Determining regularization parameters for derivative free neural learning

Ranadhir Ghosh, Moumita Ghosh, John Yearwood, Adil Bagirov,

School of InformationTechnology and Mathematical Sciences, University of Ballarat,
PO Box 663, Ballarat – 3353, Australia
{r.ghosh, m.ghosh, j.yearwood, a.bagirov}@ballarat.edu.au

Abstract. Derivative free optimization methods have recently gained a lot of attractions for neural learning. The curse of dimensionality for the neural learning problem makes local optimization methods very attractive; however the error surface contains many local minima. Discrete gradient method is a special case of derivative free methods based on bundle methods and has the ability to jump over many local minima. There are two types of problems that are associated with this when local optimization methods are used for neural learning. The first type of problems is initial sensitivity dependence problem – that is commonly solved by using a hybrid model. Our early research has shown that discrete gradient method combining with other global methods such as evolutionary algorithm makes them even more attractive. These types of hybrid models have been studied by other researchers also. Another less mentioned problem is the problem of large weight values for the synaptic connections of the network. Large synaptic weight values often lead to the problem of paralysis and convergence problem especially when a hybrid model is used for fine tuning the learning task. In this paper we study and analyse the effect of different regularization parameters for our objective function to restrict the weight values without compromising the classification accuracy.

1 Introduction

Artificial neural networks (ANN) are the interconnection of basic units called artificial neurons. The performance of an ANN depends on both the weights and the transfer function. Most common form of transfer function used in literature is sigmoidal which makes the ANN model nonlinear. Using non-linear transfer function such as sigmoidal in the hidden layer provides the power of non-linearity to the network. For classification task, a choice between a linear or non-linear transfer functions exist for the output layer. Least square based methods such as householder transformation etc for finding the output layer weights often produce very large weights. This is the case especially when non-linear transfer function is used. Using

linear transformations on the other hand produce less weight values for the output layer however at the expense of an unaffordable classification accuracy decrease. Large weights decrease the performance of ANN by affecting its Generalization ability, also causes paralysis during learning.

Another problem that causes due to large weight is known as network paralysis. When the weights of the neural network become very high, no change occurs in the weight updating process. So the Root mean square error does not get change. Hence the network never learns further. Often the network is required to be retrained for many purposes. Even in the simplest case using a hybrid model is sometimes a necessity for fine tuning the learning task. Adaptation is another big issue quite often faced by many learning algorithms. Leaving large synaptic weights makes retraining extremely difficult and it requires a very long training time also.

One way to avoid large weight is to apply linear transfer function in output layer. It decreases the weight in output layer but generates poor classification accuracy.

The best way to avoid the problems is Regularization. Regularization is not a new term in the ANN community [22 – 27]. It is quite often used when least square based methods or ridge regression techniques are used for finding the weights in output layer. However the term regularization is not very common for multi-layered perceptron (MLP) as it is for radial basis function (RBF) network. This can be justified by the fact that least square based techniques are not used often for MLP. Regularization increases the generalization ability and avoid overfitting. There exists several way to regularize the weights. Regularization adds a penalty term to the error function. The usual penalty is the sum of squared weights times a decay constant. This process tends to minimize the large coefficients. The generalization ability of the network can depend crucially on the decay constant. At the very least, we need two different decay constants for input to hidden layer weight, and hidden to output layer weights. Choosing the decay constant is critical issue. One way to calculate the decay constant is to iteratively update the decay constant during training. Adjusting all these decay constants to produce the best estimated generalization error often requires vast amounts of computation.

In this research we apply a local search method to find the decay constant. The derivative for local search method simultaneously minimize the classification error and estimates the decay constants which bounds the weights within certain limit.

2 The Optimization Problem

If we consider a network with differentiable activation functions, then the activation functions of the output units become differentiable functions of both the input variables and of the weights and biases. If we define an error function (E), such as the sum of squares function, which is a differentiable function of the network outputs, then this error function is itself a differentiable function of the weights. We can therefore evaluate the derivatives of the error with respect to the weights, and these derivatives can then be used to find weight values, which minimize the error function, by using a variety of learning algorithms such as: backpropagation (BP), conju-

gate gradient, quazi-Newton and Levenberg-Marquardt (LM) methods [1]. Viewed from the mathematical programming perspective [2, 3], supervised batch training of a neural network is a classical non-linear optimisation problem: find the minimum of the error function given some set of training data. Traditionally this is accomplished by a suitable local descent technique, such as backpropagation. The independent variables are the weights **w**, and the objective function is usually the sum of squared errors (although other measures of error are also used). It is formulated mathematically as

$$\min E_1(w_o, w_k) = \sum \left(f\left(w_o^T z_k\right) - y_k \right)^2, \textit{where} \;\; z_k = f\left(w_k^T x_k\right)$$

Here *f* denotes the transfer function (in this work the transfer function was the standard sigmoid $f(t) = (1+e^{-t})^{-1}$), $\mathbf{w}_o$ denote output weights, $\mathbf{w}_h$ denote hidden layer weights, x_k are the input training data, y_k is the desired output and z_k denote the activations of hidden neurons. This problem is a global optimisation problem with many local minima.

We can consider a second error function called the absolute error function which is the summation of the absolute value of the error between the actual output and the desired output. It is mathematically formulated as

$$\min E_0(w_o, w_k) = \sum \left(\left| f\left(w_o^T z_k\right) - y_k \right| \right) \backslash, \textit{where} \;\; z_k = f\left(w_k^T x_k\right).$$

Despite its popularity, backpropagation has been widely criticized for its inefficiency [4, 5], and more advanced minimization techniques have been tried. Gradient based or gradient descent learning is named because of the learning characteristic of the algorithm, which uses gradient information of the error surface. Minimizing error with gradient descent is the least sophisticated but nevertheless in many cases a sufficient method. Typical response surfaces often possess local minima. Optimization techniques based on gradient descent may stagnate at these potentially suboptimal solutions, rendering the network incapable of sufficient performance. Newton and quasi-Newton methods may also fall prey to such entrapment.

Research indicates that empirical MLP error surfaces have an extreme ratio of saddle points. The results and experience of research into the properties of the error surface have identified an important feature of MLP error surfaces, which has implications for successful training of the net. The presence of relatively steep and flat regions is a fundamental feature of the error surface. Because of the complexity of the surface it is sometimes very hard and costly to compute the derivative. This is also known as the ill conditioning of the error surface. Algorithms that do not use gradient information directly will be affected implicitly through their reliance on the values of the error function. Algorithms such as Quasi-Newton (QN) and Levenberg-Marquardt [4], which use second order information, may not converge much faster than gradient methods in such a situation, and due to their increased computation effort may actually result in slower execution times.

All these techniques converge to the closest local minimum of the error function, which is very unlikely to be the global one. As a consequence, the network trained with a local algorithm may exhibit marginal performance. In this connection, the primitive backpropagation may result in a better solution than more sophisticated methods, because its disadvantages turn to the benefits of avoiding some shallow local minima [4]. The problem of many local minima has been widely addressed in the past [3, 6]. It was shown that training even a simple perceptron with non-linear transfer function may result in multiple minima [7].

Many tricks have been invented for avoiding this problem, such as restarting with a new random set of weights, training with noisy exemplars, and perturbing the weights when they appear to prematurely converge. While these methods may lead to improved solutions, there is no guarantee that such minima will not also be only locally optimal. Further, the same suboptimal solution may be rediscovered, leading to fruitless oscillatory training behaviour. Stochastic techniques offer an alternative to conventional gradient methods. The new stochastic optimisation algorithms significantly outperform the local methods, yet they do not provide any guarantee that their solution is indeed the global minimum. What is more, the number of local minima of the error function grows exponentially with the number of neurons, and the likelihood that these stochastic methods will find the global minimum is not that high. Deterministic global optimisation techniques could be found in [8, 9, 10]. They are based on more or less systematic exploration of the search space, and involve some assumptions about the class of the error function, such as Lipschitz properties. With a suitable choice of neuron transfer functions, these properties are satisfied. The biggest problem of deterministic techniques is their computational complexity, which grows exponentially with the number of variables (weights). Hence they are applicable only to small dimensional problems. On the other hand, it is in the systems with few neurons where global optimisation techniques are most needed. Indeed one of the goals of using global optimisation in ANN training is to reduce the number of neurons without sacrificing the performance, and this has been achieved in many cases [6]. The remedies include starting local descent from several random points, using tabu search, simulated annealing and genetic algorithms.

As a stochastic process simulated annealing can serve to generate weight and bias sets [11]. The popularity of tabu search has grown significantly in the past few years as a global search technique. Glover initially introduced (1986) and later developed Tabu search [12] into a general framework. Independently, Hansen (1987) proposed the Steepest Ascent, Mildest Descent (SAMD) algorithm [13] that uses similar ideas. Tabu search can be thought of as an iterative descent method. The use of evolutionary based algorithms for training neural networks has recently begun to receive a considerable amount of attention. Although the origins of evolutionary computing can be traced back to the late 1950's, the field remained relatively unknown to the broader scientific community over the last few decades. The fundamental work of Holland, Rechenberg, Schwefel and Fogel served to slowly change this picture during the 1970s [14]. Much of the research however has focused on the training of feed forward networks [Fogel, Fogel, and Porto, 1990; Whitley, Starkweather, and Bogart, 1990] [15] [16] [17]. Just as neurobiology is the inspiration for artificial neural

networks, genetics and natural selection are the inspiration of the genetic algorithm (GA). It was developed by John Holland [18]. They are based on a Darwinian type, survival of the fittest strategy. An advantage of using GAs for training neural networks is that they may be used for network with arbitrary topologies.

3 Regularization

We add the decay constant and the weights as a penalty term in the error function. As large weights in the output layer can produce outputs that are far beyond the range of the data hence it is more important to control the output layer weights. We use a linear penalty term for the hidden layer weights and nonlinear penalty term for output layer weights. If we consider the sum of square error function as

$\min E_1(w_o, w_k) = \sum \left(f\left(w_o^T z_k\right) - y_k \right)^2$, *where* $z_k = f\left(w_k^T x_k\right)$, the regularized error function will be as follows

$$E_{reg}(w_o, w_k) = E_1(w_o, w_k) + w_k \lambda_k + w_o^2 \lambda_0$$

where λ_k is the matrix of all hidden layer decay constant , and λ_0 is the matrix of all output layer decay constant. The actual objective function becomes

$$\min E_{reg}(w_o, w_k) = \min\left(\sum \left(f\left(w_o^T z_k\right) - y_k \right)^2 + w_k \lambda_k + w_o^2 \lambda_o \right), \text{where } z_k = f\left(w_k^T x_k\right)$$

If we consider the absolute error function as

$$\min E_0(w_o, w_k) = \sum \left(\left| f\left(w_o^T z_k\right) - y_k \right| \right), \text{where } z_k = f\left(w_k^T x_k\right)$$

the regularized error function will be as follows

$E_{reg}(w_o, w_k) = E_0(w_o, w_k) + w_k \lambda_k + w_o^2 \lambda_0$, where λ_k is the matrix of all hidden layer decay constant , and λ_0 is the matrix of all output layer decay constant.

The actual objective function becomes

$$\min E_{reg}(w_o, w_k) = \min\left(\sum \left(\left| f\left(w_o^T z_k\right) - y_k \right| \right) + w_k \lambda_k + w_o^2 \lambda_o \right), \text{where } z_k = f\left(w_k^T x_k\right)$$

4 Method

In this section we will give a brief description of the discrete gradient method. The full description of this method can be found in [19]. The discrete gradient method

can be considered as a version of the bundle method [22] when sub-gradients are replaced by their approximations - discrete gradients

Let f be a locally Lipschitz continuous function defined on R^n. A function f is locally Lipschitz continuous on R^n if in any open bounded subset $S \subset R^n$ there exists a constant $L > 0$ such that

$$f(x) - f(y) \le L\|x - y\|, \quad \forall x, y \in S. \tag{1}$$

The locally Lipschitz function f is differentiable almost everywhere and one can define for it a set of generalized gradients or a Clarke sub-differential [93], by

$$\partial f(x) = co\left\{v \in R^{n:} : \exists\left(x^k \in D(f), x^k \to x, k \to \infty\right) : v = \lim_{k \to +\infty} \nabla f\left(x^k\right)\right\}$$

Here $D(f)$ denotes the set where f is differentiable, co denotes the convex hull of a set and $\nabla f(x)$ stands for a gradient of the function f at a point $x \subset R^n$.

Let

$$S_1 = \left\{g \in R_n : \|g\| = 1\right\}$$
$$G = \left\{e \in R^n : e = (e_1, e_2, ..., e_n), \left|e_j\right| = 1, \ j = 1, ..., n\right\}$$
$$P = \left\{z(\lambda) : z(\lambda) \in R^1, \ z(\lambda) > 0, \ \lambda > 0, \ \lambda^{-1} z(\lambda) \to 0, \ \lambda \to 0\right\}$$
$$I(g, \alpha) = \left\{i \in \{1, ..., n\} : \left|g_i\right| \ge \alpha\right\}$$

where $\alpha \in \left(0, n^{-\frac{1}{2}}\right)$ is a fixed number. Here S_1 is the unit sphere, G is a set of vertices of the unit hyper cube in R^n and P is a set of univariate positive infinitesimal functions.

We define operators $H_i^j : R^n \to R^n$ for i =1,….,n, j = 0,…,,n by the formula

$$H_i^j g = \begin{cases} \left(g_1, ..., g_j, 0, ..., 0\right) & if \quad j < i, \\ \left(g_1, ..., g_{i-1}, 0, g_{i+1}, ..., g_j, 0, ..., 0\right) & if \quad j \ge i \end{cases}$$

We can see that $H_i^0 g = 0 \in R^n$ for all $i = 0, ..., n$. Let $e(\beta) = \left(\beta e_1, \beta^2 e_2, ...\beta^n e_n\right), \ \beta \in (0,1]$. For $x \in R^n$ we will consider vectors

$$x_i^j(g) \equiv x_i^j(g,e,z,\lambda,\beta) = x + \lambda g - z(\lambda)H_i^j e(\beta). \quad (8$$

Definition 1: The discrete gradient of the function *f* at the point $x \in R^n$ is the vector

$$\Gamma^i(g,e,z,\lambda,\beta) = \left(\Gamma_1^i, \Gamma_2^i, \ldots\ldots, \Gamma_n^i\right) \in R^n,\ g \in S_1,\ i \in I(g,\alpha)$$

with the following coordinates:

$$\Gamma_j^i = \left[z(\lambda)e_j(\beta)\right]^{-1}\left[f\left(x_i^{j-1}(g)\right) - f\left(x_i^j(g)\right)\right],\ j = 1,\ldots\ldots,n,\ j \neq i,$$

$$\Gamma_i^i = (\lambda g_i)^{-1}\left[f\left(x_i^n(g_i)\right) - f(x) - \sum_{j=1,\ j\neq i}^{n}\Gamma_j^i\left(\lambda g_j - z(\lambda)e_j(\beta)\right)\right]$$

Remark1: From the definition of the discrete gradient we can see that it is defined with respect to a given direction $g \in S_1$ and in order to calculate the discrete gradient we use step $\lambda > 0$ along this direction. The $n-1$ coordinates of the discrete gradient are defined as finite difference estimates to a gradient in some neighborhood of the point $x + \lambda g$. The *i*th coordinate of the discrete gradient is defined so that to approximate a sub-gradient of the function *f*. Thus the discrete gradient contains some information about the behavior of the function *f* in some region around the point *x*.

5 Experimental Result

All experiments were conducted for 5 different datasets (Austral, Breast Cancer, Cleveland Heart Disease, Diabetes and Liver) taken from UCI ML repository. The details of these datasets can be obtained from the UCI website. All results are given for weight determination using the discrete gradient method (DG) with two different error functions, the absolute error function and the sum of squared error function. Ten fold cross validation is used with 20 % of each dataset being withheld for testing. Each experiment is conducted for a range of hidden neurons (2-8).

The following table (Table 1) shows the classification accuracy of the ANN as a percentage, CPU time in seconds and the corresponding initial weight range for the discrete gradient method with the absolute error function (Error function 0). B stands for the weight range before applying the regularization factor, and A stands for the weight range after applying the regularization factor. The average classification accuracies were same before and after applying the proposed regularization factor. But our preliminary results have shown that the existing regularization factors produce less synaptic weight values at the expense of classification accuracy.

Table 1 Results for all data sets for discrete gradient method with error function 0

Dataset	#HN	Classification Accuracy (%)	CPU Time (s)	Weight range Hidden Layer				Weight range Output Layer			
				Min		Max		Min		Max	
				B	A	B	A	B	A	B	A
Austral	3	87.8	58.65	-50	-25	50	15	-30	-5	30	10
Breast Cancer	4	100	74.13	-50	-30	50	20	-50	-8	50	15
Clevelnnd	2	83.3	16.13	-50	-20	50	10	-50	-5	50	25
Diabetes	2	74	14.25	-50	-20	50	15	-50	-15	50	20
Liver	3	80	12.13	-50	-10	50	20	-50	-10	50	14

The following table (Table 2) shows the classification accuracy as a percentage, CPU time in seconds and the corresponding initial weight range for the discrete gradient method with sum of squares error function (Error function 1). B stands for the weight range before applying the regularization factor, and A stands for the weight range after applying the regularization factor.

Table 2 Results for all data sets for discrete gradient method with error function 1

Dataset	#HN	Classification Accuracy (%)	CPU Time (s)	Weight range Hidden Layer				Weight range Output Layer			
				Min		Max		Min		Max	
				B	A	B	A	B	A	B	A
Austral	2	86.7	34.54	-50	-10	50	10	-30	-15	30	10
Breast Cancer	2	100	24.52	-50	-10	50	20	-50	-10	50	10
Clevelnnd	2	80	18.5	-50	-20	50	20	-50	-5	50	20
Diabetes	4	76.5	64.12	-50	-30	50	10	-50	-10	50	15
Liver	3	86.7	16.13	-50	-10	50	20	-50	-10	50	14

6 Conclusion and Further Research

In this paper we have proposed a new weight regularization technique for MLP learning using a derivative free optimization method without loosing any classification accuracy. Less weight values increases the generalization ability and solve the problem of paralysis. Thus it helps to retrain the network to increase its adaptability and also fine tuning the learning task by applying further a hybrid model. In future we will modify our existing discrete gradient method to incorporate the constraints as a secondary objective function.

References

[1] Bishop, C. M.: Neural Networks for Pattern Recognition, Oxford Press, (1995).

[2] Mangasarian, O. L.: Mathematical programming in neural networks, ORSA Journal on Computing, Vol. 5 (1993), 349--360.

[3] Zhang, X. M., and Chen, Y. Q.: Ray-guided global optimization method for training neural networks, Neurocomputing, Vol. 30 (2000), 333-337.

[4] Masters, T.: Practical neural network recipes in C++, Academic Press, Boston, (1993).

[5] Masters, T.: Advanced algorithms for neural networks : a C++ sourcebook, Wiley, New York, (1995).

[6] Duch, W., and Korczak, J.: Optimization and global minimization methods suitable for neural networks, Neural computing surveys (1999).

[7] Coetzee, F. M., and Stonick, V. L.: On the uniqueness of weights in single-layer perceptrons, IEEE Transactions on Neural Networks, Vol. 7 (1996), 318(8).

[8] Horst, R. and Pardalos, P. M.: Handbook of global optimization, Kluwer Academic Publishers, Dordrecht ; Boston, (1995).

[9] Pinter, J.: Global optimization in action : continuous and Lipschitz optimization--algorithms, implementations, and applications, Kluwer Academic Publishers, Dordrecht ; Boston, (1996).

[10] Torn, A. and Zhilinskas, A.: Global optimization, Springer-Verlag, Berlin ; New York, (1989).

[11] Porto, V.W., Fogel, D. B., Fogel, L. J.: Alternative Neural Network training algorithm, Intelligent system, June (1995), Vol. 10, no. 3, 16 – 22.

[12] Glover, F., Future path for integer Programming and Links to Artificial Intelligence", Computer Operations Research, Vol. 13, 533 – 549.

[13] Hansen, P., and Jaumard, B., Algorithms for the Maximum satisfatibility problem, RUTCOR Research Report, 43 – 87, Rutger Unoversity, New Burnswick, NJ.

[14] Rechenberg, I.: Cybernetic solution path of an experimental problem, *Royal Aircraft Establishment, Library Translation* no. 1122, Farnborough, Hants, U.K, Aug, 1965.

[15]Whitley, D.,Starkweather, T., and Bogart, C.: Genetic algorithms and neural networks - optimizing connections and connectivity, Parallel Computing, Vol. 14, (1990), 347-361.

[16] Montana, D. and Davis, L.: Training feed forward neural networks using genetic algorithms, Proceedings of 11th International Joint Conference on Artificial Intelligence IJCAI-89, Vol. 1, (1989). 762-767.

[17] Goldberg, D. E.: Genetic algorithms in search, optimization, and machine learning, Addison-Wesley, Reading, MA, (1989).

[18] Holland, J. H.: Adaptation in natural and artificial systems, Ann Arbor, MI: The University of Michigan Press, (1975).

[19] Bagirov, A.M.: Derivative-free methods for unconstrained nonsmooth optimization and its numerical analysis, Investigacao Operacional, Vol. 19, (1999), 75-93.

[20] Bagirov, A.M.: Minimization methods for one class of nonsmooth functions and calculation of semi-equilibrium prices, Applied Optimization, Vol. 30: Progress in Optimization: Contribution from Australasia. A. Eberhard et al. (eds.), Kluwer Academic Publishers, Dordrecht, (1999), 147-175.

[21] Bagirov, A.M.: A method for minimization of quasidifferentiable functions, *Optimization Methods and Software*, Vol. 17, No. 1, (2002), 31-60.

[22] Hiriart-Urruty, J. B., and Lemarechal, C.: Convex Analysis and Minimization Algorithms, Springer-Verlag, Berlin, New York, 1993.

[23] Bartlett, P.L. For valid generalization, the size of the weights is more important than the size of the network. Mozer, M.C., Jordan, M.I., and Petsche, T., (eds.) Advances in Neural Information Processing Systems 9, Cambrideg, MA: The MIT Press, (1997), 134-140.

[24] Bishop, C.M.: Neural Networks for Pattern Recognition, Oxford: Oxford University Press, (1995),.

[25] Geman, S., Bienenstock, E. and Doursat, R.: Neural Networks and the Bias/Variance Dilemma. Neural Computation, Vol. 4, (1992), 1-58.

[26] Ripley, B.D.: Pattern Recognition and Neural Networks, Cambridge: Cambridge University Press, (1996).

[27] Weigend, A. S., Rumelhart, D. E., & Huberman, B. A.: Generalization by weight-elimination with application to forecasting. In: R. P. Lippmann, J. Moody, & D. S. Touretzky (eds.), Advances in Neural Information Processing Systems 3, San Mateo, CA: Morgan Kaufmann, (1991).