IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 1, Ver. II (Jan. 2014), PP 31-35
www.iosrjournals.org

Comparison of Neural Network Training Functions for Hematoma Classification in Brain CT Images

Bhavna Sharma[1] and Prof. K. Venugopalan[2]
[1]*(Ph.D Scholar, Computer Science Dept., Mohanlal Sukhadia University, Udaipur, India)*
[2]*(Professor, Computer Science Dept., Mohanlal Sukhadia University, Udaipur, India)*

Abstract:*Classification is one of the most important task in application areas of artificial neural networks (ANN).Training neural networks is a complex task in the supervised learning field of research. The main difficulty in adopting ANN is to find the most appropriate combination of learning, transfer and training function for the classification task. We compared the performances of three types of training algorithms in feed forward neural network for brain hematoma classification. In this work we have selected Gradient Descent based backpropagation, Gradient Descent with momentum, Resilence backpropogation algorithms. Under conjugate based algorithms, Scaled Conjugate back propagation, Conjugate Gradient backpropagation with Polak-Riebreupdates(CGP) and Conjugate Gradient backpropagation with Fletcher-Reeves updates (CGF).The last category is Quasi Newton based algorithm, under this BFGS, Levenberg-Marquardt algorithms are selected. Proposed work compared training algorithm on the basis of mean square error, accuracy, rate of convergence and correctness of the classification. Our conclusion about the training functions is based on the simulation results.***

Keywords: *Artificial Neural Network, Back propagation, Gradient Descent, Levenberg-Marquardt.*

I. Introduction

Classification is one of the most commonly encountered decision making tasks in medical image analysis [1]. Brain hematoma is caused due to a sudden stroke to a person after blood leaks out from the blood vessels in the brain. Brain hematomas can be epidural hematoma (EDH), subdural hematoma (SDH) and intracerebral hematoma (ICH). All these hematomas are hyper dense in nature and brighter than the other brain tissue having different shapes. Classification of these types of hematomas can be done by ANN [2]. Classification problem can be solved ANN mathematically and in a non-linear fashion.

An ANN is a biologically inspired computational model composed of various processing elements called artificial neurons. They are connected with coefficients or weights which constructs the neural network's structure [3]. The processing elements have weighted inputs, transfer function and outputs for processing information. There are many types of neural networks with different structures have been designed, but all are described by the transfer functions used in processing elements(neurons), the way of training given or learning rule and by the connection formula.ANN is composed of single-layer or multiple layer neurons. For complex problems multilayer perceptron (MLP) is the best model as it overcomes the drawback of the single-layer perceptron by the adding more hidden layers. In a feedforward multilayer perceptron network the inputs signals are multiplied by the connection weights are first summed and then directed to a transfer function to give output for that neuron. The transfer function (purelin, hardlim,sigmoid, logistic) executes on the weighted sum of the neuron's inputs.

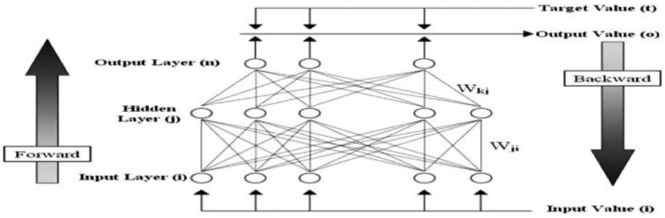

Fig.1 : Multilayer perceptron Neural network

A neural network is trained with input and target pair patterns with the ability of learning. In MLP network, backpropagation (BP) learning algorithm is used [4]. Inputs data is fed forward through the network to optimize the weights between neurons. Adjustment of the weights is done by backward propagation of the error during training phase. The network takes the input and target values in the training data set and changes the value of the weighted links to reduce the difference between the output and target values. The error is minimized across many training cycles called epoch. During each cycle network reaches to specified level of accuracy. The number of processing elements per layer, as well as the number of layers, greatly affects the abilities of the

MLP. Too few of them can slow down the learning process, and too many of them can alter the generalizing abilities of the MLP due to overfitting or memorization of the training data set [5].

To classify the types of hematoma from brain CT scan images first step is to preprocess the image to reduce the noise, second step is to segment the image using appropriate method [6] than in the next step we extracted statistical, shape and texture based features. The aim of feature extraction is to measure certain properties, attributes in original data that distinguish one input pattern from another pattern. In total 16 features are extracted from segmented image [7]. These features are fed to multilayer perceptron neural network.

In this paper, we focused on the different training algorithms that can be applied on the set of input data patterns. The MLP has to be trained to attain the desired output according to the training patterns or examples. Main objective is to find out the best training functions for classification of brain hematomas. For performance evaluation of training functions parameters are recognition accuracy, speed of training, correctness. One of the most important parameter is the mean squared error

$$\text{MSE} = \frac{\sum_{i=1}^{N}(y_i - o_i)^2}{N} \quad (1)$$

Where y_i is the target and o_i is the observed output and N is the number of data set.

II. Training Algorithms

There are number of batch training algorithms which can be used to train a network. Here, three types of training algorithms having eight training functions have been evaluated for classification of brain hematoma. They are Gradient Descent algorithms (traingd, traingdm, trainrp), Conjugate Gradient algorithms (trainscg, traincgf, traincgp), Quasi-Newton algorithms (trainbfg,trainlm) [8].

2.1 Gradient Descent algorithms

These are the most popular training algorithms that implements basic gradient descent algorithm and updates weights and biases in the direction of the negative gradient of the performance function.

2.1.1 Gradient Descent backpropagation algorithm (traingd) is a gradient descent local search procedure. It measures the output error, calculates the gradient of the error by adjusting the weights in the descending gradient direction.

2.1.2 Gradient Descent with Momentum (traingdm) algorithm is steepest descent with momentum that allows a network to respond to the local gradient as well as recent trends in the error surface. It acts like a lowpass filter that means with momentum the network ignores small features in the error surface. A network can get stuck in to a shallow local minimum but with momentum it slides through such local minimum [9].

2.1.3 Resilence backpropogation (trainrp) training algorithm eliminates the effects of the magnitudes of the partial derivatives [10]. In this sign of the derivative is used to determine the direction of the weight update and the magnitude of the derivative have no effect on the weight update. The size of the weight change is determined by a separate update value. The update value for each weight and bias is increased by a factor whenever the derivative of the performance function with respect to that weight has the same sign for two successive iterations [11]. The update value is decreased by a factor whenever the derivative with respect that weight changes sign from the previous iteration. If the derivative is zero, then the update value remains the same. Whenever the weights are oscillating weight change will be reduced.

2.2 Conjugate Gradient algorithms

The basic gradient descent algorithm adjusts the weights in the negative of the gradient, the direction in which the performance function is decreasing most rapidly. This does not necessarily produce the fastest convergence. In the conjugate gradient algorithms a search is performed along conjugate directions, which produces generally faster convergence than steepest descent directions. The conjugate gradient algorithms require only a little more storage than the other algorithms. Therefore, these algorithms are good for networks with a large number of weights [12].

2.2.1 Scaled Conjugate Gradient (trainscg*)* does not require line search at each iteration step like other conjugate training functions. Step size scaling mechanism is used which avoids a time consuming line search per learning iteration. This mechanism makes the algorithm faster than any other second order algorithms. The trainscg function requires more iteration to converge than the other conjugate gradient algorithms, but the number of computations ineach iteration is significantly reduced because no line search is performed [13].

2.2.2 Conjugate Gradient backpropagation with Fletcher-Reeves Updates (traincgf) is the ratio of the norm squared of the current gradient to the norm squared of the previous gradient. The conjugate gradient algorithms are usually much faster than other algorithms but the result depends on the problem [14].

2.2.3 Conjugate Gradient backpropagation with Polak-Riebre Updates (traincgp) is the ratio of the inner product of the previous change in the gradient with the current gradient to the norm squared of the previous gradient. The storage requirements for Polak-Ribiére (four vectors) are slightly larger than for Fletcher-Reeves [15].

2.3 Quasi-Newton algorithms

Newton's method gives better and fast optimization than conjugate gradient methods. The basic step of Newton's method is the Hessian matrix (second derivatives) of the performance index at the current values of the weights and biases. Newton's method converges faster than conjugate gradient methods but these methods are complex and take more time to compute the Hessian matrix for feed forward neural networks. Based on Newton's method which doesn't require calculation of second derivatives is called quasi-Newton or secant method. They update an approximate Hessian matrix in each iteration of the algorithm.

2.3.1 BFGS(Broyden–Fletcher–Goldfarb–Shanno) (trainbfg) algorithm approximates Newton's method, a class of hill-climbing optimization techniques that seeks a stationary point of a function. For such problems, a necessary condition for optimality is that the gradient be zero [16]. This algorithm requires more storage and computation than the conjugate gradient methods, but it converges in fewer iterations. BFGS have good performance even for non smooth optimizations and an efficient training function for smaller networks.

2.3.2 Levenberg–Marquardt backpropagation (trainlm) algorithm locates the minimum of a multivariate function that can be expressed as the sum of squares of non-linear real-valued functions. It is an iterative technique that works in such a way that performance function will always be reduced in each iteration of the algorithm. This feature makes trainlm the fastest training algorithm for networks of moderate size. Similar to trainbfg, trainlm function has drawback of memory and computation overhead caused due to the calculation of the gradient and approximated Hessian matrix [17].

III. Experimental Results And Discussion

All these experiments were carried out on windows 7 (32-bit) operating system with i5 processor and 4 GB RAM. All training functions used are coded in MATLAB using ANN toolbox. The experiment data consists of 100 brain CT images. The selected images are of same quality but with different resolutions. Images have various types of hematoma with different sizes, shapes and at different locations in brain. The sample data of 100 images having 16 features for each image were presented to the ANN. For learning process, data was divided into sets for training (70%), validation (15%) and for testing (15%). To avoid possible bias in the presentation order of the sample patterns to the ANN these sample sets were randomized. Sigmoid transfer function is used for the hidden layer.

Basic system training parameters are max_epochs=1000, show=5, performance goal=0, time=Inf, min_grad=1e-010, max_fail=6 are fixed for each training function. The parameters for comparison are CPU time elapsed, no of epoch (E) at the end of training, correct classification (C) percentage, Regression (R) on training, R on validation. All these parameters are checked for 10, 20 and 30 number of neurons (H) in hidden layer. The network is trained until the mean squared error is less than 0.0.

Table 1: Comparison of Training Functions

Algorithm	Trainingfunction	H	Best validation MSE at epoch	Epoch	Classification %	R on training	R on validation
Gradient Descent	traingd	10	0.20919at 1000	1000	73%	0.2045	0.3385
		20	0.13524at 1000	1000	72%	0.6887	0.6830
		30	0.27341at 1000	1000	73%	0.4303	0.2693
	traingdm	10	0.15082at 1000	1000	82%	0.5975	0.5676
		20	0.13215at 1000	1000	88%	0.6080	0.6556
		30	0.10244at881	1000	88%	0.5442	0.7620
	trainrp	10	0.000166 at 30	36	95%	0.9976	0.9952
		20	0.201113 at 33	39	97%	0.9986	0.9349
		30	0.027887 at29	35	95%	0.9981	0.9203
Conjugate Gradient	trainscg	10	0.003119at26	32	99%	1	0.9985
		20	0.00430at 33	39	98%	0.9973	0.9847
		30	0.02733 at 40	46	98%	0.9988	0.8959
	traincgp	10	0.21541at 16	20	56%	0.1683	0.1304
		20	0.24983at 20	26	57%	0.1674	0.1279
		30	0.01003 at 50	56	56%	0.0392	0.0760
	traincgf	10	0.00008at 79	85	67%	0.4427	0.5997
		20	0.93732at 34	35	66%	0.2214	0.4818
		30	0.13689at 26	32	67%	0.5582	0.6905
Quasi Newton	trainbfg	10	0.45559at 18	24	62%	0.3848	0.8121
		20	0.10043at 13	19	62%	0.8573	0.7780
		30	0.00722 at30	36	68%	0.8554	0.8653
	trainlm	10	0.13564 at 10	16	97%	1	0.9989
		20	0.00550 at 11	17	99%	1	0.9686
		30	0.00016at40	44	99%	0.9128	0.9985

CPU time elapsed at the end of training is only few seconds for each algorithm. traingdm is usually faster than simple gradient descent, traingd, because while maintaining stability it allows higher learning rates. Memory requirements of trainrp function are relatively small and much faster than standard gradient descent algorithms. The trainscg and trainlm are almost as fast as trainrp. Other training function requires more time than these functions. For small networks, trainbfg is an efficient training function and converges in a few iterations. Similarly trainlm and trainscg converges in lesser number of iterations than the all other training functions. Within 30 epochs (iteration), trainbfg and trainlm and trainscg achieve the performance goal while other function takes more epochs. Table 1, provides detailed information on the training performances of these functions.

Networks simulated using various training functions are affected according to the number of neurons in their hidden layer. Functions trainrp and trianscg are not much affected by increasing no of neurons in hidden layers. Table 1 shows the number of iterations (epochs) at the end of training and also the best validation performance (MSE) at epoch. Convergence rate increases or more epochs required for each training function as number of neurons are increase in hidden layer.

The confusion matrix gives the percentage of correct and incorrect classifications in the simulated feed-forward back propagation networks. Table 1 shows the overall classification percentage for each training function. Among those trainrp, trainscg and trainlm classification percentage is acceptable but trainlm gives high percentage than the other functions.

The Regression analysis function compares the actual outputs of the neural network with the corresponding desired outputs (targets). It returns the correlation coefficient (R) between them and also the slope and the intercept of the best-linear-fit equation. R can be in the range [0.0,1.0].The more the values of R are near to 1.0 and the more correct the response of the network. Regression value on training and validation in Table 1 clearly indicates trainrp, trainscg and trainlm training functions qualifies among other functions and suits for classification task. We further analyzed these functions on other parameters like increasing number of layers, reducing error goal and increasing the sample size of input patterns presented to the network. Increasing the number of layers in network does not affect performance or correct classification for all the training functions only the rate of convergence increases.

Table 2: MSE vs. number of epochs

Training function	1E-01	1E-02	1E-03	1E-04	1E-05
trainrp	39	36	36	37	37
trainscg	22	18	20	18	20
trainlm	11	16	17	17	16

Table 3: Sample size vs. number of epochs

Training function	100	200	400	800	1600
trainrp	33	55	111	138	142
trainscg	39	47	72	92	100
trainlm	11	18	22	24	44

For different values of MSE the number of epochs used to train network was tabulated in Table 2. It is clear from the table that if we decrease the performance goal or mean square error value, the number of epochs does not vary much and best performance remains within a range for all the functions. If we keep trying to reduce the error goal there is problem of overfitting in all the training functions. From the table it is clear that trainlm needs least number of epochs to converge.

Doubling the size of dataset as shown in Table 3 from 100 samples to 1600, we found that rate of convergence increases for all the training functions. It is almost doubled in trainrp, increases at slower rate in trainscg but very slowly in case of trainlm. We can say that trainlm fits for small as well as moderate size data and converges in less iteration and in very less time than others.

IV. Conclusion

Feed-forward back propagation neural network can be used as a highly effective tool for types of brain hematoma classification with appropriate combination of learning, transfer and training functions. The ANNs were simulated and trained with all the above mentioned algorithms using the training dataset. In the proposed work we found no significant differences between the correct classification percentage for trainrp, trainscg and trainlm, they are in acceptable range. The convergence speed of trainlm and trainscg are higher than other training functions. Comparing on epochs and MSE parameters, trainlm and trainscg outperformed other training function. Considering the sample size of input patterns we found that trainlm suits to larger data set. It converges in less number of iterations and in lesser time than the other training functions.

References

[1] Atam P. Dhawan, H. K. Huang, DaeShik Kim, Principles and advanced methods in medical imaging and image analysis(World Scientific, 2008).

[2] L. Fu., Neural Networks in Computer Intelligence (Tata McGraw-Hill, 2003).

[3] S. Haykin, Neural Networks- A Comprehensive Foundation (2nd ed., Pearson Prentice Hall, 2005).

[4] R. Roja, The Backpropagation Algorithm, Chapter 7: Neural Networks (Springer-Verlag, Berlin, 1996) pp. 151-184.

[5] M. K. S. Alsmadi, K. B. Omar, S. A. Noah, "Back propagation algorithm: The best algorithm among the multi-layer perceptron algorithm", International Journal of Computer Science and Network Security, vol., 9(4), 2009, pp. 378 – 383.

[6] Bhavna Sharma, Prof K. Venugopalan, "Performance comparison of standard segmentation techniques for brain CT images", International Journal of Computer Engineering and Technology (IJCET), vol. 3(1), Jan-June 2012, pp.126-134.

[7] Bhavna Sharma, Prof K. Venugopalan, "Classification of hematomas in brain CT images using neural network", Proceedings of IEEE Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT-2014) (Accepted).

[8] S. Ali and K. A. Smith, "On learning algorithm selection for classification",Applied Soft Computing, (6), 2006,pp.119–138.

[9] M. Beale, M. Hagan, H. Demut, Neural Network Toolbox User's Guide, 2010.

[10] A.D. Anastasiadis, G.D. Magoulas, and M.N. Vrahatis, "New globally convergent training scheme based on the resilient propagation algorithm",Neurocomputing, 64, 2005, pp.253–270.

[11] http://www-rohan.sdsu.edu/doc/matlab/toolbox/nnet/backpr57.html

[12] W.W. Hager and H. Zhang, "A survey of nonlinear conjugate gradient methods", Pacific of Journal Optimization, 2:35, 2006, pp. 35–58.

[13] M. F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning", Neural Networks, 6, 1993, pp. 525–533.

[14] R Fletcher , C. M. Reeves, Computer journal ,vol. 7, 1964, pp.149-153

[15] Demuth, Howard, Mark Beale, and Martin Hagan, "Neural Network Toolbox™ 6", User Guide, COPYRIGHT2008.

[16] http://www.mathworks.in/help/nnet/ref/trainbfg.html

[17] D.Pham, S. Sagiroglu, "Training multilayered perceptrons for pattern recognition: a comparative study of four training algorithms", International Journal of Machine Tools and Manufacture, vol.41, 2001, pp. 419–430.