TRUST-TECH based Neural Network Training

Hsiao-Dong Chiang and Chandan K. Reddy
School of Electrical and Computer Engineering
Cornell University, Ithaca, NY - 14853
Email: ckr6@cornell.edu

Abstract

Supervised learning using artificial neural networks has numerous applications in various domains of science and engineering. Efficient training mechanisms in a neural network play a vital role in deciding the network architecture and the accuracy of the classifier. Most popular training algorithms tend to be greedy and hence get stuck at the nearest local minimum of the error surface. To overcome this problem, some global methods (like multiple restarts, genetic algorithms, simulated annealing etc.) for efficient training make use of stochastic approaches in combination with local methods to obtain an effective set of training parameters. Due to the stochastic nature and lack of effective *fine tuning* capability, these algorithms often fail to obtain an optimal set of training parameters. In this paper, a new method to improve the subspace parameter search capability of training algorithms is proposed. This new method takes advantage of TRUST-TECH (TRansformation Under STability-reTaining Equilibrium CHaracterization) to compute neighborhood local minimum of the error surface. The proposed approach obtains multiple local optimal solutions surrounding the current local optimal solution in a systematic manner. Empirical results on different machine learning datasets indicate that the proposed algorithm outperforms current algorithms available in the literature.

Keywords - artificial neural networks, gradient descent, nonlinear dynamical systems, global optimization, training, stability regions, local optimal solutions.

I. Introduction

Artificial neural networks (ANN) were developed in analogy to the human brain for the purpose of improving conventional learning capabilities. They are used for a wide variety of applications in diverse areas such as function approximation, time-series prediction, medical diagnosis, character recognition, load forecasting, speaker identification and risk management. These networks serve as excellent approximators of nonlinear continuous functions [17]. However, using an artificial neural network to model a system usually involves dealing with certain difficulties in achieving the best representation of the classification problem.

The two challenging tasks in the process of learning using ANNs are network architecture selection and optimal training. In deciding the architecture for the neural network (also known as Multi-Layer Perceptron, MLP), a larger network will always provide better prediction accuracy for the data available. However, such a large network that is too complicated and customized to some given problem will lose its generalization capability for the unseen data [5]. Also, every additional neuron translates to increased hardware cost. Hence, it is vital to develop algorithms that can exploit the potential of a given architecture which can be achieved by obtaining the global minimum of the error on the training data. Hence, the goal of optimal training of the network is to find a set of weights that achieves the global minimum MSE [12]. Fig. 1 shows the architecture of a single hidden layer neural network with n input nodes, k hidden nodes and 1 output node. The network is trained to deliver the output value (Y_i) of the the i^{th} sample at the output node which will be compared to the actual target value (t_i).

The main focus of this paper is to develop a robust training algorithm for obtaining the optimal set of weights of an artificial neural network. Several training algorithms have been extensively studied in the literature [12]. Backpropagation (BP) algorithm is a very robust deterministic local method that have received significant attention. Though BP is comparatively cheaper in terms of time and easy to implement, it can only attain local optimal solutions nearest to the starting conditions. These solutions, however, might

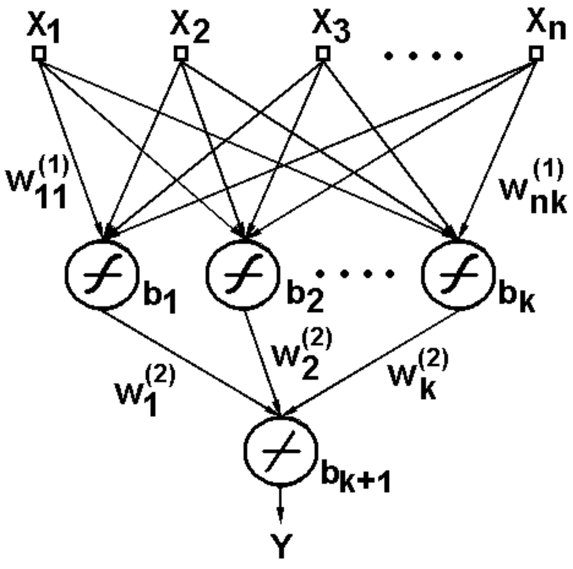

Fig. 1. The Architecture of MLP with single hidden layer having k hidden nodes and the output layer having a single output node. $x_1, x_2, ..., x_n$ is an n-dimensional input feature vector. w_{ij} are the weights and $b_1, b_2, ..., b_k$ are the biases for these k nodes. The activation function for the hidden nodes is sigmoidal and for the output node is linear.

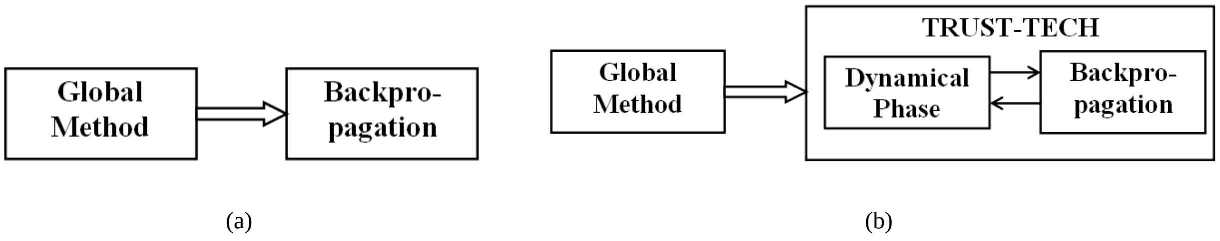

Fig. 2. Comparison between the two frameworks (a) Traditional approach and (b) TRUST-TECH based approach. The main difference is the inclusion of the stability region based dynamical phase that can explore the neighborhood solutions.

not be the best ones available in the vicinity of the search space. On the contrary, some global methods like multiple random starts, genetic algorithms and simulated annealing can identify promising regions of the weight space, but are essentially stochastic in nature and computationally expensive. Expecting such stochastic algorithms to fine-tune the training weights will be even more time consuming. Thus, there is a necessity to efficiently search for good solutions in promising regions of the solution space, which can be accomplished by the newly proposed TRUST-TECH based algorithm. In this paper, we introduce a novel algorithm that will search the subspace in a systematic tier-by-tier manner. Fig. 2 compares the traditional approach with our proposed approach. The main difference between the two approaches is the dynamical phase, where an improved set of training weights are obtained by systematically exploring the neighborhood local optimal solutions in a promising subspace.

The rest of this paper is organized as follows: Section II discusses the relevant background and the problem formulation. Section III introduces various notations and discusses the details about training neural networks. Section IV explains the details about the problem transformation and introduces the proposed algorithm. Section V gives the necessary implementation details. Section VI then discusses the results on standard benchmark datasets and finally, Section VII concludes our discussion along with some future research directions.

II. Background

The performance of a network is usually gauged by measuring the mean square error (MSE) of its outputs from the expected target values. The goal of optimal training is to find a set of parameters that achieves the global minimum of the MSE [3], [25], [17]. For an n-dimensional dataset, the MSE over Q samples in the training set is given by:

$$C(W) = \frac{1}{Q} \sum_{i=1}^{Q} [t(i) - y(X, W)]^2 \tag{1}$$

where $t(i)$ is the target output for the i^{th} sample, X is the input vector and W is the weight vector. The MSE as a function of the parameters will adopt a complex topology containing several local optimal solutions. The network's weights and thresholds must be set so as to minimize the prediction error made by the network. Since it is not possible to analytically determine the global minimum of the error surface, the neural network training is essentially an exploration of the error surface for an optimal set of parameters that attains this globally optimal solution.

Training algorithms can be broadly classified into '*local*' and '*global*' methods. Local methods begin at some initial points and deterministically move towards a local minimum. From an initial random configuration of weights and thresholds, these local training methods incrementally(greedily) seek for improved solution until they reach a local minimum. Typically, some form of the gradient information at the current point on the error surface is calculated and used to make a downhill move. Eventually, the algorithm stops at a low point, which usually is a local minimum. In the context of training neural networks, this local minima problem is a well-studied research topic [9]. The most commonly used training method in MLP is the backpropagation algorithm [24] which has been tested successfully for different kinds of problems. Despite having many variant implementations, BP faces the problem of stopping at local minimum instead of proceeding towards the global minimum [13], [25]. Modifications [16] to the basic BP model have been suggested to help the algorithm escape from being trapped in a local minimum. However, while these improved methods reduce the tendency to sink into local minimum by providing some form of perturbations to the search direction, it does not train the network to converge to a global minimum within a reasonable number of iterations [14], [26]. Based on the movement towards improved solutions, local methods can be subdivided into two categories:

1) **Line search methods:** These algorithms select some descent direction (based on the gradient information) and minimize the error function value along this particular direction. This process is repeated until a local minimum is reached. Most popular choices for the descent directions are Newton's direction or conjugate direction. In the context of neural networks, apart from the obvious steepest descent methods, other widely used line search algorithms are Newton's method [2], the BFGS method [20] and conjugate gradient methods [7], [18].
2) **Trust region methods:** Trust region methods are by far the fastest convergent methods compared to the above mentioned line-search methods. The surface is assumed to be a simple model (like a parabola) such that the minimum can be located directly if the model assumption is good which usually happens when the initial guess is close to the local minimum. They require more storage space compared to conjugate gradient methods [11] and hence are not optimal for large-scale applications.

All these local methods discussed so far assume that they already have an initial guess to begin with. Usually the quality of the final solution depends significantly on the initial set of parameters available. Hence, in practice, none of these local methods are used alone. They are typically combined with stochastic global methods which yield a promising set of parameters in the weight space. These global methods explore the entire error surface and thus the chance of attaining a near-global optimal solution is high. More advanced techniques like Genetic algorithms [5] and simulated annealing [1] are applied in combination with standard BP inorder to allow for more promising solutions and avoid being stuck at local minimum

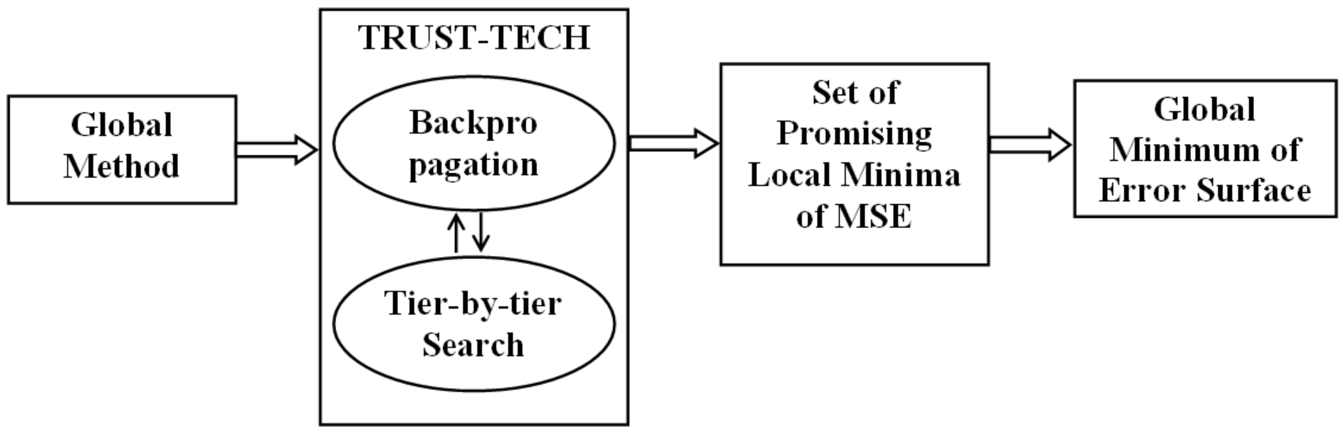
Global Method
TRUST-TECH
Backpro pagation
Tier-by-tier Search
Set of Promising Local Minima of MSE
Global Minimum of Error Surface

Fig. 3. Block Diagram of the architecture of our method.

[22]. The use of various global optimization algorithms for finding an effective set of training parameters is comprehensively given in [25]. Although these methods (asymptotically) guarantee convergence to the global minimum, it usually exhibits very slow convergence even for simple learning tasks. Though methods can explore the entire solution space effectively and obtain promising local optimal solutions, it lacks fine-tuning capabilities to obtain a precise final solution and requires local methods like BP to be employed. Other traditional methods like Monte Carlo method, Tabu search, ant colony optimization and particle swarm optimization are also stochastic in nature and suffer from the same problems described above.

From the above discussion, one can realize that there is a clear gap between global and local methods. Typically, most of the successful practical algorithms are a combination of these global and local methods. In other words, these two approaches do not communicate well between each other. Approaches that might resemble our methodology are TRUST [6] and dynamic tunneling [23]. These methods attempt to move out of the local minimum in a stochastic manner. The training algorithm proposed in this paper differs from these two methods by deterministically escaping out of the local minimum and systematically exploring multiple local minima on the error surface in a tier-by-tier manner in order to advance towards the global minimum. This approach is based on the fundamental concepts of stability regions that were established in [8], [15]. Fig. 3 shows the block diagram of the TRUST-TECH methodology. Basically, a global method yields initial points in certain promising regions of the search space. These initial points are used to search the neighborhood subspace in a systematic manner. TRUST-TECH relies on a robust, fast local method to obtain a local optimal solution. It explores the parameter subspace in a tier-by-tier manner by transforming the function into its corresponding dynamical system and exploring the neighboring stability regions. Thus, it gives a set of promising local optimal solutions from which a global minimum is selected. In this manner, TRUST-TECH can be treated as an effective interface between the global and local methods, which enables the communication between these two methods. It also allows the flexibility of choosing different global and local methods depending on their availability and performance for certain specific classification tasks.

III. Training Neural Networks

Without loss of generality, we consider a feedforward neural network with one input layer, one hidden layer and one output layer. Specifically, the output layer contains only one node that will yield all the possible target values depending on its activation function. Table I gives the important notations used in the rest of the paper.

Let k be the number of hidden nodes in the hidden layer and the input vector is n-dimensional. Then the final nonlinear mapping of our model is given by :

TABLE I
DESCRIPTION OF THE NOTATIONS USED

Notaion	Description
Q	Number of training samples
X	Input vector
W	Weight vector
n	Number of features
k	Number of hidden nodes
w_{0j}	weight between the output node and the j^{th} hidden node
w_{ij}	weight between the i^{th} input node and the j^{th} hidden node
b_0	bias of the output node
b_j	bias of the j^{th} hidden node
ϕ_1	Activation function of the hidden nodes
ϕ_2	Activation function of the output node
t_i	target value of the i^{th} input sample
y	output of the network
e_i	Error for the i^{th} input sample

$$y(W,X) = \phi_2\left(\sum_{j=1}^{k} w_{0j}\phi_1\left(\sum_{i=1}^{n} w_{ij}x_i + b_j\right) + b_0\right) \tag{2}$$

where ϕ_1 and ϕ_2 are the activation functions of the hidden nodes and the output nodes respectively. ϕ_1 and ϕ_2 can be same functions or can be different functions. We have chosen to use ϕ_2 to be sigmoidal and ϕ_1 to be linear. Results in the literature [10], suggest that this set of activation functions yield the best results for feedforward neural networks. As shown in Fig. 1, w_{0j} indicate the weights between the hidden layer and the output layer and w_{ij} indicate the weights between the input layer and the hidden layer. b_j are the biases of the k hidden nodes and b_0 is the bias of the output node. x_i is the n-dimensional input feature vector and X_i indicates the i^{th} training sample. The task of the network is to learn associations between the input-output pairs $(X_1, t_1), (X_2, t_2), ..., (X_Q, t_Q)$. The weight vector to be optimized is constructed as follows:

$$W{=}(w_{01}, w_{02}, .., w_{0k}, .., w_{n1}, w_{n2}, .., w_{nk}, b_0, b_1, b_2.., b_k)^T$$

which includes all the weights and biases that are to be computed. Hence, the problem of training neural networks is s-dimensional unconstrained minimization problem where $s = (n+2)k+1$.

$$\min_{W} C(W) \tag{3}$$

The mean squared error which is to be minimized can be written as

$$C(W) = \frac{1}{Q}\sum_{i=1}^{Q} e_i^2(W) \tag{4}$$

where the error

$$e_i(W) = t_i - y(W, X_i) \tag{5}$$

The error cost function $C(\cdot)$ averaged over all training data is a highly nonlinear function of the synaptic vector W Ignoring the constant for simplicity, it can be shown that

$$\nabla C(W) = J^T(W)e(W) \tag{6}$$

$$\nabla^2 C(W) = J^T(W)J(W) + S(W) \tag{7}$$

where $J(w)$ is the Jacobian matrix

$$J(w) = \begin{bmatrix} \frac{\partial e_1}{\partial W_1} & \frac{\partial e_1}{\partial W_2} & . & . & \frac{\partial e_1}{\partial W_N} \\ \frac{\partial e_2}{\partial W_1} & \frac{\partial e_2}{\partial W_2} & . & . & \frac{\partial e_2}{\partial W_N} \\ . & . & . & & . \\ . & . & & . & . \\ \frac{\partial e_Q}{\partial W_1} & \frac{\partial e_Q}{\partial W_2} & . & . & \frac{\partial e_Q}{\partial W_N} \end{bmatrix}$$

and

$$S(w) = \sum_{i=1}^{Q} e_i(w) \nabla^2 e_i(w) \tag{8}$$

Generally, if we would like to minimize $J(w)$ with respect to the parameter vector w, any variation of Newton's method can be written as

$$\begin{aligned} \Delta w &= - \left[\nabla^2 C(w)\right]^{-1} \ \nabla C(w) \\ &= - \left[J^T(w) J(w) + S(w)\right]^{-1} J^T(w) e(w) \end{aligned} \tag{9}$$

IV. TRUST-TECH BASED APPROACH

In this paper, we exploit the topological structure of the error surface to explore multiple local optimal solutions in a systematic manner. Firstly, we describe the transformation of the original problem into its corresponding nonlinear dynamical system and then propose a new algorithm for finding multiple local optimal solutions.

A. Problem Transformation

This section mainly deals with the transformation of the original likelihood function into its corresponding nonlinear dynamical system and introduces some terminology pertinent to comprehend our algorithm. This transformation gives the correspondence between all the critical points of the error surface and that of its corresponding gradient system. To analyze the geometric structure of the error surface, we build a *generalized gradient system* described by

$$\frac{dw}{dt} = -grad_R \ C(w) = -R(w)^{-1} \nabla C(w) \tag{10}$$

where the error function C is assumed to be twice differentiable to guarantee unique solution for each initial condition $w(0)$ and $R(w)$ is a positive definite symmetric matrix (also known as *Reimannian metric*) for all $w \in \Re^s$. The state vector w belongs to the Euclidean space $\Re^s$, and the vector field $C : \Re^s \rightarrow \Re^s$ satisfies the sufficient condition for the existence and uniqueness of the solutions. The solution curve of Eq. (10) starting from w at time $t = 0$ is called a *trajectory* and it is denoted by $\Phi(w, \cdot) : \Re \rightarrow \Re^s$. A state vector w is called an *equilibrium point* of Eq. (10) if $\nabla C(w) = 0$. An equilibrium point is said to be *hyperbolic* if the Jacobian of C at point $\bar{w}$ has no eigenvalues with zero real part.

Definition 1: A hyperbolic equilibrium point is called a (asymptotically) *stable equilibrium point* (SEP) if all the eigenvalues of its corresponding Jacobian have negative real part.

An equilibrium point is called a *type-k equilibrium point* if its corresponding Jacobian has exact k eigenvalues with positive real part. The *stable* ($W^s(\tilde{x})$) and *unstable* ($W^u(\tilde{x})$) manifolds of an equilibrium point, say $\tilde{x}$, is defined as:

$$W^s(\tilde{x}) = \{x \in \Re^s \ : \ \lim_{t\to\infty} \Phi(x,t) = \tilde{x}\} \tag{11}$$

$$W^u(\tilde{x}) = \{x \in \Re^s \ : \ \lim_{t\to-\infty} \Phi(x,t) = \tilde{x}\} \tag{12}$$

It is interesting to note the relationship between (10) and (9) and obtain different local solving methods used to find the nearest local optimal solution with guaranteed convergence. For example, if $R(w) = I$, then it is a naive error back-propagation algorithm. If $R(w) = [J(w)^T J(w)]$ then it is the Gauss-Newton method and if $R(w) = [J(w)^T J(w) + \mu I]$ then it is the Levenberg-Marquardt method.

B. Stability Regions

Now, the task of finding multiple local minima on the error surface has been transformed into the task of finding multiple stable equilibrium points on its corresponding dynamical system. The advantage of our approach is that this transformation into the corresponding negative gradient system will yield more knowledge about the various dynamic and geometric characteristics of the original surface and leads to the development a powerful method for finding improved neighborhood solutions. For our algorithm, we are particularly interested in the properties of the local minima and their one-to-one correspondence of the critical points. To comprehend this transformation, we need to define *energy function*. A smooth function $V(\cdot) : \Re^s \rightarrow \Re^s$ satisfying $\dot{V}(\Phi(w,t)) \ < \ 0$, $\forall \ x \notin$ {set of equilibrium points (E)} and t $\in \Re^+$ is termed as energy function.

Theorem 4.1: [8]: $C(w)$ is a energy function for the gradient system (10).

Definition 2: A type-1 equilibrium point x_d (k=1) on the practical stability boundary of a stable equilibrium point x_s is called a *decomposition point*.

Definition 3: The *practical stability region* of a stable equilibrium point x_s of a nonlinear dynamical system (10), denoted by $A_p(x_s)$ and is the interior of closure of the stability region $A(x_s)$ which is given by :

$$A(x_s) = \{x \in \Re^s \ : \ \lim_{t\to\infty} \Phi(x,t) = x_s\} \tag{13}$$

The boundary of practical stability region is called the *practical stability boundary* of x_s and will be denoted by $\partial A_p(x_s)$. Theorem 4.2 asserts that the practical stability boundary is contained in the union of the closure of the stable manifolds of all the decomposition points on the practical stability boundary. Hence, if the decomposition points can be identified, then an explicit characterization of the practical stability boundary can be established using (14). This theorem gives an explicit description of the geometrical and dynamical structure of the practical stability boundary.

Theorem 4.2: (Characterization of practical stability boundary)[15]: Consider a negative gradient system described by (10). Let σ_i , i=1,2,... be the decomposition points on the practical stability boundary $\partial A_p(x_s)$ of a stable equilibrium point, say x_s. Then

$$\partial A_p(x_s) = \bigcup_{\sigma_i \in \partial A_p} \overline{W^s(\sigma_i)}. \tag{14}$$

Our approach takes advantage of these concepts of stability regions to compute neighborhood local minima on the error surface. Originally, the basic idea of our algorithm was to find decomposition points on the practical stability boundary. Since, each decomposition point connects two local minima uniquely, it is important to obtain the decomposition points from the given local minimum and then move to the next local minimum through this decomposition point [21]. Though, this procedure gives a guarantee that the local minimum is not revisited, the computational expense for tracing the stability boundary and identifying the decomposition point is high compared to the cost of applying the local method directly using the exit point without considering the decomposition point.

C. TRUST-TECH Algorithm

The proposed algorithm for training neural networks , uses a promising starting point (A^*) as input and outputs the best local minimum of the neighborhood in the weight space. Figure 4 shows the flowchart of our approach.

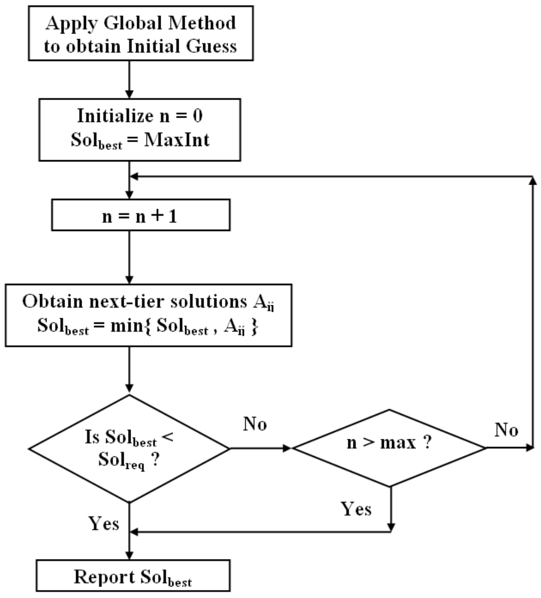
Apply Global Method to obtain Initial Guess
Initialize n = 0
Sol_best = MaxInt
n = n + 1
Obtain next-tier solutions A_ij
Sol_best = min{ Sol_best , A_ij }
Is Sol_best < Sol_req ?
No
n > max ?
No
Yes
Yes
Report Sol_best

Fig. 4. Flow chart of our method

Input: Initial guess(A^*), Tolerance (τ), Step size (s)
Output: Best local minimum (A_{ij}) in the neighborhood
Algorithm:
Step 1: Obtaining good initial guess (A^*): The initial guess for the algorithm can be obtained from other global search methods or from a purely random start. Some domain knowledge about the specific dataset that the network is being trained on, might help in eliminating non-promising set of initial weights.
Step 2: Moving to the local minimum (m_i): Using an appropriate local solver (such as conjugate-gradient, quasi-Newton or Levenberg-Marquardt), the local optimum m_i is obtained using A^* as the initial guess. The starting point will be x_i for the later stages.
Step 3: Determining the search direction (d_j): The eigenvectors d_j of the Jacobian are computed at m_i. These eigenvector directions might lead to promising regions of the subspace. Other search directions can also be chosen based on the specific problem that is being dealt.
Step 4: Escaping from the local minimum: Taking small step sizes away from m_i along the d_j directions increases the objective function value till it hits the stability boundary. However, the objective function value then decreases after the search trajectory moves away from the exit point on the stability boundary. x_i is used as initial guess and local solver is applied again (go to Step 2).
Step 5: Finding Tier-1 local minima (A_{1i}): Exploring the neighborhood of the local optimal solution corresponding to the initial guess leads to tier-1 local optima. Exploring from tier-k local optima leads to tier-$k+1$ local optima.
Step 6: Exploring Tier-k local minima (A_{kj}): Explore all other tiers in the similar manner described above (see Fig. 5). From all these solutions, the best one is chosen to be the desired global optimum.
Step 7: Termination Criteria: The procedure can be terminated when the best solution obtained so far is satisfactory (lesser than sol_{req}) or a predefined maximum number of tiers is explored.

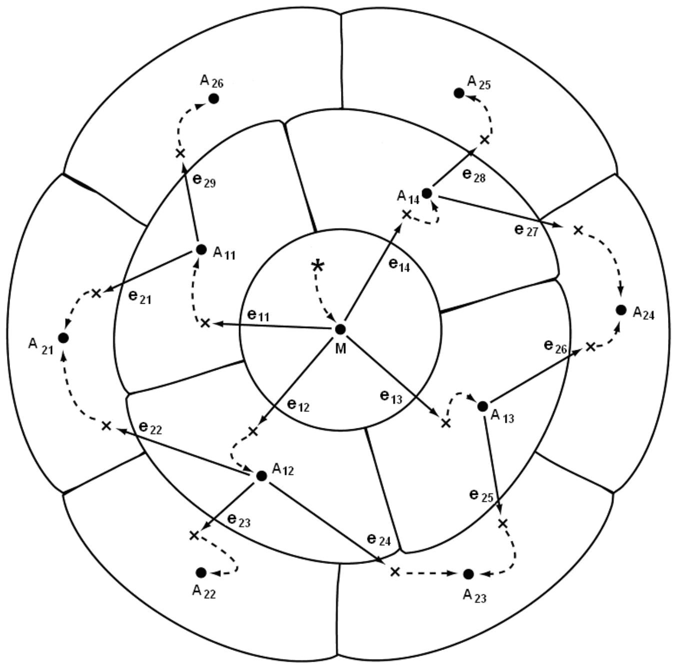

Fig. 5. Diagram Illustrating two tier exit point strategy. The '*' represents the initial guess. Dotted arrows represent the convergence of the local solver. Solid arrows represent the gradient ascent linear searches along eigenvector directions. 'X' indicates a new initial condition in the neighboring stability region. M represents the local minimum obtained by applying local methods from 'X'. A_{1i} indicates Tier-1 local minima. e_{1i} are the exit points between M and A_{1i}. Similarly, A_{2j} and e_{2j} are the second-tier local minima and their corresponding exit points respectively.

V. Implementation Details

All programs were implemented in MATLAB v6.5 and run on Pentium IV 2.8 GHz machines. This section describes the various implementation details used in our simulations. The following issues are discussed in detail : (i) Architecture and local methods, (ii) Initialization Schemes and (iii) TRUST-TECH.

A. Architecture and Local Methods

As described in the introduction section, we have chosen to demonstrate the capability of our newly proposed TRUST-TECH algorithm on a network with single hidden layer and an output layer containing only one output node. This architecture is not complicated and has the capability to precisely demonstrate the problems with the existing approaches. A network described in this paper contains n (number of attributes) input nodes which is equal to the number of features available in the dataset, one hidden layer with k nodes and one output node. Thus, each network has nk weights and k biases to the hidden layer, and k weights and one bias to the output node. Hence, training a neural network is necessarily a search problem of dimensionality $(n+2)k+1$. Each hidden node has a tangent-sigmoid transfer function and the output node has a pure linear transfer function. The number of nodes in the hidden layer is determined by incrementally adding hidden nodes, and selecting the architecture that achieves a compromise between minimal error value and minimal number of nodes. The trust region based Levenberg-Marquardt algorithm is chosen because of efficiency in terms of time and space consumption. It utilizes the approximation of the Jacobian in its iterative gradient descent, which will be used for generating promising directions in TRUST-TECH.

B. Initialization Schemes

Two different initialization schemes were implemented. The most basic global method which is multiple random starts with initial set of parameters between -1 and 1. More effective global method namely

Nguyen-Widrow (NW) algorithm [19] has also been used to test the performance of our algorithm. The NW algorithm is implemented as the standard initialization procedure in MATLAB. In both cases, the best initial set of parameters n terms of training error is chosen and improved with our TRUST-TECH algorithm.

Algorithm 1 $New\_Wts \;\; TRUST\_TECH(NET, Wts, s, \tau)$

```
$Wts = Train(NET, Wts, \tau)$
$Error = Estimate(NET, Wts)$
$Thresh = c * Error$
$Wts1[\,] = Neighbors(NET, Wts, s, \tau)$
for $k = 1$ to $size(Wts1)$ do
    if $Estimate(NET, Wts1[k]) < Thresh$ then
        $Wts2[k][\,] = Neighbors(NET, Wts1, s, \tau)$
    end if
end for
Return $best(Wts, Wts1, Wts2)$
```

C. TRUST-TECH

It is effective to use TRUST-TECH methodology for those promising solutions obtained from stochastic global methods. Algorithm 1 describes the two-tier TRUST-TECH algorithm. NET assumes to have a fixed architecture with a single output node. s is the step size required for moving out of the stability region to obtain the exit point. τ is the tolerance of error used for the convergence of the local method. $Weights$ give the initial set of weight parameter values. $Train$ function implements the Levenberg-Marquardt method that obtains the local optimal solution from the initial condition. The procedure $Estimate$ computes the mean square error (MSE) value of the network model. A threshold value ($Thresh$) is set based on this MSE value. The procedure $Neighbors$ returns all the next tier local optimal solutions from a given solution. After obtaining all the tier-1 solutions, Neighbors is again invoked (only for promising solutions) to obtain the second-tier solutions. The algorithm finally compares the initial solution, tier-1 and tier-2 solutions and returns the network corresponding to the lowest error among all these solutions.

Algorithm 2 $Wts[\,]$ $Neighbors$ ($NET, Wts,$ s, τ)

```
$[Wts, Hess] = Train(NET, Wts, \tau)$
$evec = Eig\_Vec(Hess)$
$Wts[\,] = NULL$
for $k = 1$ to $size(evec)$ do
    $Old\_Wts = Wts$
    $ext\_Pt = Find\_Ext(NET, Old\_Wts, s, evec[k])$
    if $(ext\_Pt)$ then
        $New\_Wts = Move(NET, Old\_Wts, evec[k])$
        $New\_Wts = Train(NET, New\_Wts, \tau)$
        $Errors = Estimate(NET, New\_Wts)$
        $Wts[\,] = Append(Wts[\,], New\_Wts, Errors)$
    end if
end for
Return $Wts[\,]$
```

The approximate Hessian matrix obtained during the updation in the Levenberg-Marquardt method used for computing the search direction. Since there is no optimal way of obtaining the search directions, the Eigen vectors of this Hessian matrix are used as search directions. Along each search direction, the exit point is obtained by evaluating the function value along that particular direction. The step size for evaluation is chosen to be the average step size taken during the convergence of the local procedure. The function value increases initially and then starts to reduce indicating the presence of exit point on the stability boundary. $Move$ function ensures that a new point (obtained from the exit point) is located in a different (neighboring) stability region. From this new initial guess, the local method is applied again to obtain the local optimal solution of the neighborhood stability region. For certain directions, there might not be exit points. For these directions, the search for exit points will be stopped after evaluating the function for certain number of steps. This avoids inefficient use of resources required to search in non-promising directions.

VI. Experimental Results

A. Benchmark Datasets

The newly proposed training method is evaluated using seven benchmark datasets taken from the UCI machine learning repository available at [4]. Since the main focus of the paper is the development of training algorithm, only simple experiments were conducted for choosing the architecture of the neural network. The hidden nodes in the hidden layer are added incrementally and the train error is computed. The final architecture is chosen with a fixed number of hidden nodes where there is no significant improvement in the training error. The number of nodes where the improvement in the training error is not significant is chosen as the final architecture. Table II summarizes the datasets. It gives the number of samples, input features, output classes along with the number of hidden nodes of the optimal architecture. These datasets have varying degrees of complexity in terms of sample size, output classes and the class overlaps. More details about these datasets are given in Appendix-A.

TABLE II
Summary of Benchmark Datasets.

Dataset ($\mathcal{D}$)	Sample Size (Q)	Input Features (n)	Output Classes (p)	Hidden Nodes (H)	Search Variables (n+2)k+1
Cancer	683	9	2	5	56
Diabetes	178	8	3	4	61
Image	2310	19	7	8	169
Ionosphere	351	34	2	9	325
Iris	150	4	3	3	19
Sonar	208	60	2	8	497
Wine	178	13	3	4	61

B. Error Estimation

To demonstrate the generalization capability (and hence the robustness) of the training algorithm, ten-fold cross validation is performed on each dataset. This practice of cross validation effectively removes any bias in the dataset segmentation. The use of the validation dataset allows early stopping of the local method and prevents over-fitting to a particular dataset. Essentially, each dataset is partitioned into ten folds of approximately equal size. Let these folds are denoted by $T_1, T_2, ...T_{10}$. Each time, the validation set will be T_i in which the the target labels will be deleted. The test set is T_j for $j = (i + 1)\ mod\ 10$. The training set comprises of the rest of the dataset and is given by:

$$\sum_{\substack{k=1 \\ k \neq i\ k \neq j}}^{10} T_k \tag{15}$$

The final MSE is the average of all the errors obtained across each of the ten folds. Usually, training error is much lower than the test error because the network is modeled using the training data and this data will be accurately classified compared to the unseen test data. All the network parameters including the architecture and the set of weights are obtained using the training data. Once the final model is fixed, the accuracy on the test data will provide an estimate of the generalization capability of the network models and training algorithm.

C. Classification Accuracy

The criteria of evaluation is given by the classification accuracy of the network model. The classification accuracy is given by the following formula:

$$\%\ accuracy = \frac{diff(\ t(i), y(W, X)\ )}{Q} * 100 \tag{16}$$

where $diff$ gives the number of misclassified samples. Tables III and IV shows the improvements in the train error and the test error using TRUST-TECH methodology. For effective implementation, only the best five tier-1 and corresponding tier-2 solutions were obtained using the TRUST-TECH strategy. For some of the datasets, there had been considerable improvement in the classifier performance.

TABLE III
CLASSIFICATION ERRORS FOR TRAINING AND TEST DATA USING TRUST-TECH WITH MULTIPLE RANDOM RESTARTS.

Dataset	Train			Test		
	MRS+BP	TRUST-TECH	Improvement	MRS+BP	TRUST-TECH	Improvement
Cancer	2.21	1.74	27.01	3.95	2.63	50.19
Image	9.37	8.04	16.54	11.08	9.74	13.76
Ionosphere	2.35	0.57	312.28	10.25	7.96	28.77
Iris	1.25	1.00	25.00	3.33	2.67	24.72
Diabetes	22.04	20.69	6.52	23.83	20.58	15.79
Sonar	1.56	0.72	116.67	19.17	12.98	47.69
Wine	4.56	3.58	27.37	14.94	6.73	121.99

TABLE IV
CLASSIFICATION ERRORS FOR TRAINING AND TEST DATA USING TRAINING AND TEST DATA USING TRUST-TECH WITH MATLAB INITIALIZATION.

Dataset	Train			Test		
	NW+BP	TRUST-TECH	Improvement	NW+BP	TRUST-TECH	Improvement
Cancer	2.25	1.57	42.99	3.65	3.06	19.06
Image	7.48	5.17	44.82	9.39	7.40	26.90
Ionosphere	1.56	0.92	69.57	8.67	6.54	32.57
Iris	1.33	0.67	100.00	3.33	2.67	25.00
Diabetes	21.41	19.55	9.53	23.70	21.09	12.37
Sonar	2.35	0.42	456.96	17.26	14.38	20.03
Wine	7.60	1.62	370.06	14.54	4.48	224.82

D. Visualization

The improvements of the TRUST-TECH method are demonstrated using spider web diagrams. Spider-web diagram (shown in Fig. 6) is a pretentious way to demonstrate the improvements in a tier-by-tier manner. The circle in the middle of the plot represents the starting local optimal solution. The basic two dimensions are chosen arbitrarily for effective visualization and the vertical axis is the percentage improvement in the classification accuracy. Unit distances are used between the tiers and the improvements are averaged out for 10 folds. The five vertical lines surrounding the center circle are the best five local minima obtained from a tier-1 search across all folds. The tier-2 improvements are also plotted. It should be noted that the best tier-1 solution need not give the best second tier solution.

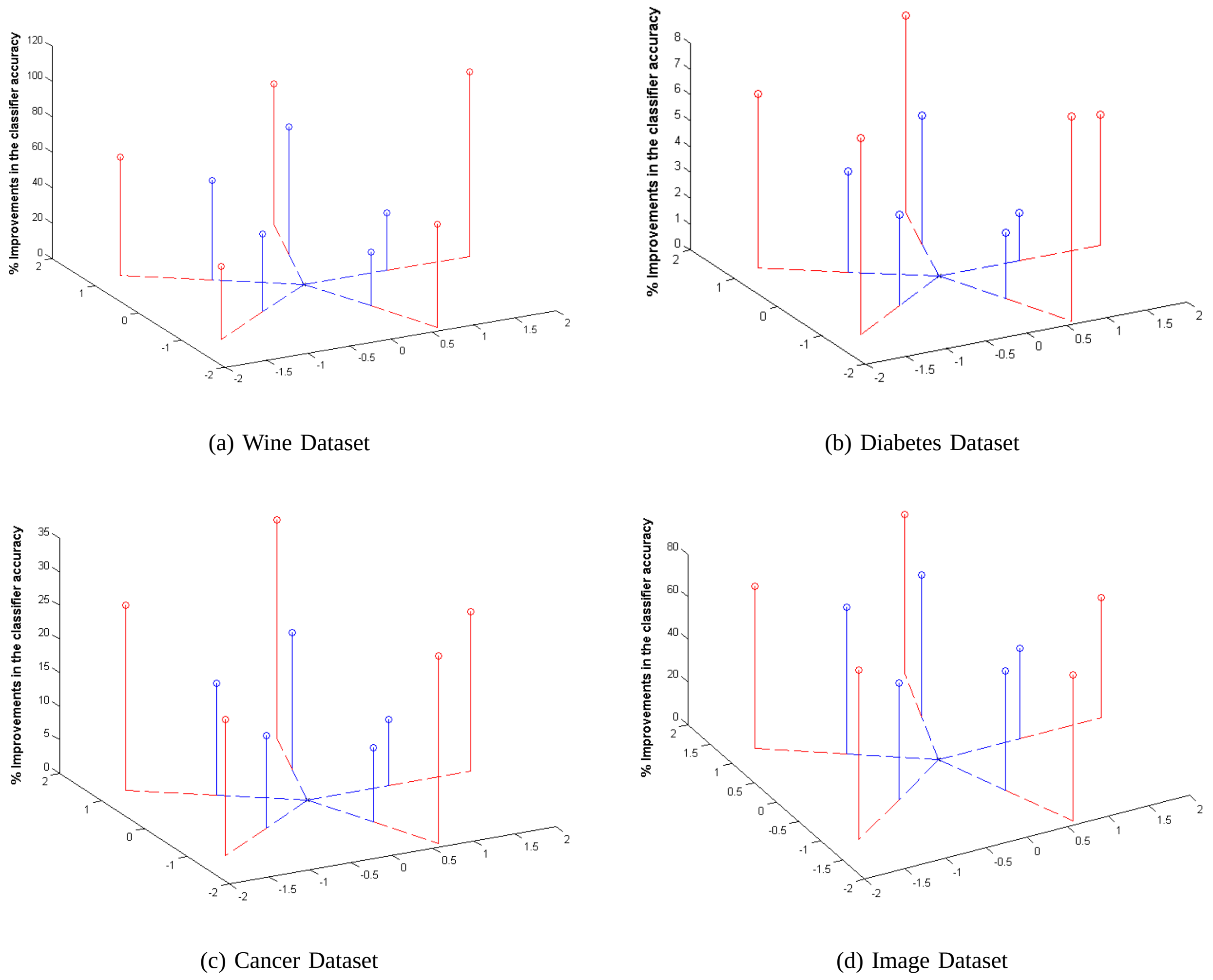

Fig. 6. Spider web diagrams showing the tier-1 and tier-2 improvements using TRUST-TECH method on various benchmark datasets. The basis two axes are chosen arbitrarily and the vertical axis represents the improvements in the classifier accuracy. The distances between each tier are normalized to unity.

VII. Conclusion and Future Work

Most successful algorithms for training artificial neural networks make use of some stochastic approaches in combination with local methods to obtain an effective set of training parameters. Due to the limited fine-tuning capability of these algorithms, even the best solutions that they can provide are locally optimal. In this paper, a new stability region based neighborhood search method for improving the local search capability of these training algorithms is proposed. This method improves the neural network model thus allowing improved classification accuracies by providing a better set of training parameters. Since it is not probabilistic in nature, multiple runs from an initial guess will provide exactly the same results. Different global and local methods work effectively on different datasets. The proposed method also allows the user to have the flexibility of choosing different global and local techniques for training. As a continuation of this work, the new training algorithm will be used for simultaneously deciding the architecture and the training parameters. Its performance on large-scale applications like character recognition, load forecasting etc will also be studied.

Appendix

APPENDIX-A: Description of Datasets

Cancer : This dataset contains data from cancer patients. It has 683 samples out of which 444 are benign cases and 239 are malignant cases. 9 attributes describing the tumor were used for classification.

Diabetes : This dataset gives information about patients who have some signs of diabetes according to World Health Organization criteria. Each sample has 8 real valued attributes. A total of 768 samples with 500 negative cases and 268 positive cases are available.

Image : This dataset contains images which were drawn randomly from a database of 7 outdoor images. The images were hand-segmented to create a classification for every pixel. There are 19 attributes that describe each instance (which is a 3x3 region) of a given image. The datset contains a total of 2310 samples.

Ionosphere : This radar data was collected by a system consisting a phased array of 16 high-frequency antennas with total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. The dataset consists of 351 samplees with 34 attributes. The classification task here is to separate good radar signals from that of the bad ones.

Iris : The dataset contains 3 classes of 50 samples each, where each class refers to a type of iris plant. It is relatively simple dataset where one class is linearly separable from the other two, but the other two have significant overlap and are not linearly separable from each other. The four attributes considered for classification are sepal length, sepal width, petal length and petal width. All attributes are measured in centimeters.

Sonar : This dataset is used for the classification of sonar signals. The task is to discriminate sonar signals bounced off a metal cylinder from those bounced off a roughly cylindrical rock. The dataset contains a total of 208 samples (111 for mines and 97 for rocks). The data set contains signals that were obtained from a variety of different aspect angles, spanning 90 degrees for the cylinder and 180 degrees for the rock. Each pattern is a set of 60 numbers in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

Wine : This dataset was obtained from the results of a chemical analysis of wines derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. A total of 178 samples with the following distribution (59,71,48).

Acknowledgments

We would like to thank Jay Huang, an undergraduate student for some preliminary MATLAB implementations of the proposed algorithm.

References

[1] S. Amato, B. Apolloni, P. Caporali, U. Madesani, and A. Zanaboni. Simulated annealing approach in backpropagation. *Neurocomputing*, 3(5):207–220, 1991.

[2] T. Battiti. First and second-order methods for learning: Between steepest descent and Newton's method. *Neural Computation*, 4(2):141–166, 1992.

[3] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[4] C.L. Blake and C.J. Merz. UCI repository of machine learning databases. *http://www.ics.uci.edu/mlearn/MLRepository.html, University of California, Irvine, Dept. of Information and Computer Sciences*, 1998.

[5] J. Branke. Evolutionary algorithms for neural network design and training. *Proceedings of the 1st Nordic Workshop on Genetic Algorithms and its Applications*, 3:145–163, 1995.

[6] B.C. Cetin, J. Barhen, and J.W. Burdick. Terminal repeller unconstrained subenergy tunnelling (TRUST) for fast global optimization. *Journal Optimization Theory and Applications*, 77(1):97–126, 1993.

[7] C. Charalambous. Conjugate gradient algorithm for efficient training of artificial neural networks. *IEEE Proceedings*, 139(3):301–310, 1992.

[8] H.D. Chiang and C.C. Chu. A systematic search method for obtaining multiple local optimal solutions of nonlinear programming problems. *IEEE Transactions on Circuits and Systems: I Fundamental Theory and Applications*, 43(2):99–109, 1996.

[9] M. Gori and A. Tesi. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86, 1992.

[10] D. Gorse, A.J. Shepherd, and J.G. Taylor. The new era in supervised learning. *Neural Networks*, 10(2):343–352, 1997.

[11] M. T. Hagan and M. Menhaj. Training feedforward networks with the marquart algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993, 1994.

[12] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[13] G. E. Hinton. How neural networks learn from experience. *Scientific American*, 267(3):144–151, 1992.

[14] D. Ingman and Y. Merlis. Local minimization escape using thermodynamic properties of neural networks. *Neural Networks*, 4(3):395–404, 1991.

[15] J. Lee and H.D. Chiang. A dynamical trajectory-based methodology for systematically computing multiple optimal solutions of general nonlinear programming problems. *IEEE Transactions on Automatic Control*, 49(6):888 – 899, 2004.

[16] A. V. Lehmen, E. G. Paek, P. F. Liao, A. Marrakchi, and J. S. Patel. Factors influencing learning by backpropagation. *Proceedings of IEEE International Conference on Neural Networks*, pages 335–341, 1988.

[17] F.H.F. Leung, H.K. Lam, S.H. Ling, and P.K.S. Tam. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks*, 14(1):79–88, 2003.

[18] A. F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.

[19] D. Nguyen and B. Widrow. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. *Proceedings of the International Joint Conference on Neural Networks*, 1990.

[20] S. Osowski, P. Bojarczak, and M. Stodolski. Fast second order learning algorithm for feedforward multilayer neural networks and its application. *Neural Networks*, 9(9):1583–1596, 1996.

[21] C. K. Reddy and H.D. Chiang. A stability boundary based method for finding saddle points on potential energy surfaces. *Journal of Computational Biology*, 13(3):745–766, 2006.

[22] C.R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley and Sons, New York, NY, 1993.

[23] P. RoyChowdhury, Y. P. Singh, and R. A. Chansarkar. Dynamic tunneling technique for efficient training of multi-layer perceptrons. *IEEE transactions on Neural Networks*, 10(1):48–55, 1999.

[24] D. E. Rumelhart, G. E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(9):533–536, 1986.

[25] Y. Shang and B. W. Wah. Global optimization for neural network training. *IEEE Computer*, 29(3):45–54, 1996.

[26] C. Wang and J. C. Principe. Training neural networks with additive noise in the desired signal. *IEEE Transactions on Neural Networks*, 10(6):1511–1517, 1999.