Fast Training Algorithms for Feed Forward Neural Networks

Luma N. M. Tawfiq
Yaseen A. Oraibi
Dept. of Mathematics/College of Education for Pure Science (Ibn AL-Haitham)
University of Baghdad

Received in: 4 October 2011 , Accepted in: 16 December 2011

Abstract

The aim of this paper, is to discuss several high performance training algorithms fall into two main categories. The first category uses heuristic techniques, which were developed from an analysis of the performance of the standard gradient descent algorithm. The second category of fast algorithms uses standard numerical optimization techniques such as: quasi-Newton . Other aim is to solve the drawbacks related with these training algorithms and propose an efficient training algorithm for FFNN.

Keyword : Artificial neural network, Feed Forward neural network, Training Algorithm.

Introduction

An Artificial neural network (Ann) is a simplified mathematical model of the human brain.It can be implemented by both electric elements and computer software. It is a parallel distributed processor with large numbers of connections, it is an information processing system that has certain performance characters in common with biological neural networks. Ann have been developed as generalizations of mathematical models of human cognition or neural biology, based on the assumptions that:

1- Information processing occurs at many simple elements called neurons that is fundamental to the operation of Ann's.

2- Signals are passed between neurons over connection links.

3- Each connection link has an associated weight which, in a typical neural net, multiplies the signal transmitted.

4- Each neuron applies an action function (usually nonlinear) to its net input (sum of weighted input signals) to determine its output signal.[1]

The units in a network are organized into a given topology by a set of connections, or weights, shown as lines in a diagram .

Ann is characterized by[2] :

1- Architecture: its pattern of connections between the neurons.

2- Training algorithm : its method of determining the weights on the connections.

3- Activation function.

Ann are often classified as single layer or multilayer. In determining the number of layers, the input units are not counted as a layer, because they perform no computation. Equivalently, the number of layers in the net can be defined to be the number of layers of weighted interconnects links between the slabs of neurons. This view is motivated by the fact that the weights in a net contain extremely important information [3].

Multilayer Feed Forward Architecture [4]

In a layered neural network the neurons are organized in the form of layers. We have at least two layers: an input and an outputlayer. The layers between the input and the output layer (if any) are called hidden layers, whose computation nodes are correspondingly called hidden neuronsor hidden units. Extra hidden neurons raise the network's ability to extract higher-order statistics from (input) data.

The Ann is said to be fullyconnectedin the sense that every node in each layer of the network is connected to every other node in the adjacent forward layer , otherwise the network is called partially connected. Each layer consists of a certain number of neurons; each neuron is connected to other neurons of the previous layer through adaptable synaptic weights w and biases b .

Training Feed Forward Neural Network [5]

Training is the process of adjusting connection weights w and biases b. In the first step, the network outputs and the difference between the actual (obtained) output and the desired (target) output (i.e.,the error) is calculated for the initialized weights and biases (arbitrary values). During the second stage, the initialized weights in all links and biases in all neurons are adjusted to minimize the error by propagating the error backwards (the back propagation algorithm). The network outputs and the error are calculated again with the adapted weights and biases, and the process (the training of the Ann) is repeated at each epoch until a satisfied output y_k(corresponding to the values of the input variables x) is obtained and the error is acceptably small. In most of the training algorithms a learning rate is used to determine the length of the weight update (step size) .

Gradient Descent Training Method(taringd) [4]

For the basic gradientdescent (GD)algorithm, the weights and biases are moved in the direction of the negative gradient of the performance function.

For the method of gradient descent, the weight update is given by :

$W_{k+1} = W_k + \eta_k(-g_k) \quad , \quad (1)$

Where $\eta_k$is a learning rate, controlling the distance between W_{k+1} and W_k and $g_k$is the gradient of the error surface at W_k, where W_k is the weight at iteration k.

But one of the drawbacks of using GD for minimization of the performance function is that the algorithm may converge to a local minimum in the error surface .

Improve Gradient Descent Method

The aim of this paper is to solve the drawbacks related with gradient descentalgorithms (GD) and propose an efficient training algorithm for FFNN .

Consider A. K. Jabber [5] improved gradient descent method when, he get new form of learning rate as following :

$$\eta_k = \frac{g_k^T g_k}{g_k^T H_k g_k} \quad , \quad (2)$$

That is (2) gives better approach to choose η at each iteration k, and he implement (2) in a new package using MATLAB 7, named **' traingdak' .**

Also , he improved calculate update weight by the following:

$$W_{k+1} = W_0 - \eta \sum_{i=1}^{N_w} \gamma_i \sum_{j=0}^{k} (1-\eta\lambda_i)^j u_i \quad , \quad (3)$$

By (3) he can calculate adjust weight W_{k+1} for any iterative k in **one-epoch** only and depending on W_0only .

Then the improvement gradient descentmethod, hold by calculating M (any integer number) iterative in one -epoch instead of calculate one iterative in each epoch .

But one drawbacks of (2) is calculating ofH_k(the Hessian matrix which is the second derivatives of the network errors with respect to the weight). In this paper we give better approach to choose η (learning rate) by the following calculating , consider the Taylor series :

$$E(W_{k+1}) = E(W_k) + g_k^T (W_{k+1} - W_k) + \frac{1}{2} (W_{k+1} - W_k)^T H_{Wk} (W_{k+1} - W_k)$$

Then gradient E with respect to W by the following :

$g_{k+1} = g_k + H_k (W_{k+1} - W_k) \quad , \quad (4)$

From adjusts weights by gradient descent method we have :

$$W_{k+1} - W_k = - \eta_k g_k \quad , \quad (5)$$

and substituting (5) in (4) we get :

$$g_{k+1} = g_k - \eta_k H_k g_k$$

Then we get new form of learning rate as following :

$$\eta_k = - (g_{k+1} - g_k) / (H_k g_k) \quad , \quad (6)$$

Then the Hessian matrix can be approximated as $H= J^T J$ (where J is the Jacobian matrix) that is :

$$\eta_k = - \Delta g_k / (J^T J\, g_k) \quad , \quad (7)$$

and we implement (7) in a new package using MATLAB 7.10 , named **" traingdly "**

Quasi-Newton Algorithms [6]

Quasi-Newton (or secant) methods are based on Newton's method but we require calculation of second derivatives (Hessian matrix) at each step. They update an approximate Hessian matrix at each iteration of the algorithm . The optimum weight value can be computed in an iterative manner by writing :

$$W_{k+1}=W_k-\eta_k H^{-1}g_k \qquad , \qquad (8)$$

whereη_k is the learning rate, g_k is the gradient of the error surface with respect to the W_k and H is the Hessian matrix (second derivatives of the error surface with respect to the weight (W_k)).

In this paper, we use BFGS Quasi-Newton algorithm (trainbfg) ,
this algorithm requires more computation for each iteration and practi- cal results show more storage require than the CG methods , although generally, converges in fewer iterations. Newton's method has a quadratic convergence property , that is $|e_{n+1}| \leq \varepsilon\, |e_n|^2$ and thus often converges faster than CG methods. Unfortunately, it is expensive because we need to compute the Hessian matrix .

Improve Quasi-Newton Algorithms

The one drawback related with this method is :
the exact evaluation of the Hessian matrixiscomputationally intensive, of order O(PW^2), P being the number of training vectors. The computation of the inverse HessianH^{-1} is even more computationally intensive, i.e. O (W^3). The aim of this paper is to solve these drawbacks.

Decrease Calculating of Hessian Matrix

Considering the sum-of-squares error function for pattern vector x_p(input data) :

$$E_p = ½ \sum_{i=1}^{k}[(y_a(x_p))_i - (y_{tp})_i]^2 = ½ [y_a(x_p) - y_{tp}]^T [y_a(x_p) - y_{tp}]$$

then the Hessian is calculated immediately as :

$$\partial^2E_p/(\partial w_{ji}\partial w_{kl}) =\sum_{i=1}^{k}(\partial y_{ai}/\partial w_{ji})(\partial y_{ai}/\partial w_{kl}) + \sum_{i=1}^{k}(y_{ai} - y_{ti})(\partial^2 y_{ai}/ (\partial w_{ji}\partial w_{kl}))$$

Considering a well trained network and the a mount of noise small then the terms ($y_{ai} - y_{ti}$) have to be small and may be neglected ; then:

$$\partial^2E_p / (\partial w_{ji}\, \partial w_{kl}) \approx \sum_{i=1}^{k} (\partial y_{ai} /\partial w_{ji})(\partial y_{ai} /\partial w_{kl}) \quad , \quad (9)$$

and $\partial y_{ai} /\partial w_{ji}$ may be found by back propagation procedure .

Decrease Calculating for Inverse of Hessian Matrix

Let consider the derivatives with respect to the weights denoted by $g = \{\partial / \partial w_{ji} \}_{ji}$, then from equation (9) the Hessian matrix may be written as a square matrix of order W x W and :

$$H_p= \sum_{p=1}^{P} g\, g^T E_p$$

where P is the number of vectors in the training set .
Now , by adding a new training vector : $H_{p+1} = H_p + g\, g^T E_{p+1}$

Then by using the following formula which may be decreased the calcultion of the inverse Hessian matrix is:
$(A+BC)^{-1} = A^{-1} - A^{-1} B (I+ BA^{-1}C)^{-1} C A^{-1}$, (where A,B and C are any three matrices and A is inversable).

By putting $H_p = A$, $g = B$ and $g^TE_{p+1} = C$, then we have :

$$H_{p+1}^{-1} = \mathbf{(H_p + g\, g^T E_{p+1})^{-1} = H_p^{-1} - H_p^{-1}\, g\, (I+ g\, H_p^{-1}g^T E_{p+1})^{-1}g^T E_{p+1}\, H_p^{-1}}$$

Using the above formula and starting with $H_0 = I$, the Hessian matrix may be computed by just one pass trough all training set .

Conclusion

The improve trained FFNN algorithms has principle advantages that
Computational complexity does not increase with the increase of the number of sampling points and provides rapid calculation for solving any problem at any given input points .

References

1.Galushkin, I. A.(2007). "Neural Networks Theory", Berlin Heidelberg,

2.Hristev, R. M.(1998) " The ANN Book ", Edition 1,.

3.Villmann,T.Seiffert, U. and Wismüller, A.(2004). " Theory and Applications of Neural maps ", ESANN2004 PROCEEDINGS - European Symposium on Ann,.25 - 38, April

4.Tawfiq, L.N.M. andNaoum , R.S. (2005)" On Training of Artificial Neural Networks " , AL-FathJornal , No 23, .

5.Jabber , A. K.(2009) " On Training Feed Forward Neural Networks for Approximation Problem ", MSc Thesis, Baghdad University, College of Education (Ibn Al-Haitham),.

6.Stanevski,N. and Tsvetkov, D.(2004). "On the Quasi-Newton Training Method for Feed-Forward Neural Networks", International Conference on Computer System and Technologies,

تسريع خوارزميات التدريب للشبكات العصبية ذي التغذية التقدمية

لمى ناجي محمد توفيق
ياسين عادل عريبي
قسم علوم الرياضيات / كلية التربية للعلوم الصرفة (ابن الهيثم) / جامعة بغداد

استلم البحث في:4 تشرين الاول 2011 ، قبل البحث في: 16 تشرين الثاني 2011

الخلاصة

الهدف من هذا البحث هو مناقشة بعض خوارزميات التدريب ذي الأداء العالي التي تصنف إلى صنفين رئيسين، الصنف الأول يستخدم التقنية الإرشادية التي طورت من خلال تحليل أداء خوارزمية انحدار الميل القياسي، الصنف الاخر لتسريع الخوارزميات يكون باستخدام تقنيات الأمثلية العددية القياسية مثل شبيه نيوتن . و الهدف الأخر من البحث هو حل العوائق المتعلقة بخوارزميات التدريب السابقة واقتراح خوارزميات تدريب كفوءة للشبكات العصبية ذي التغذية التقدمية .

الكلمات المفتاحية:.Artificial neural network, Feed Forward neural network, Training Algorithm