An Improved Multilayer Perceptron for BCI with Evolutionary Optimization

G. V. Sridhar
Associate Professor
Department of Electronics & Communication Engineering
Avanthi Institute Of Engg & Technology, Visakhapatnam

DR. P. Mallikarjuna Rao
Professor
Department of Electronics & Communication Engineering
AUCE(A),Visakhapatnam

Abstract

Brain Computer Interface (BCI) research is based on recording and analyzing electroencephalographic (EEG) data and recognizing EEG patterns associated with various mental states. BCIs had become an active research area in the last decade. The ability of The Multilayer Perceptron Neural Networks (MLPNN) to model non-linear relationship is appropriate to model the complex nature of EEG signals. MLPNN's convergence rate is relatively slow and it often yields sub-optimal solutions. In this paper, it is proposed to enhance the classification ability of the MLPNN incorporating Particle Swarm Optimization (PSO). This optimization helps to overcome the slow convergence rate and to avoid local minimum. The features of motor imagery in the frequency domain are extracted using Hilbert transform, and Principal Component Analysis is used for feature reduction. This is classified by MLPNN modified with PSO.

Keywords: Brain Computer Interface (BCI), EEG, Multilayer Perceptron (MLP), Particle Swarm Optimization (PSO)

I. INTRODUCTION

A Brain Computer Interface (BCI) allows a person to control special computer applications like a computer cursor or robotic limb through the use of his/her thoughts [1, 2]. BCI research is mainly based on recording and analyzing electroencephalographic (EEG) brain activity and recognizing EEG patterns associated with various mental states. Supervised classification methods are employed to recognize these EEG activity patterns to learn the mapping between the EEG data and classes corresponding to mental tasks like movement of the left hand [3].

From data mining point of view this was difficult because of two reasons. First, EEG data are noisy and correlated as many electrodes were required to be fixed on a small scalp surface with each electrode measuring the activity of thousands of neurons [4, 5]. Various types of methods have been used in EEG classification including conceptual and statistical models. The application of time series methods may be complicated by non-stationary and non-linearity in the data, requiring experience and expertise from the modeler. Due to the difficulties, Artificial Neural Networks (ANNs) approach was applied for EEG classification [6-8].

The ability of ANNs to model non-linear relationship is appropriate to model the complex nature of EEG signals. ANNs offer a relatively fast and flexible means of modeling and are data independent. ANNs do not impose functional relationship between independent and dependent variables. The functional relationship is determined by the data in the training process. The main advantage of ANNs is that a network with sufficient hidden units is able to approximate any continuous function to any degree of accuracy by performing efficient training [9, 10]. The Multilayer Perceptron Neural Networks (MLPNN) are generally trained by standard backpropagation algorithm (BPNN). Although BPNN proved to be efficient in some applications, its convergence rate is relatively slow and it often yields sub-optimal solutions [11, 12].

BPNN learning basically is a hill climbing technique. Therefore, there is a risk of being trapped in local minima where every small change in synaptic weight increases the cost function. Sometime, the network is stuck where there exist another set of synaptic weight for which the cost function is smaller than the local minimum in the weight space. This made the BPNN undesirable to have the learning process terminate at a local minimum.

To overcome this slow convergence rate problem, Particle Swarm Optimization (PSO) is used. The function of PSO in NN is to get the best set of weight (or particle position) where several particles (problem solution) are trying to move or fly to get best solution. PSO with MLPNN had been successfully applied in solving classification problems [13].

Thus, the goal of this study is to evaluate BCI classification task using MLPNN modified with PSO, based on a benchmark dataset. Hilbert Transform for feature extraction and Principal Component Analysis is used for feature reduction.

II. RELATED WORKS

Some of the works in the literature using neural networks are reviewed in this section. Khare et al [14] investigated the performance of three classifiers for classification of five mental tasks. Wavelet Packet Transform (WPT) was used for feature extraction of the relevant frequency bands from raw Electroencephalograph (EEG) signal. The three classifiers namely used were Multilayer Back propagation Neural Network, Support Vector Machine and Radial Basis Function Neural Network. In MLP-BP

NN five training methods used were a) Gradient De-scent Back Propagation b) Levenberg-Marquardt c) Resilient Back Propagation d) Conjugate Learning Gradient Back Propagation and e) Gradient Descent Back Propagation with momentum. The result showed the performance of neural network with resilient back propagation training method, support vector machine and radial bases function Neural Network for classifying of mental tasks w.r.t baseline. RBF (Radial Basis Function) neural network method has best performance among all the classifiers for classification of mental tasks w.r.t baseline.

Nakayama [6] proposed a BCI system, using orthogonalized EEG data sets and multiple multilayer neural networks (MLNNs) in a parallel form. In order to emphasize feature of multi-channel EEG data, Gram-Schmidt orthogonalization has been applied. Since there are many channel orders to be orthogonalized, many kinds of orthogonalized data sets can be generated for the same EEG data set by changing the channel order. These data sets have different features. In the proposed method, different channel orders are assigned to the multiple MLNNs in a training phase and in the classification process. A good solution can be searched for by changing the channel orders within a small number of trials. By using EEG data for five mental tasks, a correct classification rate is increased from 88% to 92%, and an error classification rate is decreased from 4% to 0%.

Nawroj et al [15] developed an event classifier to analyze the collected EEG signals and distinguish between different events. The classifier introduced in this study was based on an ANN model. The training process of the neural network required large amount of sample data and target results. A trained artificial neural network can then predict the outcome of an event based on the information of the corresponding EEG signal. The architecture of the artificial neural network involved hidden layers in addition to the input and output layers, which satisfied the non-linearity of the problem that the classifier was designed to solve. Experiments were conducted to validate this approach by using the classifier to distinguish whether subjects placed their fingers into hot or cold water. Validation results demonstrated the effectiveness of the classifier and its potential application in other fields.

Hazrati et al [16] presented a new online single-trial EEG-based brain–computer interface (BCI) for controlling hand holding and sequence of hand grasping and opening in an interactive virtual reality environment. An adaptive probabilistic neural network (APNN) was introduced for working in a time-varying environment for classification of EEG signals. The experimental evaluation on ten naïve subjects demonstrated that an average classification accuracy of 75.4% was obtained during the first experiment session (day) after about 3 min of online training without offline training, and 81.4% during the second session (day). The average rates during third and eighth sessions are 79.0% and 84.0%, respectively, using previously calculated classifier during the first sessions, without online training and without the need to calibrate. The results obtained from more than 5000 trials on ten subjects showed that the method could provide a robust performance over different experiment sessions and different subjects.

III. METHODOLOGY

A. Dataset

To investigate the proposed method publicly available dataset available in [17] was obtained. The IV A dataset used in the brain computer interface competition provided by Intelligent Data Analysis Group is used as dataset for experimentation [17]. The EEG recordings complied from five healthy subjects sitting in a chair with arms resting on armrests forms the dataset. Visual cues for 3.5 s were shown for the subject to perform 3 motor imageries: (L) left hand, (R) right hand, (F) right foot. The dataset contains continuous signals of 118 EEG channels and markers that indicate the time points of 280 cues for each of the 5 subjects (aa, al, av, aw, ay). Subject aa was used in our study.

B. Multilayer Perceptron (MLP)

Artificial neural networks (ANN) were seen as attempts to reproduce human brain potentialities, specially its learning ability. The values of input signals and related weights determine the neuron output. Hence, perceptron or artificial neuron is a mathematical model of a neuron cell and the basic unit compounding artificial neural network. Perceptron architecture includes a set of *n* inputs (*xi*), each related to a weight (*wi*) and an activation function (*fi*). Perceptrons are organized to form layers, where all are linked to same inputs but with distinct outputs.

Multilayer perceptron (MLP) networks have an input layer (*Xi*), one or more intermediary or hidden layers (*HL*) and an output layer (*Y*). A weight matrix (*W*) is defined for each layer. The ANN topology solves classification problems with non-linearly separable patterns and is also used as a universal function generator.

MLP have two clear phases: training and execution. A popular algorithm for MLP network training is backpropagation with variants. This learning approach is more complex than that for a perceptron network and it is of the supervised type [18].

The basic MLP learning algorithm is developed below [19].

1) Initialise network, with weights set to random numbers between -1 and +1.
2) Present first training pattern, and obtain output.
3) Compare network output with target output.
4) Propagate error backwards.

1) Correct Output Weights Layer with the Formula Below

$$w_{ho} = w_{ho} + \left(\eta \delta_o o_h\right)$$

where w_{ho} is the weight connecting hidden unit h with output unit o, η is the learning rate, o_h is the output at hidden unit h. δ_o is given as follows:

$$\delta_o = o_o\left(1-o_o\right)\left(t_o - o_o\right)$$

Where o_o is output at node o of output layer, and t_o is target output for that node.

2) Correct Input Weights Using Following Formula

$$w_{ih} = w_{ih} + \left(\eta \delta_h o_i\right)$$

where w_{ih} is weight connecting node i of input layer with node h of hidden layer, o_i is input at node i of input layer, δ_h is calculated as follows.

$$\delta_h = o_h\left(1-o_h\right)\sum_o\left(\delta_o w_{ho}\right)$$

5) Calculate error, by taking average difference between target and output vector. The following function can be used as an example.

$$E = \frac{\sqrt{\sum_{n=1}^{p}\left(t_o - o_o\right)^2}}{p}$$

Where p is number of units in output layer.

6) Repeat from 2 for each pattern in training set to complete one epoch.
7) Shuffle training set randomly, to prevent network being influenced by the data order.
8) Repeat from step 2 for a specific number of epochs, or till the error ceases to change.

The transfer function is selected by the designer after which parameters and they will be adjusted by a learning rule to ensure that the neuron input/output relationship meets a specific goal. Sigmoid and tanh transfer function are used in this study. The transfer function is obtained as follows:

- Sigmoid Function

$$P\left(t\right) = \frac{1}{1+e^{-t}}$$

- Tanh function

$$\tanh = \frac{e^{2x}-1}{e^{2x}+1}$$

C. Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO), a new branch of the soft computing paradigms [20] called evolutionary algorithms (EA), was developed by Kennedy and Eberhart (1995). It is a group-based stochastic optimization technique for continuous nonlinear functions. It is also a simple concept adapted from nature decentralized and self-organized systems where all the particles move to get better results. PSO is initialized with a group of random particles (solutions), which were assigned with random positions and velocities. The algorithm then searches for optima through a series of iterations where the particles are flown through the hyperspace searching for potential solutions. These particles learn over time in response to their own experience and other particles experience in their group [21].

Each particle keeps track of its best fitness position in hyperspace that has achieved so far [22]. This best position value is called personal best or "pbest". The overall best value
obtained by any particle so far in the population is called global best or "gbest". During iteration, every particle is accelerated towards its own "pbest" as well as in the direction of the "gbest" position. This is achieved by calculating a new velocity term for each particle based on the distance from its "pbest" as well as its distance from the "gbest" position. These two "pbest" and "gbest" velocities are then randomly weighted to produce the new velocity value for this particle, which will affect the next position of the particle in next iteration [22]. The advantage of the PSO over many of the other optimization algorithm is its relative simplicity [23].

In this study, PSO is applied to train MLPNN for enhancing the convergence rate. The learning process involves finding a set of weights that minimizes the learning error. The position of each particle in PSONN represents a set of weight for current iteration [24]. The dimension of each particle is the number of weights associated with the network. The learning error of this network is computed using Mean Squared Error (MSE). The particle will move within the weight space attempting to minimize learning error. The "pbest" value (each particle's lowest learning error so far) and "gbest" value (lowest learning error found in entire learning process so far) are applied to the velocity update equation to produce a value for positions adjustment to the best solution or targeted learning error. The new sets of positions (NN weight and bias) are produced by adding the calculated velocity value to the current position value using movement equation. Then, the new sets of positions are used to produce new learning error in MLPNN. This process is repeated until the stopping conditions either minimum learning error or maximum

number of iteration is met. The optimization output, which is the solution for the optimization problem, was based on gbest position value. The movement equation and the velocity update [25] are given as follows:

$$currentLocation = prevLocation + V_i \Delta t$$

$$where \;\; V_i = current\ velocity$$

$$\Delta t = discrete\ time - \text{int}\ erval$$

$$V_i = \omega V_{i-1} + c_1 * rand() * \left(pbest - currentLocation \right) + c_2 * rand() * \left(gbest - currentLocation \right)$$

where ω is the inertia weight, rand is the random numbers, c_1 and c_2 are the learning factors.

IV. RESULTS AND DISCUSSION

Labview was used to implement the Hilbert Transform for feature extraction. The maximum and minimum energy are computed for all the evoked responses. Principal Component Analysis is used for feature reduction. Classification is carried out using multilayer perceptron modified with PSO with different learning rate and Momentum. Experiments are conducted using tenfold cross validation using a MLP classifier. Table 1 gives the classifier model parameters. Experiments are conducted using sigmoid, tanh functions and the proposed PSO based MLPNN. The learning rate was 0.1 and momentum of 0.2 was maintained.

Table 1:
MLPNN Parameters

Random Number Seed	*0*
Learning Rate	*0.1*
Learning Rate Function	*Static learning rate*
Constant Bias Input	*1*
Training Iterations	*500*
Training Mode	*Batch Training - weight changes are applied at the end of each epoch*
Transfer Function	*Sigmoid, S-shape function between +1 and 0 ; Tanh, S-shape function between +1 and -1*
Momentum	*0.2*
Weight Decay	*0*
Bias Input Value	*1*
Num Inputs:	*49*
Hidden Layer 1	*15* *15, 20, 25 for PSO based MLPNN*
Output Layer	*2*

The classification accuracy obtained from the proposed method and MLP as the classifier is shown in Figure 2. Table I tabulates the classification accuracy. Table 2 gives the classification accuracy. It is seen that tanh transfer function performance is the lowest.

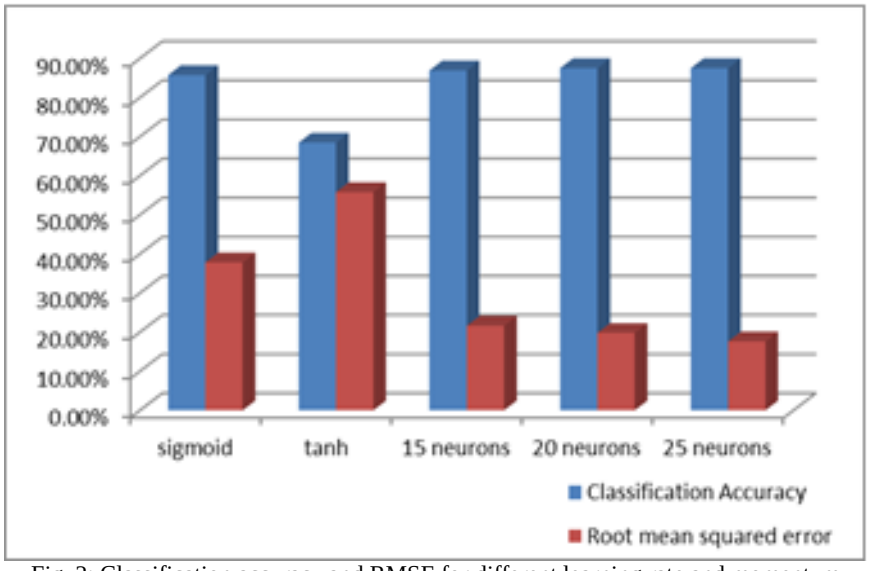

Fig. 2: Classification accuracy and RMSE for different learning rate and momentum

Table 2:
Classification Accuracy and Root Mean Squared Error

Technique	*Classification Accuracy*	*Root mean squared error*
sigmoid	*85.71%*	*0.378*
tanh	*68.45%*	*0.558*
15 neurons in PSO-MLPNN	*86.90%*	*0.216*
20 neurons in PSO-MLPNN	*87.50%*	*0.198*
25 neurons in PSO-MLPNN	*87.50%*	*0.176*

V. CONCLUSION

In this paper, the EEG signal is preprocessed to extract features by converting the time series EEG data to frequency domain using Hilbert Transform and PCA for feature reduction. It is proposed to enhance the classification ability of the MLPNN incorporating Particle Swarm Optimization (PSO). This optimization helps to overcome the slow convergence rate and to avoid local minimum. Experiments are conducted using tenfold cross validation and the features are classified using Multilayer Perceptron using sigmoid and tanh function and the proposed PSO based MLPNN. The accuracy obtained is comparable with the results obtained from other researchers in literature. The proposed method is extremely fast in both feature extraction and classification.

REFERENCES

[1] Wolpaw, J.R., Birbaumer, N, Heetderks, W.J., McFarland, D.J, Peckham, P.H., Schalk, G., Donchin, E., Quatrano, L.A, Robinson, C.J., Vaughan, T.M. (2000): Brain-computer interface technology: a review of the first international meeting. IEEE Transactions on Rehabilitation Engineering **8**:164-173.
[2] Dornhege, G., Millán, J. del R., Hinterberger, T., McFarland, D.J. and Müller K.-R. (2007): Toward BrainComputer Interfacing. Cambridge, MIT Press.
[3] Lotte, F., Congedo M., Lecuyer A., Lamarche F. And Arnalsi B. (2007): a review of classification algorithms for EEG-based brain-computer interfaces. Journal of Neural Engineering **4**: R1-R13.
[4] Lee, F., Scherer, R., Leeb, R., Neuper, C., Bischof, H. and Pfurtscheller, G. (2005): A comparative analysis of multi-class EEG classification for brain computer interface. Proc. 10th Computer Vision Winter Workshop (CVWW), Technical University of Graz, Austria.
[5] Thulasidas, M., Guan C. and Wu, J. (2006): Robust classification of EEG signal for brain-computer interface. IEEE Transactions on Neural Systems and Rehabilitation Engineering **14**:24-9.
[6] Nakayama, K., Horita, H., & Hirano, A. (2010). A BCI system based on orthogonalized EEG data and multiple multilayer neural networks in parallel form. Artificial Neural Networks–ICANN 2010, 205-210.
[7] Faradji, F., Ward, R. K., & Birch, G. E. (2009). Plausibility assessment of a 2-state self-paced mental task-based BCI using the no-control performance analysis. Journal of neuroscience methods, 180(2), 330.
[8] F. Lotte, M. Cengedo, A. Lecuyer, F. Lamarche, and B. Arnaldi. A review of classification algorithms for eeg-based brain computer interfaces. Journal of Neural Engineering, 4, 2007.
[9] Cybenko, G. (1989). "Approximation By Superposition of a Sigmoidal Function, Math.Control Signals Syst". 2, 303-314.
[10] Hornik, K., Stinchcombe, M., White, H. (1989). "Multilayer Feedforward Networks are Universal Approximators". Neural Networks 2(5), 359-366.
[11] Mulenbein, H. (1990). "Limitations of Multilayer Perceptron Networks-Steps Towards Genetic Neural Networks". Parallel Computing 14, 249-260.
[12] Sima, S. (1996). "Backpropagation is Not Efficient". Neural Networks 9(6), 1017-1023.
[13] Zhang, C., Shao, H. and Li, Y. (2000). "Particle Swarm Optimization for Evolving Artificial Neural Network". IEEE: 2487-2490.
[14] Khare, V., Santhosh, J., Anand, S., & Bhatia, M. (2010). Performance comparison of three artificial neural network methods for classification of electroencephalograph signals of five mental tasks. Journal of Biomedical Science and Engineering, 3(2), 200-205.
[15] Nawroj, A., Wang, S., Jouny, I., Yu, Y. C., & Gabel, L. (2012, March). An event classifier using EEG signals: An artificial neural network approach. In Bioengineering Conference (NEBEC), 2012 38th Annual Northeast (pp. 386-387).
[16] Hazrati, M. K., & Erfanian, A. (2010). An online EEG-based brain–computer interface for controlling hand grasp using an adaptive probabilistic neural network. Medical engineering & physics, 32(7), 730-739.
[17] Guido Dornhege, Benjamin Blankertz, Gabriel Curio, and Klaus-Robert Müller. Boosting bit rates in non-invasive EEG single-trial classifications by feature combination and multi-class paradigms. IEEE Trans. Biomed. Eng., 51(6):993-1002, June 2004.
[18] Victor Hugo Costa Albuquerque, Auzuir Ripardo de Alexandria, Paulo César Cortez, João Manuel R. S. Tavares," Evaluation of Multilayer Perceptron and Self-Organizing Map Neural Network Topologies applied on Microstructure Segmentation from Metallographic Images"
[19] Leonardo Noriega ,"Multilayer Perceptron Tutorial"
[20] J. Kennedy and R.C. Eberhart (1995). "Particle Swarm Optimization". In Proceedings of the IEEE International Joint Conference on Neural Networks, pages 1942–1948. IEEE Press.
[21] Ferguson, D. (2004). "Particle Swarm". University of Victoria, Canada.
[22] R. Eberhart, Y. Shi. (2001). "Particle swarm optimization: developments, applications and resources". IEEE Int. Conf. on Evolutionary Computation, 2001: 81-86.
[23] Van den Bergh, F. and Engelbrecht, A. P. (2000). "Cooperative Learning in Neural Networks using Particle Swarm Optimizers". South African Computer Journal. (26):84-90.
[24] Al-kazemi, B. and Mohan, C.K. (2002). "Training Feedforward Neural Network Using Multi-phase Particle Swarm Optimization". Proceeding of the 9th International Conference on Neural Information Processing. New York.
[25] Jones M.T (2005). "AI Application Programming". 2nd Ed. Hingham, Massachusetts.