RESEARCH ARTICLE OPEN ACCESS

Performance Comparison on Face Recognition System Using Different Variants of Back-Propagation Algorithm with Radial Basis Function Neural Networks

Kiran Arya and Virendra P. Vishwakarma
Department of Computer Science and Engineering, Inderprastha Engineering College (Affiliated to Mahamaya Technical University, UP, India), Ghaziabad 201010, UP, India

ABSTRACT
This paper presents the performance comparison of two architectures of neural networks: multi-layer perceptron (MLP) neural networks and radial basis function (RBF) neural networks on face recognition system (FRS). We are training MLP using different variants of back-propagation (BP) algorithm. AT&T database has been used for performance comparison. The BP is gradient descent based iterative algorithm which takes larger training time for high dimensional pattern recognition problem. Local minimum, improper learning rate and over-fitting are some of the other issues. To overcome these issues, we used RBF based FRS that is robust than other conventional methods like BP algorithm and has better performance of recognition rate. The training results show that in all the situations, RBF provides better generalization performance in compared of BP

General Terms: Artificial Neural Network, BP, Radial Basis Function
Keywords: Face recognition, Performance Comparison

I. INTRODUCTION

It seems to be easy for human being to recognize the faces but for the machine based system it is still not easy to achieve completely reliable and robust performance of FRS. These problems occurred due to different visual variations of images like illumination, head size, orientation, pose, expression, age, facial hair, glasses, background, and change in environment conditions etc. MLP is widely used to solve complex problems in pattern classification. One of the characteristics of the MLP is its learning (or training) ability. By training, the neural network can give correct answers not only for learned examples, but also for the models similar to the learned examples, showing its strong associative ability and rational ability which are suitable for solving large, nonlinear, and complex classification and function approximation problems [1].

Gradient based methods are one of the most widely used error minimization methods used to train BP networks. The back-propagation (BP) training algorithm is a supervised learning method for multi-layered feed-forward neural networks [2].

R.A. Finan, A.T. Sapelu, and R.I. Damper [3] trained the MLP using BP and RBF for text-dependent speaker recognition and compared the training results. They had shown that an RBF system can provide even better results with a suitable training set. For a good training set, a significant improvement would be expected for an RBF network relative to an MLP, whereas a poor training set will not show much improvement.

Haddadnia and Ahmadi [4] used a hybrid N feature (RBF) neural network method for face recognition system, which extracts a set of different kind of features from face images with RBF networks. These are combined together for classification purpose through the majority rule. They have used three different feature domains for features extraction from input images.

Vu N.P. Dao and Rao Vemuri [5] used five different methods of BP which are the gradient descent BP, the gradient descent BP with momentum, the variable learning rate Gradient descent BP, the conjugate gradient BP, and the quasi-Newton method, for training the neural network. They applied these methods to define the problem of computer network intrusion detection as a classification problem and showed that the performance of the BP methods depends on the neural networks topology but it was not as reliable as RBF.

N. M. Nawi, R. S. Ransing and M. R. Ransing [6] proposed a new computationally efficient algorithm CGFR/AG for training of MLP, in which the conjugate gradient optimization algorithm is combined with the modified BP algorithm. This algorithm is generic and easy to implement in all commonly used gradient based optimization processes. It is robust and has a potential to significantly enhance the computational efficiency of the training process.

In [7] it’s shown that the performance of training algorithm depends on many factors, including the complexity of the problem, the no of data points in the training set, the no. of weights and biases in the

network. So it is difficult to know which methods or algorithm performs better than the other.
Performance Comparison between the architectures of BP and RBF neural network shown in [8], which based on four classification and function approximation problem and comparative study of BP methods is shown in[9].

V. P. Vishwakarma and M. N. Gupta[11] proposed a new learning Algorithm SLFN_BVOI for the training of single hidden layer feed-forward neural network (SLFN) which perform effectively on high dimension and high variations problems of Face recognition. They showed that this algorithm learns on an average 734.2 times faster than BP and 9.2 times faster than ELM for different sizes of training set on AT&T face database.

In this paper, MLP and RBF networks are studied and compared based on their generalization capability and learning speed. All methods applied to the FRS system and used AT&T face database for recognition purpose.

This paper is organized as follows. The description of RBF along with BP is presented in Section 2. Section 3 describes the details of database used for our purpose. In Section 4, the experimental results and discussions on the data set are presented. Finally, conclusions are drawn in Section 5.

II. Radial Basis Function Network and Back Propagation Algorithm

2-1 BP Algorithm:

The BP algorithm is a technique used in training MLP in a supervised manner. BP also known as the error BP algorithm is based on the error-correction learning rule [14]-[16]. BP algorithm is a stochastic algorithm based on the steepest decent principle, wherein the weights of the neural network are updated along the negative gradient direction in the weight space. The simplest implementation of BP learning updates the network weights and biases in the direction in which the performance function decreases most rapidly i.e. the negative of the gradient [15]. There are a number of variations of the BP algorithm that are based on other standard optimization techniques. This chapter explains how to use each of these variations and discusses the advantages and disadvantages of each.

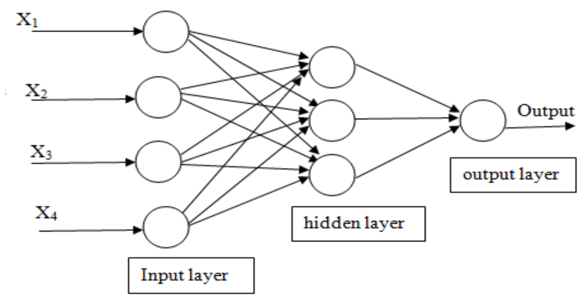


Fig. 1: Multi-layered neural network.

The MLP consists of three types of layers[8]. The first layer is the input layer and corresponds to the problem input variables with one node for each input variable. The second layer is the hidden layer used to capture non-linear relationships among variables. The third layer is the output layer used to provide predicted values.
Computations related with the single neuron include:
i) Input for hidden layer is given by:

$$net_j = \sum_{i=1}^{j} x_j w_{ji} \quad (1)$$

ii) The units of output vector of hidden layer after passing through the activation function are given by:

$$h_j = \frac{1}{1+\exp(-net_j)} \quad (2)$$

In same manner, input for output layer is given by

$$net_k = \sum_{i=1}^{k} h_m w_{ki} \quad (3)$$

and the units of output vector of output layer are given by

$$o_k = \frac{1}{1+\exp(-net_k)} \quad (4)$$

For updating the weights, we need to calculate the error. This can be done by

$$\xi(k) = \frac{1}{2}\sum_{i=1}^{k}(O_i - t_i)^2 \quad (5)$$

The input weights and biases for the next iteration are given as:

$$W(k+1) = w(k) - \eta\frac{\partial\xi(k)}{\partial W} \quad (6)$$

$$b(k+1) = b(k) - \eta\frac{\partial\xi(k)}{\partial b} \quad (7)$$

Similarly, the output weight and biases are updated as:

$$\beta(k+1) = \beta(k) - \eta\frac{\partial\xi(\boldsymbol{k})}{\partial\beta} \quad (8)$$

$$\alpha(k+1) = \alpha(k) - \eta\frac{\partial\xi(\boldsymbol{k})}{\partial\alpha} \quad (9)$$

Here η-is the learning rate. The performance of the algorithm is very sensitive to the proper setting of learning rate. If learning rate is set to high, the algorithm may oscillate and become unstable. If the learning rate is too small, the algorithm will take too long to converge. There are different variants of BP algorithm. With standard steepest descent BP, the learning rate is held constant throughout training. It is not practical to determine the optimal setting for the learning rate before training, and, in fact, the optimal learning rate changes during the training process, as the algorithm moves across the performance surface [8]-[10].

Some variants of BP:

There are two BP training algorithms: gradient descent and gradient descent with momentum are often too slow for practical problems. So we discuss several high performance algorithms that can converge from ten to one hundred times faster than these algorithms [15].

Gradient Descent BP (GD): This method updates the network weights and biases in the direction of the performance function that decreases most rapidly, i.e. the negative of the gradient. The new weight vector $\mathbf{w}_k+1$ is adjusted according to:

$$\mathbf{w}_{k+1}=\mathbf{w}_k - \alpha\, \mathbf{g}_k \quad (10)$$

The parameter α is the learning rate and **g**k is the gradient of the error with respect to the weight vector. The negative sign indicates that the new weight vector $\mathbf{w}_k+1$ is moving in a direction opposite to that of the gradient.

Gradient Descent BP with Momentum (GDM):

Momentum allows a network to respond not only to the local gradient, but also to recent trends in the error surface. Momentum allows the network to ignore small features in the error surface. Without momentum a network may get stuck in a shallow local minimum. With momentum a network can slide through such a minimum [14]-[16]. Momentum can be added to BP method learning by making weight changes equal to the sum of a fraction of the last weight change and the new change suggested by the gradient descent BP rule. The magnitude of the effect that the last weight change is allowed to have is mediated by a momentum constant, μ, which can be any number between 0 and 1. When the momentum constant is 0 a weight change is based solely on the gradient. When the momentum constant is 1 the new weight change is set to equal the last weight change and the gradient is simply ignored. The new weight vector **w**k+1 is adjusted as:

$$\mathbf{w}_{k+1}=\mathbf{w}_k - \alpha\, \mathbf{g}_{k+}\, \mu \mathbf{w}_{k-1} \quad \mathbf{(11)}$$

Gradient Descent with Adaptive learning rate (GDA): With standard steepest descent, the learning rate is held constant throughout training. The performance of the algorithm is very sensitive to the proper setting of the learning rate. If the learning rate is set too high, the algorithm may oscillate and become unstable. If the learning rate is too small, the algorithm will take too long to converge. It is not practical to determine the optimal setting for the learning rate before training, and, in fact, the optimal learning rate changes during the training process, as the algorithm moves across the performance surface.

The performance of the steepest descent algorithm can be improved if we allow the learning rate to change during the training process. An adaptive learning rate will attempt to keep the learning step size as large as possible while keeping learning stable. The learning rate is made responsive to the complexity of the local error surface.

An adaptive learning rate requires some changes in the training procedure used by traingd. First, the initial network output and error are calculated. At each epoch new weights and biases are calculated using the current learning rate. New outputs and errors are then calculated.

This procedure increases the learning rate, but only to the extent that the network can learn without large error increases. When a larger learning rate could result in stable learning, the learning rate is increased. When the learning rate is too high to guarantee a decrease in error, it gets decreased until stable learning resumes. BP training with an adaptive learning rate is implemented with the function traingda.

Variable Learning Rate BP with Momentum (GDX):

The learning rate parameter is used to determine how fast the BP method converges to the minimum solution. The larger the learning rate, the bigger the step and the faster the convergence. However, if the learning rate is made too large the algorithm will become unstable. On the other hand, if the learning rate is set to too small, the algorithm will take a long time to converge. To speed up the convergence time, the variable learning rate gradient descent BP utilizes larger learning rate α when the neural network model is far from the solution and smaller learning rate α when the neural net is near the solution. The new weight vector wk+1 is adjusted the same as in the gradient descent with momentum above but with a varying αk. Typically, the new weight vector wk+1 is defined as:

$$\mathbf{w}_{k+1}=\mathbf{w}_k - \alpha_{k+1}\, \mathbf{g}_k + \mu \mathbf{w}_{k-1} \quad \mathbf{(12)}$$

$$\alpha_{k+1}=\beta\, \alpha_k \quad \mathbf{(13)}$$

Resilient BP (Trainrp): The purpose of the resilient BP (Rprop) training algorithm is to eliminate these harmful effects of the magnitudes of the partial derivatives. Only the sign of the derivative is used to determine the direction of the weight update; the magnitude of the derivative has 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 delt_inc 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 delt_dec 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 the weight change will be reduced. If the weight continues to change in the same direction for several iterations, then the magnitude of the weight change will be increased.

2.2 Radial Basis Function Network (RBFN)

The back-propagation algorithm of a multi-layer feed-forward ANN is a gradient descent algorithm that may terminate at a local optimum, in addition to its long training time. We used of number of different methods for training MLP neural networks but no one gives faster speed of training neural network. This problem is overcome in RBF networks by incorporating the non-linearity in the activation functions of the nodes of the hidden layer [10].

An RBF neural network describe by Fu [11], The structure of RBF is similar to a traditional feed-forward neural network which is shown in the Fig. 2. The construction of the RBF neural network involves three different layers with feed-forward architecture. The input layer of this network is a set of n units, which accept the elements of an n-dimensional input feature vector. The input units are fully connected to the hidden layer with r hidden units. The goal of the hidden layer is to cluster the data and reduce its dimensionality. In this structure hidden layer is named RBF units. The RBF units are also fully connected to the output layer. The output layer supplies the response of neural network to the activation pattern applied to the input layer. The transformation from the input space to the RBF-unit space is nonlinear Gaussian function is used as a non-linear function, whereas the transformation from the RBF unit space to the output space is linear.

It should be noted that x is an n-dimensional input feature vector, c_i is an n-dimensional vector called the center of the RBF unit, σ_i is the width of RBF unit and r is the number of the RBF units. Typically the activation function of the RBF units is chosen as a Gaussian function with mean vector c_i and variance vector σ_i as follows:

$$\Phi_j(x)=\exp\frac{||x-ci||2}{\sigma i2}),1\leq j\leq n \qquad (14)$$

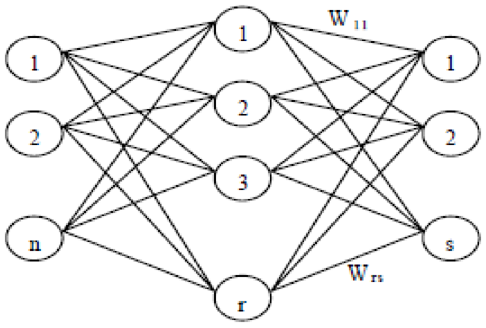


Fig. 2: RBF Neural Network

The output units are linear and therefore the response of the j-th output unit for input x is given as:

$$\mathbf{y_j(x)=b(j)+\sum_{i=1} r\Phi_j(x)w_2(i,j)} \qquad (15)$$

Where $w_2(i,j)$ is the weight of the link between the i^{th} hidden layer neuron and the j^{th} output layer neuron, b(j) is the bi j^{th} output later neuron.

Functions of RBF Neural Network: Radial basis function network can be designed with the functions newrbe and newrb.

Newrbe:

This function can produce a network with zero error on training vectors. It is called in the following way.

net = newrbe (P, T, SPREAD)

The function newrbe takes matrices of input vectors P and target vectors T, and a spread constant SPREAD for the radial basis layer, and returns a network with weights and biases such that the outputs are exactly T when the inputs are P.

The drawback to newrbe is that it produces a network with as many hidden neurons as there are input vectors. For this reason, newrbe does not return an acceptable solution when many input vectors are needed to properly define a network, as is typically the case.

Newrb:

The function newrb iteratively creates a radial basis network one neuron at a time. Neurons are added to the network until the sum-squared error falls beneath an error goal or a maximum number of neurons have been reached. The call for this function is:

net = newrb (P, T, GOAL, SPREAD)

The function newrb takes matrices of input and target vectors, P and T, and design parameters GOAL and, SPREAD, and returns the desired network. The design method of newrb is similar to that of newrbe. The difference is that newrb creates neurons one at a time. At each iteration the input vector that results in lowering the network error the most, is used to create a radbas neuron. The error of the new network is checked, and if low enough newrb is finished. Otherwise the next neuron is added. This procedure is repeated until the error goal is met, or the maximum number of neurons is reached.

Probabilistic Neural Networks

Probabilistic neural networks can be used for classification problems. When an input is presented, the first layer computes distances from the input vector to the training input vectors, and produces a vector whose elements indicate how close the input is to a training input. The second layer sums these contributions for each class of inputs to produce as its net output a vector of probabilities. Finally, a compete transfer function on the output of the second layer picks the maximum of these probabilities, and produces a 1 for that class and a 0 for the other classes.

2.3 Comparison between MLP and RBFN:-

In this work we present the properties of two types of neural networks: traditional neural networks and RBF networks, both of which are considered as universal approximators. The advantages and disadvantages of the two types of neural network architectures are analyzed and compared based on FRS [7].

- An RBFN has a single hidden layer, whereas an MLP may have one or more hidden layers.

- Typically the computation nodes of an MLP, located in a hidden layer or an output layer, share a common neuronal model. On the other hand, the computation nodes in the hidden layer of RBFN are quite different and serve a different purpose from those in the output layer of the network.
- Both are universal approximations. Thus, a RBF network exists for every MLP, and vice versa.
- The activation functions of the hidden nodes of an RBF network is based on the Euclidean norm of the input with respect to a center, while that of an MLP is based on the inner product of input and weights.
- MLPs construct global approximations to nonlinear input-output mapping. This is a consequence of the global activation function (sigmoid) used in MLPs
- RBF networks construct local approximations to input-output data. This is a consequence of the local Gaussian functions.

III. Database Used AT&T Face Database

The AT&T database [12] contains 400 gray-scale images of 40 persons. Each person has 10 different images of own, each having a resolution of 112×92, and 256 gray levels. Images of the individuals have been taken varying light intensity, facial expressions (open/closed eyes, smiling/not smiling) and facial details (glasses/no glasses). All the images were taken against a dark homogeneous background, with tilt and rotation up to 20^o sample of images shown in the Fig. 3.

Fig. 3: sample of images from AT&T database

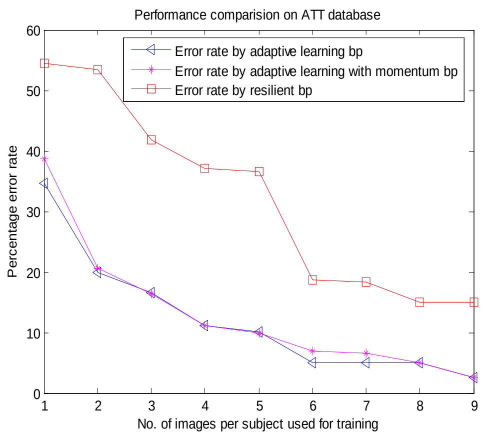
IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

For evaluating the proposed work, the experiments have been performed on AT&T [12] face database. To establish the improvement in generalization capability and learning speed of the neural network, these parameters of generalization performance have been compared with those of BP and RBF. The error rate variations for BP learning algorithm along with RBF have
been evaluated and compared on AT&T database to show the generalization capability of neural network. The training time has been measured for these methods to establish the improvement in the learning speed of the present algorithm. The experiments have been performed on a laptop pc with windows 8, 2.0 GHz core i3, Intel processor using MATLAB 7.0.1.
The experiments have been performed with different size of training set. The size of training set and test set is varying based on the number of images per subject, used for training. For example, if we take one image per subject for training, the training set size is 40 for AT & T face database. Similarly for two images per subject; the training set size is 80 and so on. The remaining images of the database are used for testing. The images are taken sequentially from database to build training set and test set, i.e. if number of images per subject for training is four, then the first four images per subject are used in training set and remaining six images are used for testing. The images of the database which have been used for training are not used for testing.
4.1 Error rate on AT&T face database: The percentage error rate variations on AT&T face database for different variants of BP algorithm as well as for RBF with respect to different training set size has been shown in Fig.4 and Fig. 5 respectively. Table 1 lists the percentage error rate for different variants of BP algorithm. The percentage error rate decreases as the number of images per subject for training, increases. The experimental results of three variants of BP show that Bp with adaptive learning rate (TRAINGDA) obtained better generalization capability rather than other methods like TRAINGDX and TRAINRP.
Table 2 lists the percentage error rate for different methods of RBF neural network. There are three training methods i.e. NEWRB, NEWRBE and NEWPNN used for trained the RBF neural network, in which NEWRBE function provides the significant reduction in percentage error rates in compare of other two.

Table 1. Comparison of percentage error rate on AT&T face database (percentage error rate using three different variants of BP i.e. TRAINGDA, TRAINRP, TRAINGDX.

%Error Rate / Training Algorithm	Number of images per subject used for training								
	1	2	3	4	5	6	7	8	9
%Error rate using adaptive learning BP	34.72	20	16.62	11.06	10.05	5	5	5	2.5
%Error rate using resilient BP	54.44	53.43	41.78	37.07	36.50	18.75	18.33	15	15

%Error rate using adaptive learning with momentum BP	38.61	20.62	16.42	11.07	10.0	6.87	6.6	5	2.5



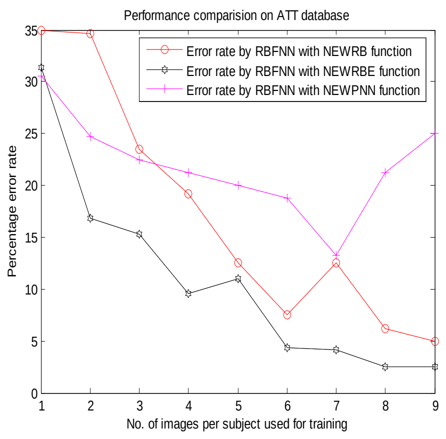


Fig. 4 Comparison of percentage error rate for variants of BP, learning algorithms

Fig. 5 Comparison of percentage error rate different for different variants of RBF, method on AT&T face database

AT&T face database.

Table 2. Comparison of percentage error rate on AT&T face database (percentage error rate using three different variants of RBF i.e. NEWRB, NEWRBE, NEWPNN).

%Error Rate / Training Algorithm	Number of images per subject used for training								
	1	2	3	4	5	6	7	8	9
%Error rate using rbfnn (newrb)	35	34.6	23.5	19.2	12.5	7.5	12.5	6.2	5
%Error rate using rbfnn (newrbe)	31.38	16.87	15.35	9.57	11.0	4.37	4.16	2.5	2.5
%Error rate using rbfnn (newpnn)	30.55	24.68	22.50	21.25	20.0	18.75	15.3	21.25	25.0

4.2 Learning speed on AT&T face database:

Fig. 6 and Fig. 7 show the learning speed of neural network using different variants of BP and RBF, which show that the RBF is much faster than the BP Algorithm in all the situations. Table 3 shows the comparison of training time obtained using three different variants of BP algorithm i.e. (TRAINGDA, TRAINGDX and TRAINRP). The training time decreases as the number of images per subject for training, increases. Experimental results show that the TRAINRP method is faster than the other two methods. Some other faster method of BP like trainlm (Levenberg- Marquardt BP) due to high dimensional training set (Out of memory error is generated when using trainlm) so it is not used for our purpose.

Table 4 shows the comparison of training time obtained using three different methods of RBF neural network i.e. (NEWRB, NEWRBE and NEWPNN). The training time decreases as the number of images per subject for training, increases. Experimental results show that the NEWRBE method is faster than the other two methods.

Table 3. Comparison of Training Time (in seconds) on AT&T face database (percentage error rate using different variants of BP training algorithm).

Training Time(Sec.) & Ratio / Training Algorithm	Number of images per subject used for training								
	1	2	3	4	5	6	7	8	9
Training Time using BP with TRAINGDA	75.25	123.57	115.35	210.85	173.96	251.29	254.65	343.50	316.70
Training Time using BP with TRAINRP	7.85	34.10	32.70	72.35	93.10	103.23	176.09	145.85	220.93
Training Time using BP with TRAINGDX	73.67	119.79	188.56	306.21	211.9	250.65	340.40	495.10	360.35

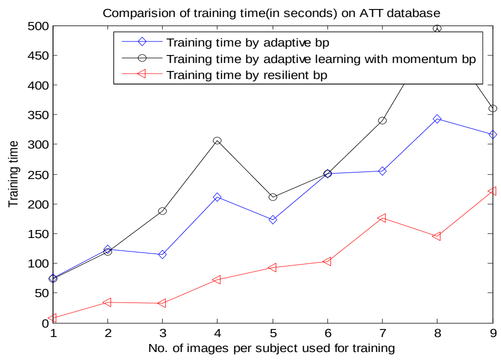


Fig. 6 Comparison of training time for different variants of BP, learning algorithms database AT&T

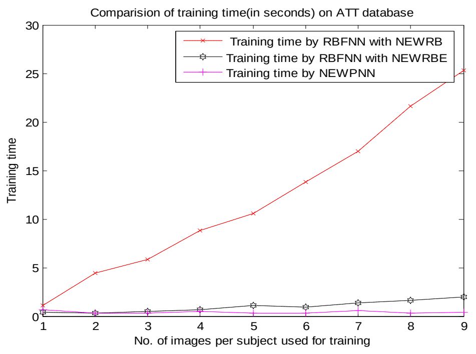


Figure 7 Comparison of training time for different variants of RBF, method on on AT&T face face database.

Table 4. Comparison of Training Time (in seconds) on AT&T face database using different variants of RBFNN training method.

Training Time(Sec.) & Ratio / Training Algorithm	Number of images per subject used for training								
	1	2	3	4	5	6	7	8	9
Training Time using rbfnn with NEWRB	1.14	4.42	5.8	8.8	10.6	13.8	17.0	21.6	25.3
Training Time using rbfnn with NEWRBE	0.43	0.32	0.46	0.64	1.14	0.95	1.32	1.59	1.96
Training Time using rbfnn with NEWPNN	0.64	0.29	0.34	0.50	0.32	0.29	0.53	0.35	0.36

4.3 Error rate with respect to different variations of Spread vales

RBF neural network is trained using NEWRBE function; training set used 4 images per subjects for training and remaining six used for the testing purpose. We used different spread values and find out the error rate with respect to these values and plotted the graph between these two parameters (i.e. spread values and error rate). In Fig. 8 it is shown that the error rate
is minimum at the point where the spread value is 25. So a spread value 25 is taken as
an optimal value and used for training purpose.

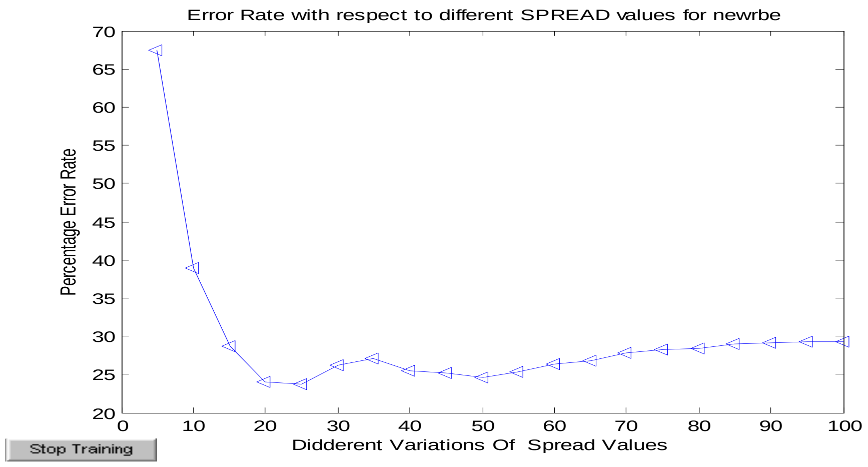


Fig. 8

4.4. Performance Comparison

The five BP methods that we used are the gradient descent (GD), the gradient descent with momentum (GDM), the variable learning rate (GDA), the variable learning rate gradient descent with momentum (GDX) and the resilient BP (RPROP). The GD algorithm is generally very slow because it requires small learning rates for stable learning. The GDM is usually faster than simple, since it allows higher learning rates while maintaining stability, but it is still too slow for many practical applications. These two methods would normally be used only when incremental training is desired. For the fast learning we can use GDX, but can only be used in batch mode training and RPROP which is simple batch mode training algorithm with fast convergence, minimal storage requirements and used for the large network.

RBF can be designed very quickly in two different ways.

The first design method, newrbe, finds an exact solution. The function newrbe creates RBF with as many radial basis neurons as there are input vectors in the training data.

The second method, newrb, finds the smallest network that can solve the problem within a given error goal. Typically, far fewer neurons are required by newrb than are returned newrbe. However, because the number of radial basis neurons is proportional to the size of the input space, and the complexity of the problem, RBF can still be larger than BP networks.

Probabilistic neural networks (PNN) can be used for classification problems. Their design is straightforward and does not depend on training. A PNN is guaranteed to converge to a Bayesian classifier providing it is given enough training data. These networks generalize well. PNN have many advantages, but it is suffer from one major disadvantage. It is slower to operate because it use more computation than other kinds of networks to do their function approximation or classification.

V. CONCLUSION

This paper presented a novel method for the recognition of human faces using multi layered feed-forward neural network which trained using different training methods of BP and RBF. In all the situations RBF gives better generalization performance in comparison of BP. RBF takes less training time to trained neural network and make system most robust than BP. It is very difficult to know which training algorithm will perform the best for a given problem, because it depends on many factors, including the complexity of the problem, the no of data points in the training set, the no. of weights and biases in the network. However, an RBF system can provide even better results with a suitable training set. For a good training set, a significant improvement would be expected for an RBF network relative to an MLP, whereas a poor training set will not show much improvement.

References

[1] Yanping Bai, Haixia Zhang and Yilong Hao," The performance of the BP algorithm with varying slope of the activation function", Chaos, Solitons and Fractals 40 (2009) 69–77.

[2] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, Learning internal representations by error propagation. in D.E. Rumelhart and J.L. McClelland (eds), Parallel Distributed Processing, 1986. 1: p. 318-362.

[3] R.A. Finan , A.T. Sapeluk and R.I. Damper, " Comparison of Multi-layer and Radial Basis Function Neural Networks for Text-Dependent Speaker Recognition"

[4] J. Haddadnia and M. Ahmadi, " N-feature neural network human face recognition," Image and Vision Computing, vol. 22, vol. 12, pp. 1071-1082, Oct. 2004.

[5] Vu N.P. Dao and Vemuri Rao", A Performance Comparison of Different BP Neural Networks Methods in Computer Network Intrusion Detection"

[6] N. M. Nawi, R. S. Ransing and M. R. Ransing," An Improved Conjugate Gradient Based Learning Algorithm for BP Neural Networks", World Academy of Science, Engineering and Technology 18 2008.

[7] Ozgur Kisi and Erdal uncuoglu,"comparison of three BP algorithms for two case studies", Indian journal of engineering and materials sciences, Vol.12 Oct 2005, pp.434-442.

[8] Tiantian Xie, Hao Yu and Bogdan Wilamowski "Comparison between Traditional Neural Networks and Radial Basis Function Networks" 978-1-4244-9312-8/11/$26.00 ©2011 IEEE

[9] Salim Lahmiri," A comparative study of back-propagation algorithm in financial prediction", International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.1, No.4, August 2011

[10] S. K. Jain, and V. P. Singh, "Applications of artificial neural networks to water resources." Water and Environment International Conference on 15-18 Dec. Bhopal, M.P., India, 2003.

[11] Fu, L. Neural networks in computer intelligence, McGraw-Hill, Inc., 1994, 460p.

[12] V. P. Vishwakarma and M. N. Gupta, "A New Learning Algorithm for Single Hidden Layer Feedforward Neural Networks", in IJCA,(0975-8887), Vol. 28-No. 6,August 2011.

[13] AT&T face database, [Online]. Available: http://www.cl.cam.ac.uk/Research/DTG/attarchive/pub/data/att_faces.tar.Z.

[14] S.Haykin, Neural Networks—A Comprehensive Foundation, 2nd Ed. Prentice Hall, 1999.

[15] M.T. Hagan, H.B. Demuth, and M. Beale, Neural Network Design, Thomson Learning, 2002

[16] J. Principe, N. Euliano, W. Lefebvre, Neural and Adaptive System – Fundamentals through Simulations, Wiley, 2000.