**Comparison of NeuralNetwork Training Functions for character Classification in handwritten text Images**

**M.SAI CHAITHANYA 15CSE1019**

**Problem Statement:**

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 two types of training algorithms in feed forward neural network for hand written characters 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) . 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.

**1. INTRODUCTION**

Handwriting recognition has been one of the most fascinating and challenging research areas in field of image processing and pattern recognition in the recent years . It contributes immensely to the advancement of an automation process and can improve the interface between man and machine in numerous applications. Several research works have been focusing on new techniques and methods that would reduce the processing time while providing higher recognition accuracy.

In the off-line recognition, the writing is usually captured optically by a scanner and the completed writing is available as an image. In the off-line systems, the neural networks have been successfully used to yield comparably high recognition accuracy levels .Several applications including mail sorting, bank processing, document reading and postal address recognition require off-line handwriting recognition systems. As a result, the off-line handwriting recognition continues to be an active area for research towards exploring the newer techniques that would improve recognition accuracy .

The first important step in any handwritten recognition system is pre-processing followed by segmentation and feature extraction.

An artificial neural Network as the backend is used for performing classification and recognition tasks. In the off-line recognition system, the neural networks have emerged as the fast and reliable tools for classification towards achieving high recognition accuracy .

**2. 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) .

**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 .

**2.1.3 Resilence backpropogation (trainrp)** training algorithm eliminates the effects of the magnitudes of the partial derivatives . 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 . 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 .

**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 .

**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 .

**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 .

**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 . 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 .

**3.Tools Used**

* MATLAB (R2017a)

**4. Experimental Results**

All these experiments were carried out on windows 8.1 (64-bit) operating system with i3 processor and 4 GB RAM. All training functions used are coded in MATLAB using ANN toolbox. The experiment data consists of 650 handwritten character images(25x26). The selected images are of same quality but with different styles. The sample data of 650 images having 108 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=10, performance goal=0, time=Inf, memory Reduction = 100, min\_grad=1e-5, lr=0.9,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 35, 44 and 53 number of neurons (H) in hidden layer. The network is trained until the mean squared error is less than 0.0.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Trainingf unction | H | Best validation MSE at epoch | Epoch | Classification % | R on training | R on validation |
| Gradient Descent | traingd | 35 | 0.26454 at 1000 | 1000 | 50 | 0.44512 | 0.32239 |
| 44 | 0.26299 at 1000 | 1000 | 75 | 0.46254 | 0.39753 |
| 53 | 0.26224 at 1000 | 1000 | 18.75 | 0.48545 | 0.43277 |
| traingdm | 35 | 0.26911 at 23 | 29 | 12.5 | 0.018775 | 0.018603 |
| 44 | 0.26881 at 16 | 22 | 0 | 0.056663 | 0.053939 |
| 53 | 0.26987 at 4 | 10 | 6.25 | 0.0096828 | 0.013104 |
| trainrp | 35 | 0.26964 at 6 | 12 | 0 | 0.069769 | 0.045015 |
| 44 | 0.2697 at 0 | 6 | 6.25 | 0.0048326 | 0.028884 |
| 53 | 0.26904 at 8 | 14 | 0 | 0.083707 | 0.08845 |
| Conjugate Gradient | trainscg | 35 | 0.25125 at 90 | 96 | 75 | 0.82883 | 0.74881 |
| 44 | 0.25669 at 78 | 84 | 75 | 0.76928 | 0.59179 |
| 53 | 0.25492 at 65 | 71 | 68.75 | 0.75031 | 0.64919 |
| traincgp | 35 | 0.27066 at 0 | 2 | 6.25 | -0.020861 | -0.012385 |
| 44 | 0.25708 at 52 | 58 | 81.25 | 0.76074 | 0.57613 |
| 53 | 0.27316 at 1 | 2 | 6.25 | 0.004102 | -0.017537 |

We have used the following figure as test input image to calculate % classification.



**5.Conclusion**

According to the experimental results the best output we got was using traincgp training algorithm (81.2%) which is a Conjugate Gradient. But we can’t base our results for just one single input image.

**6.References**

1. 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’.

2. ‘Handwritten Character Recognition of South Indian Scripts: A Review’

By Jomy John, Pramod K. V, Kannan Balakrishnan