TRUST-TECH Based Neural Network Training

Hsiao-Dong Chiang and Chandan K. Reddy

***Abstract*— Efficient Training in a neural network plays a vital role in deciding the network architecture and the accuracy of these classifiers. Most popular local training algorithms tend to be greedy and hence get stuck at the nearest local minimum of the error surface and this corresponds to suboptimal network model. Stochastic approaches in combination with local methods are used to obtain an effective set of training parameters. Due to the lack of effective fine-tuning capability, these algorithms often fail to obtain such an optimal set of parameters and are computationally expensive. As a trade-off between computational expense and accuracy required, a novel method to improve the local search capability of training algorithms is proposed in this paper. This approach takes advantage of TRUST-TECH (TRansformation Under STability-reTaining Equilibrium CHaracterization) to compute neighborhood local minima on the error surface surrounding the current solution in a systematic manner. Empirical results on different real world datasets indicate that the proposed algorithm is computationally effective in obtaining promising solutions.**

I. Introduction

Artificial neural networks (ANN) are powerful statistical machine learning tools that are widely used in several domains of science and engineering for problems like function approximation, timeseries prediction, medical diagnosis, character recognition, load forecasting, speaker identification, risk management etc. They were developed in analogy to the human brain for the purpose of improving conventional learning capabilities. These networks serve as excellent approximators of nonlinear continuous functions [6]. However, using an artificial neural network to model a system involves in dealing with certain difficulties in achieving the best representation of the classification problem. Hence, it is vital to develop algorithms that can exploit the power of a given network architecture and this can be achieved by obtaining the global minimum of the error on the training data along with the cross validation procedure. The main goal of optimal training of the network is to find a set of weights that achieves the global minimum of the mean square error (MSE) [5]. We will consider a simple one hidden layer neural network with n (number of features) input nodes, k hidden nodes and 1 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 effectively is necessarily a search problem of dimension $s = (n + 2)k + 1$. 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 local minima problem is one of the thoroughly studied aspects of neural networks [5].

Dr. Hsiao-Dong Chiang and Dr. Chandan Reddy are with the School of Electrical and Computer Engineering, Ithaca, NY - 14853. Corresponding author: ckr6@cornell.edu.

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 by searching the parameter subspace in a systematic manner using nonlinear transformation and the concept of stability regions. 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 develops the stability region based training algorithm. Implementation details of our algorithm are given in section V. Section VI then discusses the results on standard benchmark datasets and finally, section VII concludes along with a discussion of 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 targets. The goal of optimal training is to find a set of parameters that achieves the global minimum of the MSE [5], [10], [6]. For a 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 is the target output, X is the input vector and W is the weight vector. The MSE as a function of the parameters will adopt a complex topology with 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 attain this global minimum.

Training algorithms can be broadly classified into '*local*' and '*global*' methods. Local methods begin at some initial guesses and deterministically lead to a nearby local minimum. From an initial random configuration of weights and thresholds, the local training methods incrementally seek for improved solution until they reach local minima. Typically, some form of the gradient information at the current point on the error surface is calculated and used to make a downhill move. Advanced techniques like Genetic algorithms and simulated annealing are applied in combination with standard Backpropagation (BP) inorder to allow for more promising solutions and avoid being stuck at local minima. These methods can explore the entire solution space effectively and obtain promising local optimal solutions, but they lack fine-tuning capability to obtain a precise final solution and require a local

methods to be employed. From both of the methods mentioned above, one can realize that there is a clear gap between these methods and our approach tries to communicate with both local and global methods.

Probably, the algorithms that resemble our methodology are TRUST [2] and dynamic tunneling [9]. These methods attempt to move out of the local minimum in a stochastic manner. The training algorithm proposed in this paper differs from these 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 region theory that were established in [3]. Basically, a global method yields initial points in certain promising regions of the search space. These promising 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 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. Also, using our procedure, we will be able to truncate global methods at early stages and not use them to fine-tune the solutions and thus saving a lot of computational effort.

III. Training Neural Networks

TABLE I
DESCRIPTION OF THE NOTATIONS USED

Notation	Description
Q	Number of training samples
X	Input vector
W	Weight vector
n	Number of features
k	Number of hidden nodes
w_{0j}	weight for j^{th} hidden and the output node
w_{ij}	weight for i^{th} input and j^{th} hidden node
b_0	bias of the output node
b_j	bias of the j^{th} hidden 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

Table I gives the important notations used in the rest of the paper. The final nonlinear mapping of the network model is given by:

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

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 [4], suggest that this set of activation functions yield the best results for feedforward neural networks. The task of the network is to learn associations between the input-output pairs $(X_1,t_1),(X_2,t_2),...,(X_Q,t_Q)$. Without loss of generality, lets construct the following weight vector :

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

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 $s=(n+2)k+1$. The mean squared error which is to be minimized can be written as

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

where the error $e_i(W) = t_i - y(W,X_i)$. 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

$$\nabla C(W) = J^T(W)e(W) \tag{3}$$

$$\nabla^2 C(W) = J^T(W)J(W) + S(W) \tag{4}$$

where $J(W)$ is the Jacobian matrix

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

and

$$S(W) = \sum_{i=1}^{Q} e_i(W)\nabla^2 e_i(W) \tag{5}$$

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

$$\begin{aligned} \Delta W &= -\left[\nabla^2 C(W)\right]^{-1} \nabla C(W) \\ &= -\left[J^T(W)J(W) + S(W)\right]^{-1} J^T(W)e(W) \end{aligned} \tag{6}$$

IV. Stability Region based Approach

In this paper, we exploit the geometric and topological structure of the error surface to explore multiple local optimal solutions in a systematic manner. We will first 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

$$\frac{dw}{dt} = -grad_R\ C(w) = -R(w)^{-1}\nabla C(w) \tag{7}$$

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. (7) 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. (7) if $C(w) = 0$. 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:

$$W^s(\tilde{x}) = \{x \in \Re^s \ : \ \lim_{t\rightarrow\infty} \Phi(x,t) = \tilde{x}\} \tag{8}$$

$$W^u(\tilde{x}) = \{x \in \Re^s \ : \ \lim_{t\rightarrow-\infty} \Phi(x,t) = \tilde{x}\} \tag{9}$$

It is interesting to note the relationship between (7) and (6) 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 (LM) 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 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 the 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: [3]: $C(w)$ is a energy function for the gradient system (7).

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 (7), denoted by $A_p(x_s)$ and is the interior of closure of the stability region $A(x_s)$ which is given by :

$$A(x_s) = \{x \in \Re^s \ : \ \lim_{t\rightarrow\infty} \Phi(x,t) = x_s\} \tag{10}$$

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 (11). 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)[3]: Consider a negative gradient system described by (7). 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

$$\partial A_p(x_s) = \bigcup_{\sigma_i \in \partial A_p} \overline{W^s(\sigma_i)}. \tag{11}$$

Our approach takes advantage of these concepts of stability regions to compute neighborhood local maxima on likelihood 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 maxima uniquely, it is important to obtain the decomposition points from the given local maximum and then move to the next local maximum through this decomposition point [8]. Though, this procedure gives a guarantee that the local maximum 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. Algorithm

The proposed algorithm uses a promising starting point (A^*) as input and outputs the best local minimum of the neighborhood in the weight space.

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 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): Using an appropriate local solver (such as conjugate-gradient, quasi-Newton or Levenberg-Marquardt), the local optimum M is obtained using A^* as the initial guess.
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. This new point 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 minima. Exploring from tier-k local minima leads to tier-$k+1$ local minima.
Step 6: Exploring Tier-k local minima (A_{kj}): Explore all other tiers in the similar manner described above. 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. 1 illustrates two tier TRUST-TECH methodology. 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 neighborhood 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. We will now describe some implementation issues of our algorithm. It is effective to use TRUST-TECH methodology for those promising solutions obtained from stochastic global methods. Hence, our algorithm assumes that it is being invoked from a promising set of initial parameters. Algorithm 1 describes the two-tier TRUST-TECH algorithm. NET assumes to have a fixed architecture with a single output node. ss 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 local 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.

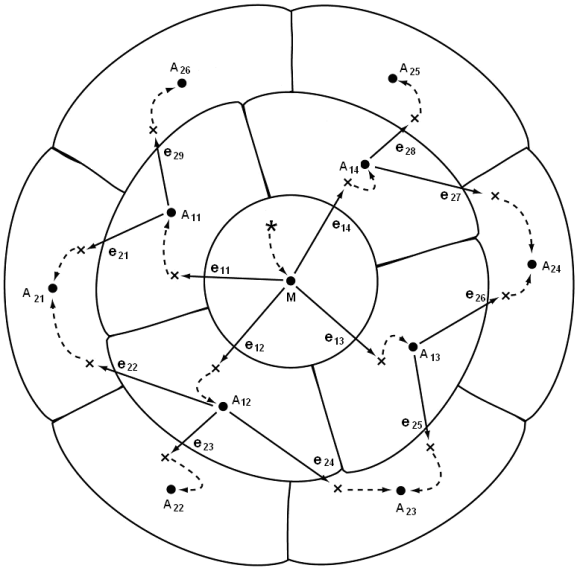


Fig. 1. Illustration of Tier-2 TRUST-TECH procedure for obtaining neighborhood local minima.

Algorithm 1 $TRUST\_TECH(NET, Wts, ss, \tau)$

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

The approximate Hessian matrix used for updation in the Levenberg-Marquardt method is utilized to obtain the search direction. Since there is no optimal way of obtaining promising search directions, the Eigen vectors of this Hessian matrix are used as search directions. Along each search direction, the exit

Algorithm 2 $Wts[\,]\ Neighbors\ (NET, Wts,\ ss,\ \tau)$

$[Wts, Hess] = Train(NET, Wts, \tau)$
$evec = Eig\_Vec(Hess)$
$Wts[\,] = NULL$
for $k = 1$ to $size(evec)$ **do**
  $Old\_Wts = Wts$
  $ext\_Pt = Find\_Ext(NET, Old\_Wts, ss, evec[k])$
  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 $Wts[\,]$

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. The search for exit points along these directions will be stopped if exit points are not found after certain number of function evaluations.

TABLE II
SUMMARY OF BENCHMARK DATASETS. DATASET (D), SAMPLE SIZE (Q), INPUT FEATURES (N), OUTPUT CLASSES (P), NO. OF HIDDEN NODES (K), NO. OF SEARCH VARIABLES ((N+2)K+1)

D	Q	n	p	k	(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

VI. EXPERIMENTAL RESULTS

Our algorithm is evaluated using seven benchmark datasets in the UCI machine learning repository [1]. Optimal architectures have been fixed for the networks based on some heuristic training. The main focus of our algorithm is to demonstrate the capability of obtaining optimal weight parameters and improvements in the generalization ability of the networks. 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. To demonstrate the generalization capability (and hence the robustness) of the training algorithm, 10-fold cross validation is performed on each dataset. The criteria of evaluation is given by the classification accuracy of the network model in terms of the percentage of misclassified samples in the test cases. Tables III and IV shows the improvements in the train error and the test error using TRUST-TECH methodology when multiple random starts (MRS) and MATLAB initialization (Nguyen-Widrow (NW) algorithm [7]) is used. Five tier-1 and corresponding tier-2 solutions were obtained using the TRUST-TECH strategy. For some of the datasets, there had been considerable improvements in the classifier performance. Spiderweb diagram (shown in Fig. 2) a pretentious way to demonstrate the improvements in a tier-by-tier manner.

VII. CONCLUSION

In this paper, a new method for improving the local search capability of these training algorithms is proposed. This method provides an optimal set of training parameters for a neural network model thus allowing improved classification accuracies. Because of its non-probabilistic nature, multiple runs of our algorithm from any given initial guess will provide exactly the same result. 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, this new stability region based 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 tested.

ACKNOWLEDGEMENTS

We would like to thank Jay Huang for helping in the initial phase of this work.

REFERENCES

[1] 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.
[2] 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.
[3] 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.
[4] D. Gorse, A.J. Shepherd, and J.G. Taylor. The new era in supervised learning. *Neural Networks*, 10(2):343–352, 1997.
[5] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] Y. Shang and B. W. Wah. Global optimization for neural network training. *IEEE Computer*, 29(3):45–54, 1996.

TABLE III

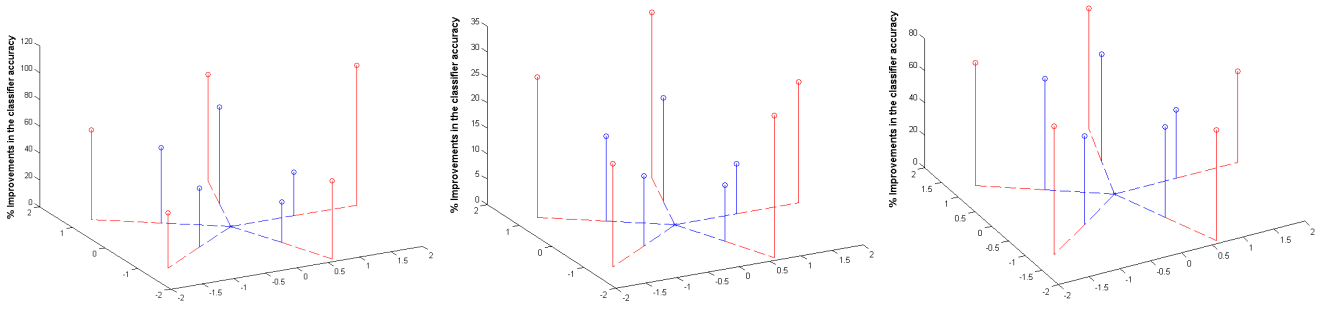
PERCENTAGE IMPROVEMENTS IN THE CLASSIFICATION ACCURACIES OVER THE TRAINING AND TEST DATA USING TRUST-TECH WITH MULTIPLE RANDOM RESTARTS.

Dataset	Train Error			Test Error		
	MRS+LM	TRUST-TECH	Improvement	MRS+LM	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

PERCENTAGE IMPROVEMENTS IN THE CLASSIFICATION ACCURACIES OVER THE TRAINING AND TEST DATA USING TRUST-TECH WITH MATLAB INITIALIZATION.

Dataset	Train Error			Test Error		
	NW+LM	TRUST-TECH	Improvement	NW+LM	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



(a) Wine Dataset (b) Cancer Dataset (c) Image Dataset

Fig. 2. Spider web diagrams showing the tier-1 and tier-2 improvements using TRUST-TECH method on various benchmark datasets. 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 represents the percentage improvement in the classification accuracy. The basis two axes are chosen arbitrarily and the vertical axis represents the improvements in the classifier accuracy. The five blue vertical lines surrounding the center circle are the best five local minima obtained from a tier 1 search across all folds. The five best second-tier improvements are plotted using red lines.