networks are tested on their corresponding 10000-sample test sets (which includes the 1000 training samples and 9000 novel samples). The average ±standard deviation of the normalized MSE values for each data set over the 100 Monte Carlo runs is summarized in Table I.

Notice that for the Laser and Engine data sets, the testing errors are only slightly larger than the training errors, which indicates good generalization without overfitting. For the Dow Jones data set, however, the testing MSE is much higher than the training MSE due to the nonstationary nature of this time series. The training set consists of the last 1000 samples (approximately three years from 1991 to 1993) of the 10000-sample (approximately 30 years from 1964 to 1993) test set. A sample prediction performance for the Dow Jones set is presented in Fig. 6. Clearly, starting in the late 1980s, the stock market changed its characteristics, therefore, the network performs well on the data portion that it is trained on, but performs very poorly on the rest.

As the final demonstration, we present training results using the spiral classification example. The spiral classification data is shown in Fig. 7(a2). The hidden layer PEs have arctan nonlinearities with range $[-1, 1]$, so the desired outputs corresponding to the two spirals are selected to be slightly perturbed values around ±0.6. It has been observed that, in classification problems, where the desired responses corresponding to a variety of input patterns are constant (as in this example), the systems of linear system equations that yield the least squares solutions for the weight matrices may become ill-conditioned. In such case, introducing a small additive noise to the desired output values, as suggested in [39], helps to improve the conditioning of these matrices as well as reduce the number of iterations required to get a very good initializing solution. The training performance using such a *noisy* desired output is presented in Fig. 7(b2). The amount of noise, in this example, is purposely selected to be much larger than necessary. It was observed in many trials with different types of data sets that adding a small perturbation to the desired output helps the algorithm achieve its minimum MSE solution in much fewer iterations compared to when the algorithm is employed on data with clean desired output values (such as ±0.6). In this example, the suggested weighting effect of the second layer is utilized in the optimization of the first layer weights. This case study serves as a useful example to demonstrate the importance of Rule 2 in some situations. Although in the previous example, the weighting factor $\mathbf{G}$ of Problem 1 was ignored (by using $\mathbf{G} = \mathbf{I}$), in classification problems it has been observed that proper usage of Rule 2 becomes crucial. For example, for the noise-free desired response of the spiral problem, while the proper approach of $\mathbf{G} = \mathbf{W}_2^T\mathbf{W}_2$ yields very good initialization results, the simplified alternative of $\mathbf{G} = \mathbf{I}$ does not yield a useful initialization solution [Fig. 7(a1) and (b1)]. The same behavior was also observed in other classification problems such as the generalized XOR (also called the parity-bit) problem.[11]

VII. Conclusion

Neural networks have become indispensable tools in many areas of engineering and are continuing to receive much attention in other fields. The well-known BP algorithm and more advanced versions that employ quasi-Newton type iterative updates made it feasible to practically utilize MLPs, a widely used topology from the family of additive networks. Two major drawbacks of iterative learning rules are local minima and dependence on the learning rate. To tackle these problems, most of the time it is necessary to either use advanced search methods and global optimization schemes. However, successful initialization of the optimization parameters can also drastically help improve learning time and performance.

In this paper, we have proposed a methodology for backpropagating the desired response of the MLP through its nonlinear layers, allowing us to train each layer by simply solving a linear system of equations per layer, which is the solution to a linear least squares problem. The proposed algorithm may be utilized as an initialization routine for a following standard BP training scheme or even as a training algorithm itself in some applications in which it is desirable to attain small, but not necessarily optimal, MSE in a fast manner.

The superior training and generalization performances of the proposed initialization scheme were demonstrated in chaotic laser time series prediction, Dow Jones time series prediction, and realistic engine manifold dynamics identification examples. Monte Carlo simulations illustrated that BP of the desired response through the layers is an effective initialization/training scheme for MLPs.

Appendix

A. Derivation of Rule 1

Using the fact that $\mathbf{y}=\mathbf{f}(\mathbf{z})$ and $\mathbf{d}=\mathbf{f}(\overline{\mathbf{d}})$, we can write

$$\begin{aligned}\min_{\mathbf{W},\mathbf{b}} & E[(\mathbf{d}-\mathbf{y})^T\mathbf{H}(\mathbf{d}-\mathbf{y})]\\ &=\min_{\mathbf{W},\mathbf{b}} E[(\mathbf{f}(\overline{\mathbf{d}})-\mathbf{f}(\mathbf{z}))^T\mathbf{H}(\mathbf{f}(\overline{\mathbf{d}})-\mathbf{f}(\mathbf{z}))]. \quad (A.1)\end{aligned}$$

Let $\overline{\mathbf{d}}=\mathbf{f}^{-1}(\mathbf{d})$ be the desired response we seek for $\mathbf{z}$ and $\overline{\boldsymbol{\varepsilon}}=\overline{\mathbf{d}}-\mathbf{z}$, be the error before the nonlinearity. If var $(\|\overline{\boldsymbol{\varepsilon}}\|)$ is small, then we can use the following first-order Taylor series approximation on each component of the output vector

$$\mathbf{f}(\mathbf{z})=\mathbf{f}(\overline{\mathbf{d}}-\overline{\boldsymbol{\varepsilon}})\approx\mathbf{f}(\overline{\mathbf{d}})-\mathbf{f}'(\overline{\mathbf{d}})\bullet\overline{\boldsymbol{\varepsilon}} \quad (A.2)$$

where $\bullet$ denotes the element-wise product of vectors. Substituting this in (A.1), we obtain

$$\min_{\mathbf{W},\mathbf{b}} E[(\mathbf{d}-\mathbf{y})^T\mathbf{H}(\mathbf{d}-\mathbf{y})]\approx\min_{\mathbf{W},\mathbf{b}} E[(\mathbf{f}'(\overline{\mathbf{d}})\bullet\overline{\boldsymbol{\varepsilon}})^T\mathbf{H}(\mathbf{f}'(\overline{\mathbf{d}})\bullet\overline{\boldsymbol{\varepsilon}})] \quad (A.3)$$

which is the result we seek.

B. Derivation of Rule 2

The weighted MSE at the output of the linear layer requires solving

$$\min_{\mathbf{x}} E[(\overline{\mathbf{d}}-\mathbf{z})^T\mathbf{H}(\overline{\mathbf{d}}-\mathbf{z})]=E[(\overline{\mathbf{d}}^T\mathbf{H}\overline{\mathbf{d}}-2\overline{\mathbf{d}}^T\mathbf{H}\mathbf{z}+\mathbf{z}^T\mathbf{H}\mathbf{z})] \quad (A.4)$$

where $\mathbf{H}$ is again the weighting factor of the MSE criterion and $\mathbf{z}=\mathbf{W}\mathbf{x}+\mathbf{b}$. Suppose that $\mathbf{d}$ is the weighted least squares solution of the equation $\mathbf{W}\mathbf{x}+\mathbf{b}=\overline{\mathbf{d}}$ with respect to a weighting factor $\mathbf{H}$. Consequently, let $\mathbf{W}\mathbf{d}=(\overline{\mathbf{d}}-\mathbf{b})+\boldsymbol{\varepsilon}_*$. Let $\boldsymbol{\varepsilon}=\mathbf{d}-\mathbf{x}$, then we have $\mathbf{z}=\overline{\mathbf{d}}+\boldsymbol{\varepsilon}_*-\mathbf{W}\boldsymbol{\varepsilon}$. Substituting the latter in (A.4) and dropping the terms that are constant with respect to the optimization variable $\mathbf{x}$, the following equivalent problem is obtained

$$\begin{aligned}&\min_{\mathbf{x}} E[(\overline{\mathbf{d}}-\mathbf{z})^T\mathbf{H}(\overline{\mathbf{d}}-\mathbf{z})]\\ &\equiv\min_{\mathbf{x}} E[(\boldsymbol{\varepsilon}_*-\mathbf{W}\boldsymbol{\varepsilon})^T\mathbf{H}(\boldsymbol{\varepsilon}_*-\mathbf{W}\boldsymbol{\varepsilon})]\\ &\equiv\min_{\mathbf{x}} -2E[(\boldsymbol{\varepsilon}_*-\mathbf{W}\boldsymbol{\varepsilon})^T\mathbf{H}\mathbf{W}\boldsymbol{\varepsilon}]+E[\boldsymbol{\varepsilon}^T\mathbf{W}^T\mathbf{H}\mathbf{W}\boldsymbol{\varepsilon}]\\ &\equiv\min_{\mathbf{x}} -2E[\boldsymbol{\varepsilon}_*^T\mathbf{H}\mathbf{W}\boldsymbol{\varepsilon}]+3E[\boldsymbol{\varepsilon}^T\mathbf{W}^T\mathbf{H}\mathbf{W}\boldsymbol{\varepsilon}]. \quad (A.5)\end{aligned}$$

Using the identity $\mathbf{W}\boldsymbol{\varepsilon}=\mathbf{W}\mathbf{d}-\mathbf{W}\mathbf{x}=(\overline{\mathbf{d}}-\mathbf{b}+\boldsymbol{\varepsilon}_*)-\mathbf{W}\mathbf{x}$, the first term in the last expression of (A.5) decomposes to $E[\boldsymbol{\varepsilon}_*^T\mathbf{H}\mathbf{W}\boldsymbol{\varepsilon}]=E[\boldsymbol{\varepsilon}_*^T\mathbf{H}(\overline{\mathbf{d}}-\mathbf{b}+\boldsymbol{\varepsilon}_*)]+E[\boldsymbol{\varepsilon}_*^T\mathbf{H}\mathbf{W}\mathbf{x}]$. The first term is constant with respect to $\mathbf{x}$ and the second term is zero due to the orthogonality of the least squares error to the input. Hence, the equivalent minimization problem reduces to $\min_{\mathbf{x}} E[\boldsymbol{\varepsilon}^T\mathbf{W}^T\mathbf{H}\mathbf{W}\boldsymbol{\varepsilon}]$.

C. Derivation of the Solution to Problem 1

In the general case of an arbitrary symmetric positive definite weighting factor $\mathbf{G}$, the gradient of the cost function with respect to all weights are found to be

$$\begin{aligned}\frac{\partial J}{\partial w_{kl}}&=\frac{1}{N}\sum_{s=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\gamma_{ij}f'(\overline{d}_{is})f'(\overline{d}_{js})\left[\overline{\varepsilon}_{is}\frac{\partial\overline{\varepsilon}_{js}}{\partial w_{kl}}+\frac{\partial\overline{\varepsilon}_{is}}{\partial w_{kl}}\overline{\varepsilon}_{js}\right]\\ &\quad k=1,\dots,n,\ \ l=1,\dots,m\\ \frac{\partial J}{\partial b_k}&=\frac{1}{N}\sum_{s=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\gamma_{ij}f'(\overline{d}_{is})f'(\overline{d}_{js})\left[\overline{\varepsilon}_{is}\frac{\partial\overline{\varepsilon}_{js}}{\partial b_k}+\frac{\partial\overline{\varepsilon}_{is}}{\partial b_k}\overline{\varepsilon}_{js}\right]\\ &\quad k=1,\dots,n \quad (A.6)\end{aligned}$$

where denoting the Kronecker-delta function with δ_{kj}

$$\frac{\partial\overline{\varepsilon}_{js}}{\partial w_{kl}}=-x_{ls}\delta_{kj},\quad \frac{\partial\overline{\varepsilon}_{js}}{\partial b_k}=-\delta_{kj}. \quad (A.7)$$

[11]In this example, the training set consisted of 50 input-desired data pairs. Note that the size of the trained MLP is comparable to the number of training patterns. Therefore, one would not expect good generalization performance. However, the point of this demonstration is not to test generalization capability, as this issue has been addressed in the previous case study. The goal of this example is to illustrate the importance of employing Rule 2 properly in classification problems. In addition, the usefulness of perturbing the desired output (class labels) is also discussed.

Equating all the derivatives in (A.6) to zero and rearranging the terms in order to obtain a square system of linear equations with $n \cdot m + n$ unknown variables yields (A.8). The solution to Problem 1 (i.e., the optimal weights of the corresponding layer in the MLP) is the solution of this linear system of equations for the variables w_{ip} and b_i. The solution could be obtained using a variety of computationally efficient approaches (e.g., Gaussian elimination with pivoting)

$$\begin{aligned}
&\sum_{i=1}^{n} b_i \gamma_{ik} \left[\sum_{s=1}^{N} f'(\bar{d}_{ks}) f'(\bar{d}_{is}) x_{ls} \right] \\
&\quad + \sum_{p=1}^{m} \sum_{i=1}^{n} w_{ip} \gamma_{ik} \left[\sum_{s=1}^{N} f'(\bar{d}_{ks}) f'(\bar{d}_{is}) x_{ls} x_{ps} \right] \\
&= \sum_{i=1}^{n} \gamma_{ik} \left[\sum_{s=1}^{N} f'(\bar{d}_{ks}) f'(\bar{d}_{is}) x_{ls} d_{is} \right] \\
&k = 1, \dots, n, \ l = 1, \dots, m \\
&\sum_{i=1}^{n} b_i \gamma_{ik} \left[\sum_{s=1}^{N} f'(\bar{d}_{ks}) f'(\bar{d}_{is}) \right] \\
&\quad + \sum_{p=1}^{m} \sum_{i=1}^{n} w_{ip} \gamma_{ik} \left[\sum_{s=1}^{N} f'(\bar{d}_{ks}) f'(\bar{d}_{is}) x_{ps} \right] \\
&= \sum_{i=1}^{n} \gamma_{ik} \left[\sum_{s=1}^{N} f'(\bar{d}_{ks}) f'(\bar{d}_{is}) d_{is} \right], \ k = 1, \dots, n.
\end{aligned} \tag{A.8}$$

References

[1] N. Rochester, J. H. Holland, L. H. Haibt, and W. L. Duda, "Tests on a cell assembly theory of the action of the brain, using a large digital computer," *IRE Trans. Inf. Theory*, vol. 2, pp. 80–93, 1956.

[2] A. M. Uttley, "Temporal and Spatial Patterns in a Conditional Probability Machine," in *Automata Studies*, C. E. Shannon and J. McCarthy, Eds. Princeton, NJ: Princeton Univ. Press, 1956, pp. 277–285.

[3] M. L. Minsky, "Steps toward artificial intelligence," in *Proc. Inst. Radio Engineers*, vol. 49, 1961, pp. 8–30.

[4] D. Gabor, W. P. L. Wilby, and R. Woodcock, "A universal nonlinear filter, predictor, and simulator which optimizes itself by a learning process," *Proc. Inst. Elect. Eng.*, vol. 108, pp. 422–435, 1960.

[5] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psych. Rev.*, vol. 65, pp. 386–408, 1958.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations of backpropagation errors," *Nature*, vol. 323, pp. 533–536, 1986.

[7] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the convergence of back-propagation method," *Biol. Cybern.*, vol. 59, pp. 257–263, 1988.

[8] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Netw.*, vol. 1, no. 4, pp. 295–308, 1988.

[9] S. Amari, "Natural gradient works efficiently in learning," *Neural Computat.*, vol. 10, pp. 251–276, 1998.

[10] R. L. Watrous, "Learning algorithms for connectionist networks: Applied gradient methods of nonlinear optimization," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 2, San Diego, CA, 1987, pp. 619–627.

[11] M. Moller, "Efficient training of feedforward neural networks," Ph.D. dissertation, Comput. Sci. Dept., Aarhus Univ. , Aarhus, Denmark, 1993.

[12] W. Duch, R. Adamczak, and N. Jankowski, "Initialization and optimization of multilayered perceptrons," in *Proc. 3rd Conf. Neural Networks Applications*, Kule, Poland, 1997, pp. 105–110.

[13] E. Barnard, "Optimization for training neural nets," *IEEE Trans. Neural Netw.*, vol. 3, no. 2, pp. 232–240, Mar. 1992.

[14] C. Fancourt and J. C. Principe, "Optimization in companion search spaces: The case of cross-entropy and the Levenberg-Marquardt algorithm," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, 2001, pp. 1317–1320.

[15] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.

[16] R. Battiti, "First and second order methods for learning: Between steepest descent and Newton's method," *Neural Computat.*, vol. 4, no. 2, pp. 141–166, 1992.

[17] C. M. Bishop, "Exact calculation of the Hessian matrix for the multilayer perceptron," *Neural Computat.*, vol. 4, no. 4, pp. 494–501, 1992.

[18] W. L. Buntine and A. S. Weigend, "Computing second derivatives in feed-forward networks: A review," *IEEE Trans. Neural Netw.*, vol. 5, no. 3, pp. 480–488, May 1994.

[19] M. A. Styblinski and T. S. Tang, "Experiments in nonconvex optimization: Stochastic approximation with function smoothing and simulated annealing," *Neural Netw.*, vol. 3, pp. 467–483, 1990.

[20] S. Bengio, Y. Bengio, and J. Cloutier, "Use of genetic programming for the search of a new learning rule for neural networks," in *Proc. IEEE World Congr. Computational Intelligence and Evolutionary Computation*, 1994, pp. 324–327.

[21] V. W. Porto, D. B. Fogel, and L. J. Fogel, "Alternative neural network training methods [Active sonar processing]," *IEEE Expert*, vol. 10, no. 3, pp. 16–22, 1995.

[22] M. Husken and C. Goerick, "Fast learning for problem classes using knowledge based network initialization," in *Proc. Int. Joint Conf. Neural Networks*, 2000, pp. 619–624.

[23] G. Thimm and E. Fiesler, "High-order and multilayer perceptron initialization," *IEEE Trans. Neural Netw.*, vol. 8, no. 2, pp. 349–359, Mar. 1997.

[24] V. Colla, L. M. Reyneri, and M. Sgarbi, "Orthogonal least squares algorithm applied to the initialization of multi-layer perceptrons," in *Proc. Eur. Symp. Artifical Neural Networks*, 1999, pp. 363–369.

[25] D. Nguyen and B. Widrow, "Improving the learning speed of two-layer neural networks by choosing initial values of the adaptive weights," in *Proc. Int. Joint Conf. Neural Networks*, vol. 3, 1990, pp. 21–26.

[26] G. P. Drago and S. Ridella, "Statistically controlled activation weight initialization (SCAWI)," *IEEE Trans. Neural Netw.*, vol. 3, no. 4, pp. 627–631, Jul. 1992.

[27] Y. F. Yam and T. W. S. Chow, "A new method in determining the initial weights of feedforward neural networks," *Neurocomput.*, vol. 16, no. 1, pp. 23–32, 1997.

[28] F. Biegler-Konig and F. Barnmann, "A learning algorithm for multilayered neural networks based on linear least squares problems," *Neural Netw.*, vol. 6, pp. 127–131, 1993.

[29] M. Di Martino and Prolasi, "Exploring and comparing the best "Direct methods" for the efficient training of MLP-networks," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1497–1502, Nov. 1996.

[30] E. Castillo, O. Fontenla-Romero, A. Alonso-Betanzos, and B. Guijarro-Berdinas, "A global optimum approach for one-layer neural networks," *Neural Computat.*, vol. 14, pp. 1429–1449, 2002.

[31] S. Y. Cho and T. W. S. Chow, "Training multilayer neural networks using fast global learning algorithm – Least-squares and penalized optimization methods," *Neurocomput.*, vol. 25, pp. 115–131, 1999.

[32] S. Lang, *Linear Algebra*, 3rd ed. New York: Springer-Verlag, 1987.

[33] G. Cybenko, "Approximation by superposition of a sigmoidal function," *Math. Control, Signals, Syst.*, vol. 2, pp. 303–314, 1989.

[34] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, pp. 251–257, 1991.

[35] A. S. Weigend and N. A. Gershenfeld, *Time Series Prediction: Forecasting the Future and Understanding the Past*. Reading, MA: Addison-Wesley, 1984.

[36] [Online]. Available: http://www.kernel-machines.org/

[37] J. D. Powell, N. P. Fekete, and C.-F. Chang, "Observer-based air-fuel ratio control," *IEEE Control Syst. Mag.*, vol. 18, no. 5, pp. 72–83, Oct. 1998.

[38] D. Erdogmus, A. U. Genc, and J. C. Principe, "A neural network perspective to extended Luenberger observers," *Inst. Meas. Control*, vol. 35, pp. 10–16, 2002.

[39] C. Wang and J. C. Principe, "Training neural networks with additive noise in the desired signal," *IEEE Trans. Neural Netw.*, vol. 10, no. 6, pp. 1511–1517, Nov. 1999.

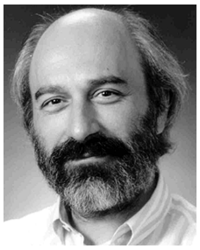
Deniz Erdogmus (M'00) received the B.S. degree in electrical and electronics engineering and mathematics in 1997 and the M.S. degree in electrical and electronics engineering, with emphasis on systems and control, in 1999, from the Middle East Technical University, Ankara, Turkey. He received the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, in 2002.

He is currently with the Department of Computer Science and Engineering, Oregon Graduate Institute, Oregon Health Sciences University, Portland, OR. His current research interests include applications of information theory to adaptive systems, signal processing, communications, and control.

Dr. Erdogmus is a member of Tau Beta Pi and Eta Kappa Nu.

Oscar Fontenla-Romero was born in Ferrol, Spain, in 1974. He received the B.S., M.S., and Ph.D. degrees in computer science from the University of A Coruna, Spain, in 1997, 1998, and 2002, respectively.

He became an Assistant Professor in the Department of Computer Science, University of A Coruna, in 2004. His current research interests include new linear learning methods and noise immunity for neural networks and functional networks. His main current areas are neural networks, functional networks, and nonlinear optimization.



Jose C. Principe (F'00) is a Distinguished Professor of Electrical and Computer Engineering and Biomedical Engineering at the University of Florida, Gainesville, where he teaches advanced signal processing, machine learning, and artificial neural networks (ANNs) modeling. He is BellSouth Professor and the Founder and Director of the University of Florida Computational NeuroEngineering Laboratory (CNEL). His primary area of interest is processing of time-varying signals with adaptive neural models. The CNEL Lab has been studying signal and pattern recognition principles based on information theoretic criteria (entropy and mutual information).

Dr. Principe is a member of the AdCom of the IEEE Signal Processing Society, Member of the Board of Governors of the International Neural Network Society, and Editor-in-Chief of the IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING. He is a member of the Advisory Board of the University of Florida Brain Institute and has authored more than 90 publications in refereed journals, ten book chapters, and 200 conference papers. He has directed 35 Ph.D. dissertations and 45 Masters theses. He recently wrote an interactive electronic book entitled *Neural and Adaptive Systems: Fundamentals Through Simulation* (New York: Wiley).

Amparo Alonso-Betanzos (M'88) was born in Vigo, Spain, in 1961. She received the B.S. degree in chemical engineering from the University of Santiago, Santiago, Spain, in 1984. In 1985, she joined the Department of Applied Physics, Santiago de Compostela, Spain, where she received the M.S. degree for work in monitoring and control of biomedical signals. In 1988, she received the Ph.D. (*cum laude* and *premio extraordinario*) degree for work in the area of medical expert systems.

From 1988 through 1990, she was a Postdoctoral Fellow in the Department of Biomedical Engineering Research, Medical College of Georgia, Augusta. She is currently a Full Professor in the Department of Computer Science, University of A Coruna. Her main current areas are hybrid intelligent systems, intelligent multiagent systems, linear optimization methods, and entropy based cost functions for neural networks and functional networks.

Dr. Alonso-Betanzos is a member of various scientific societies, including the Association for Computing Machinery.

Enrique Castillo received the M.Sc. and Ph.D. degrees in civil engineering from the Polytechnical University of Madrid, Spain, in 1969 and 1974, respectively, the M.Sc. degree in mathematics from the Complutense University of Madrid, Spain, in 1974, and the Ph.D. degree in civil engineering from Northwestern University, Evanston, IL, in 1972.

He joined University of Cantabria's Department of Applied Mathematics and Computational Sciences as an Assistant Professor, where he is currently a Full Professor. He is the Director of a program of study on informatics and computation in South America. His research interests are in the areas of applied statistics (mainly extreme values), functional equations, uncertainty in artificial intelligence, and expert systems and probabilistic network models. He has authored and coauthored 12 books and won several awards. He has published more than 300 works in 82 different journals and congresses.

Dr. Castillo is a member of several national and international societies related to statistics, engineering, and mathematics. He is a Member of the Royal Spanish Academy of Engineering.