Performance Analysis of Gradient Descent with Momentum & Adaptive Back Propogation and BFGS Quasi-newton Back Propagation of Multilayer Feed Forward Neural Networks for Hand Written Hindi Characters of Swars

Rajesh Lavania*
Jawahar Thakur**
Manu Pratap Singh***

Abstract

In this paper we are analyzing the performance of multi layer feed forward Neural Networks with Gradient descent with momentum & adaptive back propagation (TRAINGDX) and BFGS quasi-Newton back propagation (TRAINBFG) for Hand written Hindi Characters of SWARS. In this analysis, five Hand written Hindi characters of SWARS from different people are collected and stored as an image. The MATLAB function is used to determine the densities of these scanned images after partitioning the image into 16 portions. These 16 densities for each character are used as an input pattern for the two different Neural Network architectures. The two learning rules as the variant of Back Propagation learning algorithm are used to train these Neural Networks. The performances of these two Neural Networks are analyzed for convergence and trends of error in the case of non convergence. There are some interesting and important observations have been considered for trends of error in the case of non convergence. The inheritance of local minima problem of back propagation algorithm is effecting massively to these two proposed learning algorithm also.

Keywords: Back propagation Algorithm, Multilayer Neural Networks, Gradient Descent, and Pattern Recognition.

1. Introduction

Recognition is a basic property of all human beings; when a person sees an object, he or she first gathers all information about the object and compares its properties and behaviors with the existing knowledge stored in the mind. If we find a proper match, we recognize it [1]. The concept of recognition is simple in the real world environment, but in the world of computer science, recognizing any object is an amazing feat. The functionality of the human brain is amazing; it is not comparable with any machines or software. The act of recognition can be divided into two broad categories namely (a) Concrete item recognition, it involves the recognition of spatial samples such as fingerprints, weather maps, pictures and physical objects and the recognition of temporal samples such as, waveforms and signatures. (b) Abstract item recognition, it involves the recognition of a solution to a problem, an old conversation or argument.

More generally the pattern recognition spans a number of scientific disciplines, uniting them in search for a solution to the common problem of recognizing the pattern of a given class and assigning the name of identified class. Pattern recognition is the categorization of input data into identifiable classes through the extraction of significant attributes of the data from

*Department of Computer Science Engineering, I E T, Khandari Campus, Dr. B.R. Ambedkar University, Agra, India.
**Department of Computer Science, H.P. University, Summerhill, Shimla, India.
***Department of Computer Science, Institute of Computer & Information Science, Dr. B.R. Ambedkar University, Khandari, Agra, India.

irrelevantbackground detail. A pattern class is a category determined by some common attributes. Therefore, a pattern is the description of a category member representing a pattern class. A pattern class is a family of patterns that shares some common properties. Pattern recognition by machine involves techniques for assigning patterns to their classes automatically with as little human interventions is possible. Pattern recognition aims to classify data (patterns) based on either a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space [3]. In the literature the complete pattern recognition system is defined with the components like; sensor that gathers the observations to be classified or described; a feature extraction mechanism that computes numeric or symbolic information from the observations; and a classification or description scheme that does the actual job of classifying or describing observations, relying on the extracted features. A very common domain in computer science for the pattern recognition is identified in the terms of character recognition or classification in precise manner. Thus our basic purpose to investigate the available methods and techniques those are related with the character recognition process.

Character recognition plays an important role in today's life [4]. It can solve many complex problems in of real life. An example of character recognition is Handwritten English alphabets. The classic difficulty of being able to correctly recognize even typed optical language symbols is the complex irregularity among pictorial representations of the same character due to variations in fonts, styles and size. This irregularity undoubtedly widens when one deals with handwritten characters [5]. Classification method designs are based on the following concepts: Member-roster concept: Under this template-matching concept, a set of patterns belonging to a same pattern is stored in a classification system. When an unknown pattern is given as input, it is compared with existing patterns and placed under the matching pattern class. Common property concept: In this concept, the common properties of patterns are stored in a classification system. When an unknown pattern comes inside, the system checks its extracted common property against the common properties of existing classes and places the pattern/object under a class, which has similar, common properties. Clustering concept: Here, the patterns of the targeted classes are represented in vectors whose components are real numbers. So, using its clustering properties, we can easily classify the unknown pattern. If the target vectors are far apart in geometrical arrangement, it is easy to classify the unknown patterns. If they are nearby or if there is any overlap in the cluster arrangement, we need more complex algorithms to classify the unknown patterns [6-7]. There are various methods have been proposed for the character classification or recognition in the literature. Some of them have shown their effectiveness for this task. Initially the attempts to accomplish the task were basically confined in statistical domain like Bayesian decision theory [8] but these methods have their own pros and corns.

Bayesian decision theory is a system that minimizes the classification error. Bayesian decision theory has a conceptual clarity leading to an elegant numerical recipe. This algorithm can deal with a broader scope of stochastic models than classical algorithms. Nearest neighbor rule [9] is used to classify the handwritten characters. The distance measured between two character images is needed to use this rule. This algorithm works well when the target patterns are far apart. Training in the nearest neighbor rule is very fast. Linear classification or discrimination [7] deals with assigning a new point in a vector space to a class separated by a boundary. It is well suited to mixed data types. It can also handle non-linear cases and missing data. The results produced by the system are very easy to interpret. All the methods mentioned above have their limitations such as Bayesian decision theory has computational difficulties. This means that the method has a difficulty filling in numerical details. The method also has an obligation to use prior information; if unavailable the theory will not work properly. The basic problem with the nearest neighbor rule is it requires the large set of data and the query time is very slow. Because of the larger set of data it is very much prone to data error. Therefore if there were any irrelevant information entered into the system the system will be easily misinterpret the results. The same problem of larger data set occurs with the linear classification method [10].

The task of character recognition can easily be accomplished by a human being without involving much effort due to its complex structure and working in parallel mechanism. The structure of biological neural network [11] has been simulated and modeled in a serial fashion that provides parallelism through ANN. One of the important advantages of ANN is its adaptive nature [12] and due to this property many existing paradigms can be fused into it easily. The powerful attribute of neural network is the ability to learn arbitrary non-linear mapping using one of the appropriate learning rules. Once the ANN system has trained, it can use for the pattern classification [13], pattern association, pattern mapping, pattern grouping [13], and feature mapping pattern and optimization control etc. Accomplish the task of pattern classification & pattern mapping the supervised multilayer feed forward neural network [14, 15] is considered with non-linear differentiable function in all processing units of output and hidden layers. The number of processing units in the input layer, corresponds to the dimensionalities of the input pattern, are linear. The number of output units corresponds to the number of distinct classes in the pattern classification. A method has been developed [16], so that network can be trained to capture the mapping explicitly in the set of input output pattern pair collected during an experiment and simultaneously expected to modal the unknown system for function from which the predictions can be made for the new or untrained set of data. The possible output pattern class would be approximately an interpolated version of the output pattern class corresponding to the input learning pattern close to the given test input pattern. This method involves the back propagation-learning rule [17] based on the principle of gradient descent along the error surface in the weight space. This algorithm is used for the training of a supervised multi-layer feed¬ forward neural network, so that the network could be trained to capture the missing implicit pattern and generate the classification for different features in the given set of input-output pattern pairs.

Character classification problem is related to heuristic logic as human beings can recognize characters and documents by their learning and experience. Hence neural networks which are more or less heuristic in nature are extremely suitable for this kind of

problem. Various types of neural networks are used for OCR classification. But it has been found that most of the work for character classification or recognition using neural network techniques is concentrated on English character recognition [49]. English Character Recognition (CR) has been extensively studied in the last half century and progressed to a level, sufficient to produce technology driven applications. But same is not in the case of Indian languages which are complicated in terms of structure and computations. Rapidly growing computational power may enable the implementation of Indic CR methodologies. Digital document processing is gaining popularity for application to office and library automation, bank & postal services, publishing houses & communication technology. The study investigates the direction of the Devnagari Optical Character Recognition research (DOCR), analyzing the limitations of methodologies for the system which can be classified based upon two major criteria: the data equation process (on-line or off-line) and the text type (machine- printed or hand-written).No matter which class the problem belongs, in general there are five major stages in DOCR problem:

1. Pre-processing.
2. Segmentation.
3. Feature Extraction.
4. Recognition.
5. Post processing

These methods are very common and apply for almost every method or technique which is used for character recognition apart from the language. Handwriting Recognition technology has been improving much under the purview of pattern recognition and image processing since a few decades. Hence various soft computing methods involved in other types of pattern and image recognition can as well be used for DOCR. Seminal and comprehensive work in DOCR has been described [18-24]. A general review of Statistical Pattern Recognition can also be found in [25-28]. These can be taken as good starting point to reach the resent studies in various types and applications of DOCR problems. An excellent overview of document analysis can also be found in [29].

In this present paper we are exploring the necessary steps those have been discussed above for the implementation and analysis of the feed forward multilayer neural networks for the hand written Hindi characters of SWARS. In our proposed method the Multilayer feed forward neural networks will train with two learning algorithms those are the variant of generalized delta learning rule namely Quasi-Newton backpropagation learning algorithm and Gradient descent with momentum and adaptive backpropagation method for the training set of the handwritten Hindi characters of SWARS. These algorithms are used to analysis the performance of neural networks for the given training set. This paper represents the analytic study for the behavior of neural networks system. The analytic study indicates the some important observation and characteristics about the nature of error in the case of non convergence.

Section 2 of this paper describes the features of Hindi characters of SWARS those are required for the feature extraction. The section 3 discusses the approach which is used for the feature extraction of the input stimuli used for the training. Section 4 for this paper describes the generalized delta learning rule in detail and the description of used learning algorithms. These learning algorithms are the variant of backpropagation or generalized delta learning rule. The description of these methods represents the implementation details of these methods which are used in the above paper. The section 5 describes the architecture and design of the network used, in terms of simulation design. Section 6 shows the results of the experiments with the discussion. Section 7 describes the conclusion and the future work based on this analysis.

2. Features of Hindi Swars Characters

India is a multi-lingual and multi-script country comprising of eighteen official languages. One of the defining aspects of Indian script is the repertoire of sounds it has to support. Because there is typically a letter for each of the phonemes in Indian languages, the alphabet set tends to be quite large. Most of the Indian languages originated from BRAMHI script. These scripts are used for two distinct major linguistic groups, Indo-European languages in the north, and Dravidian languages in the south [30]. DEVNAGARI is the most popular script in India. It has 11 vowels and 33 consonants. They are called basic characters. Vowels can be written as independent letters, or by using a variety of diacritical marks which are written above, below, before or after the consonant they belong to. When vowels are written in this way they are known as modifiers and the characters so formed are called conjuncts. Sometimes two or more consonants can combine and take new shapes. These new shape clusters are known as compound characters. These types of basic characters, compound characters and modifiers are present not only in DEVNAGARI but also in other scripts. Hindi, the national language of India, is written in the DEVNAGARI script and also Hindi is the third most popular language in the world [28] which consists with SWARS and VYANJANS. There are 13 SWARS and 34 VYANJANS. In this paper we are concerning about the recognition of SWARS characters of Hindi language. So, our domain of problem is restricted only for the description of SWARS. A sample of Hindi SWARS characters set is provided in table 1.

Table 1: Characters in Hindi SWARS.

SWARS	अ	आ	इ	ई	उ	ऊ	ऋ	ए	ऐ	ओ	औ	ं	अः

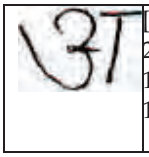
All the characters have a horizontal line at the upper part, known as SHIROREKHA or headline. No English character has such characteristic and so it can be taken as a distinguishable feature to extract English from these scripts. In handwritten form, from left to right direction, the SHIROREKHA of one character joins with the SHIROREKHA of the previous or next character of the same word. In this fashion, multiple characters and modified shapes in a word appear as a single connected component joined through the common SHIROREKHA. All the characters and modified shapes in a word appear to hang from the hypothetical SHIROREKHA of the word. Also in DEVNAGARI there are vowels, Consonants, vowel modifiers and compound characters, numerals. Moreover, there are many similar shaped characters. All these variations make DOCR, a challenging problem [31]. As for as concern with the Hindi language characters of SWARS the features are as same as of any DEVNAGARI script. So that, these features can consider in the same form.

3. Feature Extraction

Feature extraction and selection can be defined as extracting the most representative information from the raw data, which minimizes the within class pattern variability while enhancing the between class pattern variability. For this purpose, a set of features are extracted for each class that helps distinguish it from other classes, while remaining invariant to characteristic differences within the class.

In this present paper we have considered the feature extraction from the input stimuli by using density function of MATLAB. In our approach we have considered the input data in the form of five different set of each hand written SWARS of Hindi characters by five different peoples. It is quite natural that the five different people considered the different hand writing and different writing style for every character. So, in this way we

Figure 1. Scanned images of five different samples of handwritten Hindi SWARS



[2.514091; 2.135156; 2.292104; 2.433057; 2.016629; 2.177502; 2.102489; 2.065017; 1.924126; 2.103291; 2.158773; 2.246736; 1.893306; 2.180586; 2.324606; 2.457668]

Figure 2. The input pattern vector of order (16X1) for character v from MATLAB function

have total 65 samples of the character sets. Each character set is containing different examples of same sample in different hand writing as shown in figure 1. Now to prepare our training set of input output pattern pairs, we consider each scanned hand written character as a bit map image. This bitmap image of a character is now partitioned in 16 equal parts. After this the row wise & column wise mean of each partition is obtained by using coded MATLAB function "Mean_character_recognition.m", So that by doing this we obtained 16 real number values for each scanned image. Hence every scanned image is now considered in the form of input pattern vector of 16 dimensions. Thus we consider each input pattern vector in the form of 16X1 row matrix form. The example of one such character & its representation in input pattern vector form can be shown in figure 2.

So that in this way we can determine the input pattern vector for every scanned image. Thus we have the training set which consist with 65 input pattern vector of size 16X1. It means if we consider the entire training set as matrix of training pattern than it will be the order of 16X65. To distinguish each character set from other character set for classification during the training the target output is needed. Therefore to classify these characters of Hindi swar there must be 13 different classes. As we know the single neuron can differentiate between two classes so to differentiate 13 different classes we required four output neurons. Hence the target output pattern for each input pattern will be of dimension four. In this proposed method we are considering the target output pattern for each character in four bit binary form as shown in figure 3.

अ	आ	ठ	ई	उ	ऊ	ऋ	ए	ऐ	ओ	औ	ट	अः
0	0	0	0	0	0	0	1	1	1	1	1	1
0	0	0	1	1	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	0	0	1	1	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1

Figure 3: Target output pattern for different handwritten Hindi SWARS

Thus, we have constructed the training set of input output patterns pair to analyze the performance of multilayer feed forward neural networks with two mentioned algorithms. We have also constructed our test pattern set to verify the performance of networks. Our test pattern set consist with another set of hand written characters i.e. Two set each by two different people. Hence our test pattern set consist with 26 samples. Input pattern for these test character set are constructed in same manner as we did for training set pattern.

4. Neural Network for Character Recognition

Neural network architectures can be classified as, feed-forward and feedback (recurrent) networks. The most common neural

networks used in the OCR systems are the multilayer perceptron (MLP) of the feed forward networks and the Kohonen's Self Organizing Map (SOM) of the feedback networks. One of the interesting characteristics of MLP is that in addition to classifying an input pattern, they also provide a confidence in the classification [26]. These confidence values may be used for rejecting a test pattern in case of doubt. MLP is proposed by U. Bhattacharya et al. [32, 33]. A detailed comparison of various NN classifiers is made by M. Egmont-Petersen [34]. He has shown that Feed-forward, perceptron higher order network, Neuro-fuzzy system are better suited for character recognition [35]. K. Y. Rajput et al. [36] used back propagation type NN classifier. Genetic algorithm based feature selection and classification along with fusion of NN and Fuzzy logic is reported in English [37, 38] but no any work is reported for Indian languages. In this paper to accomplish the task of classification for Hindi characters of SWARS we are using the feed forward multilayer neural network with generalized delta learning rule and its two variants namely Quasi-Newton method and conjugate descent gradient with adaptive learning.

4.1 Generalized Delta Learning Rule

The backpropagation (BP) learning algorithm is currently the most popular supervised learning rule for performing pattern classification tasks [39, 40]. It is not only used to train feed forward neural networks such as the multilayer perceptran, it has also been adapted to recurring neural networks [41]. The BP algorithm is a generalization of the delta rule, known as the least mean square algorithm [39]. Thus, it is also called the generalized delta rule. The BP overcomes the limitations of the perceptron learning enumerated by Minsky and Papert [42]. Due to the BP algorithm, the MLP can be extended to many layers. The BP algorithm propagates backward the error between the desired signal and the network output through the network. After providing an input pattern, the output of the network is then compared with a given target pattern and the error of each output unit calculated. This error signal is propagated backward, and a closed-loop control system is thus established. The weights can be adjusted by a gradient-descent-based algorithm. In order to implement the BP algorithm, a continuous, nonlinear, monotonically increasing, differentiable activation function is required as Logistic Sigmoid function or hyperbolic tangent function.

We want to train a multi-layer feed forward network by gradient descent to approximate an unknown function, based on some training data consisting of pairs. The vector x represents a pattern of input to the network, and the vector z the corresponding desired output from the training set S. The objective function for optimization is defined in the sum of instantaneously squared error as.

$$E^P = \frac{1}{2}\sum_{j=1}^{J}(T_j - S_j)^2 \tag{4.1.1}$$

where $(T_j - S_j)^2$ is the squared difference between the actual output of the network on the output layer for the presented input pattern P and the target output pattern vector for the pattern P.

All the network parameters $W^{(m-1)}$ and θ^m, m = 2 · · · M, can be combined and represented by the matrix $W = [w_{ij}]$. The error function E can be minimized by applying the gradient-descent procedure as:

$$\Delta W = -\eta \frac{\partial E}{\partial W} \tag{4.1.10}$$

where η is a learning rate or step size, provided that it is a sufficiently small positive number.

Applying the chain rule the equation (4.1.10) can express as

$$\frac{\partial E}{\partial w_{ij}^{(m)}} = \frac{\partial E}{\partial u_j^{(m+1)}} \frac{\partial u_j^{(m+1)}}{\partial w_{ij}^{(m)}} \tag{4.1.11}$$

while

$$\frac{\partial u_j^{(m+1)}}{\partial w_{ij}^{(m)}} = \frac{\partial}{\partial w_{ij}^{(m)}}\left(\sum w_j^{(m)} o^{(m)} + \theta_j^{(m+1)}\right) = o_i^{(m)} \tag{4.1.12}$$

and

$$\frac{\partial E}{\partial u_j^{(m+1)}} = \frac{\partial E}{\partial o_j^{(m+1)}} \frac{\partial o_j^{(m+1)}}{\partial u_j^{(m+i)}} = \frac{\partial E}{\partial o_j^{(m+1)}} \phi_j^{(m+1)}\left(u_j^{(m+1)}\right) \tag{4.1.13}$$

For the output unit *m=M-1*

$$\frac{\partial E}{\partial o_j^{(m+1)}} = e_j \tag{4.1.14}$$

For the hidden units, *m = 1,2,3……,M - 2,*

$$\frac{\partial E}{\partial o_j^{(m+1)}} = \sum_{\omega=2}^{j_{m+2}} \frac{\partial E}{\partial u_\omega^{m+2}} \omega_{j\omega}^{m+1} \tag{4.1.15}$$

Define the delta function by

$$\delta_j^{(m)} = \frac{\partial E}{\partial u_p^{(m)}} \tag{4.1.16}$$

for *m = m = 2,3……,M*. By substituting (4.1.11), (4.1.15), and (4.1.16) into (4.1.13), we finally obtain the following.
For the output units, *m = M - 1,*

$$\delta_j^{(M)} = -e_j \phi_j^{(M)}\left(u_j^{(M)}\right) \tag{4.1.17}$$

For hidden units, *m = 1,……..,M - 2,*

$$\delta_j^{(M)} = -e_j \phi_j^{(M)}\left(u_j^{(M)}\right) \sum_{\omega=1}^{J_{m+2}} \delta_\omega^{(m+2)} \omega_\omega^{m+1} \tag{4.1.18}$$

Equations (4.1.17) and (4.1.18) provide a recursive method to solve $\delta_j^{(m+1)}$ for the whole network. Thus, W can be adjusted by

$$\frac{\partial E}{\partial \omega_{ij}^{(m)}} = -\delta_j^{(m+1)} o_i^{(m)} \tag{4.1.19}$$

For the activation functions, we have the following relations
For the logistic function

$$\phi(u) = \beta\phi(u)[1 - \phi(u)] \tag{4.1.20}$$

For the *tanh* function

$$\phi(u) = \beta\left[1-\phi^2(u)\right] \tag{4.1.21}$$

The update for the biases can be in two ways. The biases in the (m+1)th layer θ(m+1) can be expressed as the expansion of the weight W(m), that is, $\theta^{(m+1)} = \left(\omega_{0,1}^{(m)}, \ldots\ldots\ldots\ldots\ldots \omega_{0,J_{m+1}}^{(m)}\right)$.

Accordingly, the output o(m) is expanded into

$$o^{(m)} = \left(1, o_1^{(m)}, \ldots\ldots\ldots\ldots, o_{J_m}^{(m)}\right)$$

Another way is to use a gradient-descent method with regard to θ(m), by following the above procedure. Since the biases can be treated as special weights, these are usually omitted in practical applications. The algorithm is convergent in the mean if

$$0 < \eta < \frac{2}{\lambda_{\max}}$$

where λmax is the largest eigenvalue of the autocorrelation of the vector x, denoted as C [44]. When η is too small, the possibility of getting stuck at a local minimum of the error function is increased. In contrast, the possibility of falling into oscillatory traps is high when η is too large. By statistically preprocessing the input patterns, namely, de-correlating the input patterns, the excessively large eigenvalues of C can be avoided and thus, increasing η can effectively speed up the convergence. PCA preconditioning speeds up the BP in most cases, except when the pattern set consists of sparse vectors. In practice, η is usually chosen to be 0 < η < 1 so that successive weight changes do not overshoot the minimum of the error surface. The BP algorithm can be improved by adding a momentum term [17] it is known as Gradient Descent with momentum term. In our proposed analysis we are using this learning rule in MATLAB as TRAINGDX (Gradient descent with momentum & adaptive back propagation). As per this learning rule the weight update between output layer and hidden layer is represented by following weight updating equations as:

$$\Delta w_{ho}(s+1) = -\eta \sum_{i=1}^{H} \frac{\partial E}{\partial w_{ho}} + \alpha \Delta w_{ho}(s) + \frac{1}{1-(\alpha \Delta w_{ho}(s))} \tag{4.1.22}$$

whereas the weight update between hidden layer and input layer can be represent as:

$$\Delta w_{ih}(s+1) = -\eta \sum_{i=1}^{N} \frac{\partial E}{\partial w_{ih}} + \alpha \Delta w_{ih}(s) + \frac{1}{1-(\alpha \Delta w_{ho}(s))} \tag{4.1.23}$$

Where α is the momentum factor, usually $0 < \alpha \leq 1$ This method is usually called the BP with momentum (BPM) algorithm.

The BP algorithm is a supervised gradient-descent technique, wherein the MSE between the actual output of the network and the desired output is minimized. It is prone to local minima in the cost function. The performance can be improved and the occurrence of local minima reduced by allowing extra hidden units, lowering the gain term, and with modified training with different initial random weights. These are efficient variant of backpropagation learning algorithm, known as Quasi-Newton method, which is used to improve the performance of feed forward multilayer network architecture for the given training set.

4.2 Quasi-newton Methods

Quasi-Newton method is very often implemented with BFGS method. The BFGS method for multilayer feed forward Neural Network training. Quasi-Newton methods approximate Newton's direction without evaluating second order derivatives of the cost function. The approximation of the Hessian or its inverse is computed in an iterative process. They are a class of gradient-based methods whose descent direction vector d(t) approximates the Newton's direction[48].

$$d(t) = -H^{-1}(t)g(t) \tag{4.2.10}$$

Thus, one can obtain $d(t)$ with:

$$H(t)d(t) = -g(t) \tag{4.2.11}$$

The Hessian is always symmetric and is often positive-definite. Quasi - Newton methods with positive-definite. Hessian is called variable-metric methods. Secant methods are a class of variable-metric methods that use differences to obtain an approximation to the Hessian. These methods approximate the classical Newton's method, thus the convergence is very fast.

There are two globally convergent strategies available, namely, the line-search and trust-region methods. The line-search method tries to limit the step size along the Newton's direction until it is unacceptably large, whereas in the trust-region method the quadratic approximation of the cost function can be trusted only within a small region in the vicinity of the current point.

In quasi-Newton methods, a line search is applied such that

$$\lambda(t) = \arg\min_{\lambda \geq 0} E(\vec{w}(t) + \lambda d(t)) \tag{4.2.12}$$

and $\vec{w}(t+1) = \vec{w}(t) + \lambda(t)d(t)$ (4.2.13)

Line search is used to guarantee at each iteration the objective function decay that is dictated by the convergence requirement. The optimal $\lambda(t)$ can be theoretically derived from

$$\frac{\partial}{\partial \lambda} E(\vec{w} + \lambda d(t) = 0 \tag{4.2.14}$$

And this yields a representation using the Hessian. The second-order derivatives are approximated by the difference of the first-order derivatives at two neighboring points, and thus λ is calculated by

$$\lambda(t) = \frac{-\tau g(t)^T d(t)}{d(t)^T \left[g_\tau(t) - g(t)\right] d(t)} \tag{4.2.15}$$

where $g_r(t) = \nabla_{\vec{w}} E(\vec{w}(t) + \tau d(t)),$

and the size of neighborhood τ carefully selected. Some inexact line-search and line-search-free optimization methods are applied to quasi-Newton methods, which are further used for training FNNs [45].

There are many secant methods of rank one or two. The Broyden family is a family of rank-two and rank-one methods generated by taking [46].

$$H(t) = (1-\wp)H_{DFP}(t) + \wp H_{BFGS}(t) \tag{4.2.16}$$

where H_{DFP} and H_{BFGS} are, respectively, the Hessian obtained by the Davidon-Fletcher-Powell(DFP) and BFGS methods, and $\wp$ a positive constant between 0 and 1. By giving different values for $\wp$, one can get the DFP ($\wp = 0$), the BFGS ($\wp = 1$), or other rank-one or rank-two formulae. The DFP and BFGS methods are two dual rank-two secant methods, and the BFGS emerges as a leading variable-metric contender in theory and practice [47].

The BFGS method [44, 46, 47] is implemented as follows. Inexact line search can be applied to the BFGS, and this significantly reduces the number of evaluations of the error function. The Hessian H or its inverse is updated by

$$H(t+1) = H(t) - \frac{H(t)s(t)s^T(t)H(t)}{s^T(t)H(t)s(t)} + \frac{z(t)z^T(t)}{s^T(t)z(t)} \qquad (4.2.17)$$

$$H^{-1}(t+1) = H^{-1}(t) + (1 + \frac{z^T(t)H^{-1}(t)z(t)}{s^T(t)z(t)})\frac{s(t)s^T(t)}{s^T(t)z(t)} - (\frac{s(t)z^T(t)H^{-1}(t) + H^{-1}(t)z(t)s^T(t)}{s^T(t)z(t)}) \qquad (4.2.18)$$

Where $z(t) = g(t+1) - g(t)$ (4.2.19)

$$s(t) = \vec{w}(t+1) - \vec{w}(t) \qquad (4.2.20)$$

For its implementation, $\vec{w}$ (0), g(0), and $H^{-1}(0)$ are needed to be specified. H^{-1} is typically selected as the identity matrix.

5. Simulation Design and Implementation

In this simulation framework we have consider two different multilayer feed forward neural network architectures, first consist with 16 neurons in the input layer, 5 neurons in the hidden layer and 4 neurons in the output layer while the second architecture consist with 16 neurons in the input layer, 10 neurons in the hidden layer and 4 neurons in the output layer. The number of neurons in input layer and output layer are selected on the basis of dimensionality of input output patterns pair. In both the architecture the single hidden layer is used with different number of neurons. Both networks are trained with improved variant of back propagation learning rule namely Gradient descent with momentum & adaptive back propagation (TRAINGDX) and BFGS Quasi-Newton back propagation (TRAINBFG). These two learning methods are considered from the MATLAB simulation tool as neural network. The first learning rule which we have used to trained neural network architectures for the given training set of input pattern sample for hand written Hindi characters of SWARS is Gradient descent with momentum & adaptive back propagation (TRAINGDX), while the second algorithm which we have used for the same training set is BFGS Quasi-Newton back propagation (TRAINBFG). The implementation description of these two learning methods is defined in following subsection.

5.1 Gradient Descent With Momentum & Adaptive Backpropagation (TRAINGDX)

In MATLAB this learning rule is implemented with the function name TRAINGDX. TRAINGDX is a network training function that updates weight and bias values according to gradient descent momentum and an adaptive learning rate. The TRAINGDX function considers the following inputs as shown in table 2.

Table 2. Input information for function TRAINGDX

NET	Neural network.
Pd	Delayed input vectors.
Tl	Layer target vectors.
Ai	Initial input delay conditions.
Q	Batch size.
TS	Time steps.
VV	Empty matrix [] or structure of validation vectors.
TV	Empty matrix [] or structure of test vectors.

This function returns the following output values as shown in table 3.

Table 3. Output values from the function TRAINGDX

NET	Trained network.
TR	Training record of various values over each epoch:
TR.epoch	Epoch number.
TR.perf	Training performance.
TR.vperf	Validation performance.
TR.tperf	Test performance.
TR.lr	Adaptive learning rate.
Ac	Collective layer outputs for last epoch.
El	Layer errors for last epoch.

Parameters with their values are used to train the neural network architecture with TRAINGDX learning rules are shown in table 4.

Table 4. Parameters used in TRAINGDX during training

Epochs	1000	Maximum number of epochs to train
Goal	0	Performance goal
Lr	0.01	Learning rate
lr_inc	1.05	Ratio to increase learning rate
lr_dec	0.7	Ratio to decrease learning rate
max_fail	5	Maximum validation failures
max_perf_inc	1.04	Maximum performance increase
Mc	0.9	Momentum constant.
min_grad	1e-10	Minimum performance gradient
Show	25	Epochs between displays (NaN for no displays)
Time	Inf	Maximum time to train in seconds

TRAINGDX can train any network as long as its weights, net input and transfer function have derivative functions. In this learning rule back propagation is used to calculate derivative of performance with respect to weights and bias value, each variable is adjusted according to the gradient descent with momentum as described in equation (4.1.22) & (4.1.23). In this rule the learning rate is of adaptive nature. For each epoch if performance decreases towards the goal then learning rate is increases by factor lr_inc. If performance increases by more than factor max_perf_inc the learning rate is adjusted by the factor lr_dec and the change, which increased the performance, is not made. The training in this learning function can be stopped due to anyone of following condition:

1) The maximum number of EPOCHS (repetitions) is reached.
2) The maximum amount of TIME has been exceeded.
3) Performance has been minimized to the GOAL.
4) The performance gradient falls below MINGRAD.
5) Validation performance has increase more than MAX_FAIL times since the last time it decreased (when using validation).

5.2 BFGS Quasi-newton Backpropagation (TRAINBFG)

In MATLAB this learning rule is implemented with the function name TRAINBFG. TRAINBFG is a network training function that updates weight and bias values according to BFGS quasi-Newton method. The TRAINBFG function considers the following inputs as shown in table 5.

Table 5. Input information for function TRAINBFG

NET	Neural network.
Pd	Delayed input vectors.
Tl	Layer target vectors.
Ai	Initial input delay conditions.
Q	Batch size.
TS	Time steps.
VV	Either empty matrix [] or structure of validation vectors.
TV	Either empty matrix [] or structure of test vectors.

This function returns the following output values as shown in table 6.

Table 6. Output values from the function TRAINBFG

NET	Trained network.
TR	Training record of various values over each epoch:
TR.epoch	Epoch number.
TR.perf	Training performance.
TR.vperf	Validation performance.
TR.tperf	Test performance.
Ac	Collective layer outputs for last epoch.
El	Layer errors for last epoch.

These following parameters with their values are used to train the neural network architecture with TRAINBFG learning rule are shown in table 7.

Table 7: Parameter used in TRAINBFG during training

Epochs	1000	Maximum number of epochs to train
Show	25	Epochs between displays (NaN for no displays)
Goal	0	Performance goal
Time	Inf	Maximum time to train in seconds
min_grad	1e-6	Minimum performance gradient
max_fail	5	Maximum validation failures
SearchFcn	'srchcha'	Name of line search routine to use

TRAINBFG can train any network as long as its weights, net input and transfer function have derivative functions. In this learning rule back propagation is used to calculate derivative of performance with respect to weights and bias value, each variable is adjusted according to the Newton quasi learning equation as described in equation (4.2.18). The training in this learning function can be stopped due to any one of the following condition.

1) The maximum number of EPOCHS (repetitions) is reached.
2) The maximum amount of TIME has been exceeded.
3) Performance has been minimized to the GOAL.
4) The performance gradient falls below MINGRAD.
5) Validation performance has increased more than MAX_FAIL times since the last time it decreased (when using validation).

6. Result and Discussion

The result showed as in table 8 & 9 for the recognition of Hindi hand written character SWARS using TRAINGDX & TRAINBFG exhibiting non convergence for both networks. The table 8 is exhibiting the result for network one i.e. 16_5_4. It can see from the result that the network is not converging for any character. In this non convergence case the behavior of error for BFGS quasi-Newton backpropagation learning algorithm is more consistent. In the result it has been observed for both the training algorithm the error for the character , is minimum and for the character _ is near to maximum. It has also been observed that the error for vkS is increasing for both learning algorithms. The results are also indicating that the BFGS quasi-Newton backpropagation learning algorithm (TRAINBFG) is involving less error than Gradient descent with momentum & adaptive back propagation learning algorithm (TRAINGDX). In our experiments both the algorithm have trained for equal number epochs and it can be seen that BFGS quasi-Newton backpropagation learning algorithm(TRAINBFG) is seems to be more effective than Gradient descent with momentum & adaptive back propagation learning algorithm (TRAINGDX). The result of second network i.e. 16_10_4 is also indicating the same performance for both the algorithms. The important point which can be observed in the second network is that the tendency for minimum error common for both the algorithms. Both the algorithms are minimizing the error for same character and increasing the error for same character. The another point which can be observed in BFGS quasi-Newton backpropagation learning algorithm is that no error is in negative where as in Gradient descent with momentum & adaptive back propagation learning algorithm for some characters the error is negative. This tendency has found in both the networks. The another issue which can be observed is that for the TRAINBFG is that for some of the character the error is more than 1 but it is not happening with TRAINGDX. The reason of negative error in TRAINGDX for some characters is due to poor approximation for the sample points. The dominating of local errors for some of the characters are also playing the dominate rule for the non convergence of given training set. The analyses of the performance for both the algorithms can also be seen in the Graphical representation as in figure 4 and 5.

Table 8.Results for the training of handwritten Hindi characters of SWARS for Network 16-5-4 with both training functions

SAMPLES	NETWORK1	
	TRAINGDX	TRAINBFG
V	-0.23535	0.555367
Vk	-0.23545	0.529788
B	0.014364	0.782192
bZ	-0.22206	0.500621
M	0.014238	0.81352
Å	0.01423	0.822936
_	0.256107	1.02635
,	-0.21767	0.441452
,s	0.030275	0.804958
Vks	0.01521	0.868561
vkS	0.267964	1.131408
Va	0.017304	0.865007
v%	0.270064	1.034424

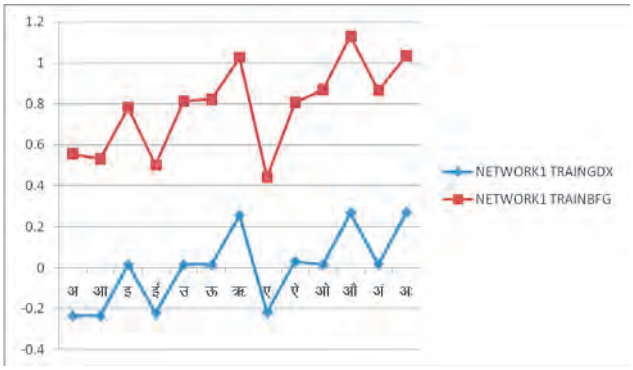


Figure 4.The performance analysis for the training set from network 16-5-4 with both training functions

Table 5. Results for the training of handwritten Hindi characters of SWARS for Network 16-10-4 with both training functions

SAMPLES	NETWORK 2	
	TRAINGDX	TRAINBFG
ट	-0.19502	0.088245
टा	-0.20109	0.118406
ठ	0.065327	0.355528
इ	-0.06217	0.09315
ड	0.06164	0.355768
ऊ	-0.00367	0.316997
ऋ	0.232999	0.528438
ए	-0.1439	0.085387
ऐ	-0.00926	0.323848
ओ	-0.02932	0.323767
औ	0.222451	0.563787
अं	0.010792	0.328683
अः	0.19633	0.585789

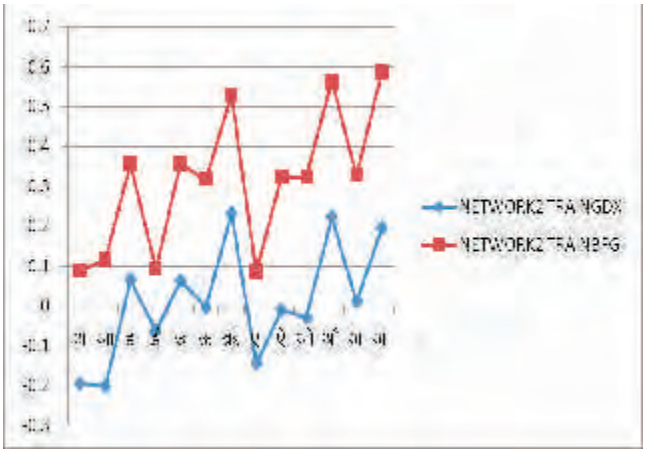


Figure 5.The performance analysis for the training set from network 16-10-4 with both training functions

6. Conclusion

The result of the experiments clearly shows that both networks are exhibiting the non convergence for the given training set of handwritten Hindi character of SWARS with both the training function. It means for any number of iteration the network will not converge at all for these two enhance variant of backpropagation learning algorithms for the given training set of Hindi characters of SWARS. It is expected if the number of neurons in hidden layer are extremely large than the error may be more reduce in BFGS quasi-Newton backpropagation learning algorithm but the non convergence definitely exist. The observation from the results for the performance of these two learning algorithms for the complex training set like handwritten character set of SWARS is reflecting the limitation of gradient descent learning algorithm for convergence due to the problem of local minima which is inherit problem of backpropagation learning algorithm. It shows that the local minima problem is the inherit feature of all the variant of gradient descent learning method or any variant of this learning rule. It means that the Gradient descent methods are not likely to be perfect searching algorithms for global optimization. It explores the possibility of incorporation for the evolutionary search for such type of complex pattern classification problem.

7. References

1. *Source: http://www.acm.org/ubiquity/, Ubiquity, vol.5 (7), (2004).*
2. *S. Watanabe, "Pattern Recognition: Human and Mechanical", New York: Wiley, (1985).*
3. *C. M. Bishop, "Pattern Recognition and Machine Learning", Springer, ISBN 0-387-31073-8, (2006)*
4. *R. G. Casey, and D. R. Ferguson, "Intelligent Forms Processing" , ZBM Syst. J. 29 (1990) 435-450.*
5. *P. Christenson, A. Maurer and G. Miner, "Handwriting recognition by neural network", (2005). (http://csci.mrs.umn.edu/UMMCSciWiki/pub/CSci4555s04/InsertTeamNameHere/handwriting.pdf.).*
6. *M. Friedman, and A. Kandel, "Introduction to Pattern Recognition: Statistical, Structural, Neural, and Fuzzy Logic Approaches", World Scientific, Singapore, (1999).*
7. *R. O. Duda, P. E. Hart and D. G. Stork, "Pattern classification (2nd edition)", Wiley, New York, (2001) ISBN 0-471-05669-3.*
8. *F. V. Jensen, "Bayesian Networks and Decision Graphs", Springer. (2001).*
9. *T. Cover and P. Hart,"Nearest Neighbor Pattern Classification", IEEE Trans. on Information Theory, 13(1) (1967) 21-27.*
10. *M. P. H. Wu., "Hand Written Character Recognition", The school on Information Technology and Electrical Engineering, University of Queensland, (2003).*
11. *B. Muller and J. Reinhardt, "Neural Networks: An Introduction.", Physics of Neural Network, New York: Springer-Verlag, (1991).*
12. *A. G. Barto, R. S. Sutton and C. Anderson, "Neuron-like adaptive elements that can solve difficult learning control problems", IEEE Transaction on Systems, Man and Cybernetics, 13 (1983) 834-846.*
13. *J. Dayhoff, "Neural-Network Architectures: An Introduction" New York: Van Nostrand Reinhold, (1990).*
14. *S. Grossberg, "Some networks that can learn, remember and reproduce any number of complicated space – time patterns", J. Math. Mech., 1(19) (1991) 53 – 91.*
15. *R. S. Sutton, A. G. Barto and R. J. Williams, "Reinforcement learning is direct adaptive optimal control", IEEE Control Systems Magazine, 12 (1999) 19 – 22.*
16. *E. D. Sontag, "Feed-forward nets for interpolation and classification", J. Computing System Sciences, 45 (1992) 20-48.*
17. *D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation", Parallel Distributed Processing: Explorations in the Microstructure of Cognition, 1 (1986) 318 – 362.*
18. *R.M.K. Sinha and V. Bansal, "On Automating Trainer for Construction of Prototypes for Devnagari Text Recognition", Technical Report TRCS-95-232, I.I.T. Kanpur, India.*
19. *R.M.K. Sinha and V. Bansal, "On Devnagari Document Processing", Int. Conf. on Systems, Man and Cybernetics, Vancouver, Canada, 1995.*
20. *R. M. K. Sinha and V. Bansal, "On Integrating Diverse Knowledge Sources in Optical Reading of Devnagari Script".*

21. *R. M. K. Sinha., "Rule Based Contextual Post-processing for Devnagari Text Recognition", Pattern Recognition, 20 (5) (1987) 475-485.*
22. *R. M. K. Sinha, "On Partitioning a Dictionary for Visual Text Recognition", Pattern Recognition, 23 (5) (1989) 497-500.*
23. *V. Bansal and R. M. K. Sinha, "On Automating Generation of Description and Recognition of Devnagari Script using Strokes", Technical Report TRCS-96-241, I.I.T. Kanpur, India.*
24. *R. M. K. Sinha, "A Journey from Indian Scripts Processing to Indian Language Processing", IEEE Annals of the History of Computing, (2009) 8-31.*
25. *R.G. Casey, D. R. Furgson, "Intelligent Forms Processing", IBM System Journal, 29 (3) 1990.*
26. *A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical Pattern Recognition: A Review", IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (1) (2000) 4-37.*
27. *G. Negi, "Twenty years of Document analysis in PAMI", IEEE Transactions on Pattern Analysis and Machine Intelligence, 20 (1) (2000) 38-62.*
28. *U. Pal, B. B. Chaudhuri, ''Indian Script Character recognition: A survey'', Pattern Recognition, 37 (2004) 1887-1899.*
29. *R. Kasturi, L. O'Gorman, V. Govindaraju, "Document Image Analysis: A Primer", Sadhana, 27 (1) (2002) 3-22.*
30. *R. Ishida, "An Introduction to Indic Scripts", http://www.w3.org/2002/Talks/09-ri-indic/indic- paper.pdf.*
31. *R. Sarkar et al, "A Script Independent Technique for Extraction of Characters from Handwritten Word Images", International Journal of Computer Applications 1 (23), (2010).*
32. *S. Arora et al., "Performance Comparison of SVM and ANN for Handwritten Devnagari Character Recognition", IJCSI International Journal of Computer Science Issues, 7 (3), (2010).*
33. *U. Bhattacharya, S. Vajda, A. Mallick, B. B. Chaudhuri and A. Belaid, "On the Choice of Training Set, Architecture and Combination Rule of Multiple MLP Classifiers for Multi resolution Recognition of Handwritten Characters", 9th Int'l Workshop on Frontiers in Handwriting Recognition (IWFHR-9) (2004).*
34. *M. E. Petersen, D. Ridder, H. Handels, "Image Processing with Neural Networks: A Review", Pattern Recognition, 35 (2002) 2279–2301.*
35. *P M Patil, T R Sontakke," Rotation, scale and translation invariant handwritten Devanagari numeral character recognition using general fuzzy neural network", Pattern Recognition, Elsevier, (2007).*
36. *K. Y. Rajput and S. Mishra, "Recognition and Editing of Devnagari Handwriting Using Neural Network", SPIT-IEEE Colloquium and Intl. Conference, Mumbai, India.*
37. *T. K. Bhowmik, S. K. Parui and U. Roy, "Discriminative HMM Training with GA for Handwritten Word Recognition", IEEE, (2008).*
38. *S. B. Cho, "Fusion of neural networks with fuzzy logic and genetic algorithm", IOS Press, (2002) 363–372.*
39. *B. Widrow, and M. E. Hoff, "Adaptive switching circuits.", IRE Eastern Electronic Show & Convention (WESCON1960), Convention Record, 4 (1960) 96–104.*
40. *P. J. Werbos, "Beyond regressions: New tools for prediction and analysis in the behavioral sciences", PhD Thesis, Harvard University, Cambridge, MA, (1974).*
41. *F. J. Pineda, "Generalization of back-propagation to recurrent neural networks.", Physical Rev Letter, 59 (1987) 2229–2232.*
42. *M. L. Minsky and S. Papert, "Perceptrons", MIT Press, Cambridge, MA, (1969).*
43. *W. Finnoff, "Diffusion approximations for the constant learning rate backpropagation algorithm and resistance to local minima", Neural Comp., 6 (2) (1994) 285–295.*
44. *R. Battiti, F. Masulli, "BFGS optimization for faster automated supervised learning", In: Proc. Int. Neural Network Conf. France, (2) (1990) 757-760.*
45. *H. S. M. Beigi, "Neural network learning through optimally conditioned quardratically convergent methods requiring no line search", In: Proc IEEE 36th Midwest Symp Circuits Syst. Detroit, MI, 1 (1993) 109-112.*
46. *R. Fletcher, "Practical methods of optimization", Wiley, New York (1991).*
47. *J. L. Nazareth, "Differentiable optimization and equation solving", Springer, New York (2003).*
48. *K. L. Du and M. N. S. Swami, "Neural networks in a soft computing framework", Springer-Verlag London Limited, e-ISBN 1-84628-303-5, (2006)*
49. *S. Shrivastava and M. P. Singh, "Performance evaluation of feed-forward neural network with soft computing techniques for hand written English alphabets", Journal of Applied Soft Computing, 11 (2011) 1156-1182*