International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:4, 2008

Automatic Removal of Ocular Artifacts using JADE Algorithm and Neural Network

V Krishnaveni, S Jayaraman, A Gunasekaran, K Ramadoss

***Abstract*—**The ElectroEncephaloGram (EEG) is useful for clinical diagnosis and biomedical research. EEG signals often contain strong ElectroOculoGram (EOG) artifacts produced by eye movements and eye blinks especially in EEG recorded from frontal channels. These artifacts obscure the underlying brain activity, making its visual or automated inspection difficult. The goal of ocular artifact removal is to remove ocular artifacts from the recorded EEG, leaving the underlying background signals due to brain activity. In recent times, Independent Component Analysis (ICA) algorithms have demonstrated superior potential in obtaining the least dependent source components. In this paper, the independent components are obtained by using the JADE algorithm (best separating algorithm) and are classified into either artifact component or neural component. Neural Network is used for the classification of the obtained independent components. Neural Network requires input features that exactly represent the true character of the input signals so that the neural network could classify the signals based on those key characters that differentiate between various signals. In this work, Auto Regressive (AR) coefficients are used as the input features for classification. Two neural network approaches are used to learn classification rules from EEG data. First, a Polynomial Neural Network (PNN) trained by GMDH (Group Method of Data Handling) algorithm is used and secondly, feed-forward neural network classifier trained by a standard back-propagation algorithm is used for classification and the results show that JADE-FNN performs better than JADE-PNN.

***Keywords*—**Auto Regressive (AR) Coefficients, Feed Forward Neural Network (FNN), Joint Approximation Diagonalisation of Eigen matrices (JADE) Algorithm, Polynomial Neural Network (PNN).

V Krishnaveni and S Jayaraman are with Department of Electronics & Communication Engineering, PSG College of Technology, Coimbatore – 641 004 India as Senior Lecturer and Professor & Head (corresponding author e-mail: venimurthy@hotmail.com, jayaramathreya@yahoo.com.

A Gunasekaran is with Cognizant Technology Solution, Chennai, India as Program Analyst Trainee.

K Ramadoss is with Department of Neurology, PSG Institute of Medical Sciences and Research, Coimbatore - 641 004 India as Professor & Head.

I. Introduction

INDEPENDENT Component Analysis (ICA) proves to be effective in removing the ocular artifacts from EEG recordings. However, while using ICA algorithms for ocular artifact correction, a crucial step is to correctly identify the artifact components among the decomposed independent components. The component based procedures used by various researchers for artifact removal [1-4] use ICA to separate the EEG into its constituent independent components (ICs) and then eliminate the ICs that are believed to contribute to the artifact sources. It is subjective, inconvenient and a time consuming process when dealing with large amount of EEG data. An ICA based method for removing artifacts semi automatically was presented by Delorme et.al [5]. It is automated to flag trials as potentially contaminated, but these trials are still examined and rejected manually via a graphical interface. The automatic artifact removal system proposed in [6] used ICA for demixing the obtained EEG recordings and then a Support Vector Machine (SVM) to classify the separated sources into EEG and artifact signals using their lagged auto-correlation structure. The use of ICA as an artifact removal method does not result into any loss of data as only the artefactual components are rejected as opposed to entire trials, and is applicable to many artifact types. Using the SVM to classify the estimated sources into EEG and artifact signals also eliminates the need for visual inspection and brings the technique one step closer to online and real-time applications. However, one of the disadvantages of this method reported by the same authors is the high dimensionality of the feature space, and it is a contributing factor in the increase of the computational complexity of the method.

Joyce et al [7] proposed an automatic method for the removal of eye movement and blink artifacts from the EEG using the second-order statistics-based blind source identification algorithm (SOBI). However, this method requires six measured EOG channels which are not available if previously recorded data are to be processed. Support vector machines (SVM) have been introduced into eye blink artifact removal by Shoker et al [8]. This method also used the SOBI algorithm to separate the EEG recordings into independent sources and then used the manually selected eye blink artifact components and the remaining non-eye blink components to train an SVM classifier, which was then used to automatically identify the eye blink artifact-independent components. The contribution of this method is that it introduces the machine learning method, SVM classification,

into the identification of the artifact component; thus it enables the automatic implementation of the ICA-based OA removal method. However, the training step in this method was complex, in which a lot of eye blink and non-eye blink artifact-independent components were needed to train an SVM classifier. In [9] an automatic method for removing the eye blink artifact from EEG recordings by using an ICA-based template matching method is proposed. The limitation of this method is that the method could only be used for the removal of eye blink artifacts from EEG.

In [10], several ICA algorithms are quantitatively compared and JADE algorithm has been found to be the best separating ICA algorithm. In this paper, the independent components are obtained by using the JADE algorithm and are classified into either artifact component or neural component. Neural Network is used for the classification of the obtained independent components. A number of neural network approaches may be used to learn classification rules from EEG data. First, a Polynomial Neural Network (PNN) trained by GMDH (Group Method of Data Handling) algorithm is used and secondly, feed-forward neural network classifier trained by a standard back-propagation algorithm is used for classification and the results are compared and the best classifier is identified [11].

A block diagram representation of the proposed work is shown in Figure1. EEGs are acquired and stored. Raw EEGs are separated into statistically independent sources using the JADE algorithm. Features (AR coefficients) are extracted from the ICs and are used to find whether the source contains the ocular artifact. Finally, the sources that are identified as non-artifacts are used to reconstruct the artifact-free EEGs through reprojection.

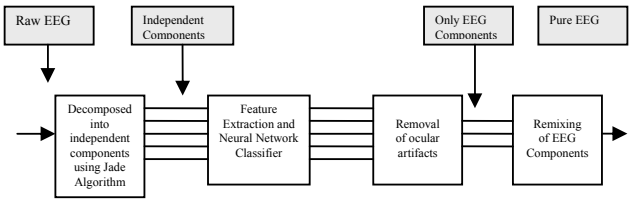


Fig. 1 Automatic ocular artifact removal system

II. POLYNOMIAL NEURAL NETWORK

Polynomial Neural Network [12,13] is a flexible neural network architecture whose structure (topology) is developed through learning. It consists of a set of middle hidden layers which are composed of a number of polynomial nodes. In particular, the number of layers of the PNN is not fixed in advance but becomes generated in fly. The basic structure of PNN is shown in Figure 2. Between the input and output, the connect function and the objective function (typically mean square error) are determined by a training process consisting of three components: connective weights between nodes, analogous to neurons which define the relative contribution of the input; training laws (criteria) that determine the adjustment of the weights during the training; and a transfer function that can be determined by a number of nodes and the connected weights.

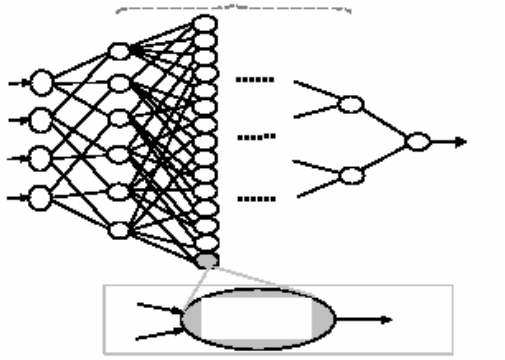


Fig. 1 Basic structure of PNN

The input–output relationship and training procedure of the PNN algorithm can be briefly described in the following steps:

i) Select input variables X = {x1, x2 , …, xn} and divide the available data into training and testing data sets.

ii) Choose a pair of variables and determine the structure of polynomial (pair of input variables and the order). This is in contrast with traditional Artificial Neural Networks (ANN) that use single-variable nodes.

iii) Calculate the connective weights between the nodes of polynomial in the training process.

iv) Identify the contributing nodes at each hidden layer and select the new input variables for the next hidden layer.

v) Check the residual error and stopping criteria, and build up the network relationship for prediction.

It is clear what distinguishes PNN from traditional ANN is its polynomial structure in the nodes and the selection of nodes in the training process. By choosing the most significant input variables and corresponding polynomial order, the optimal extracted polynomial descriptions can be obtained. According to both the selection of nodes at each layer and the generation of hidden layers, the procedure leads to an optimal network structure of PNN. As discussed, PNN is a multi-layered network consisting of neurons whose transfer function g is a short-term polynomial. For example, a non-linear polynomial is given by

$$y = g(\mathbf{x}, w) = w_0 + w_1 x_1 + w_2 x_2 + w_{12} x_1 x_2 \quad (1)$$

where g is the transfer function of the neuron, $\mathbf{x} = (1, x_1, x_2, x_1 x_2)$ is a input vector and $w = (w_0, w_1, w_2, w_1 w_2)$ are the polynomial coefficients or a weight vector. Eqn 1 is a polynomial description of a system model. However, to determine the

coefficients of the polynomial (1) for a general nonlinear system is rather difficult because they depend on not only the number of polynomial terms used but also the number of variables and data. Earlier researches have combined statistics, pattern recognition and least square methods to directly search for the best estimation of the polynomial; most of them, however, ran into problems with the ill conditional data structure and/or limited experimental data [14,15]. The GMDH algorithm is one of the most successful fitting algorithms for obtaining an approximate description of formula (3) by combining polynomial units of two variables in multiple layers and sifting these units with certain sorting criteria [15,16].

III. GROUP METHOD OF DATA HANDLING (GMDH) ALGORITHM

The objective of GMDH algorithm is to build an analytical function (called model) which would behave itself in such a way that the predicted value of the output would be as close as possible to its actual value. The computation process comprises three basic steps [17]:

Step 1 – Select input variables and divide them to pairs as a training set at each layer.

Step 2 – Select new variables as input of the next middle layer and truncate the subsequent computation. With the identification of the optimal output of polynomials at each layer, the selection of new variables enables the network to quickly converge to the target solution. Once the polynomial equations at each unit are selected, the residual error in each layer is further checked to determine whether the set of equations of the model should be further improved within the subsequent computation.

Step 3 – Build the final model and compute the predicted value. The final prediction model can be obtained with selected variables in each layer and the coefficients of polynomials between the connected layers.

The GMDH algorithm secures an optimal structure of the model from successive generations of polynomials after filtering out those intermediate variables that are insignificant for predicting the correct output. Most improvement of GMDH has focused on the generation of the polynomial, the determination of its structure and the selection of intermediate variables. GMDH training algorithms are based on an evolutionary principle, which is performed as following [11]: At the first layer r = 1, using all possible combinations by two from m inputs, generates the first population of neuron- candidates. Since the neuron-candidates are fed by two different inputs, the number L1 of the combinations, or a size of the population at the first layer, is equal to mC_2.

In the first layer, the outputs of the neuron-candidates are $y_1^{(1)} \ldots . y_{L1}^{(1)}$. Then an algorithm selects from this population of the neurons, F best ones, F < L1. The selection of the best neurons is performed in accordance with a predefined fitness function whose value depends on the classification accuracy of the neurons-candidates. Selection criterion is predefined such that when its value is decreased the classification accuracy of the neuron is increased. In the second and next layers r, the size L_r of the population defined by the number F, i.e., $L_r = FC_2$. The generation and selection of the neurons are again performed. The new layers are created while the criterion value is decreased. In Figure 3 an example of the polynomial network consisting of 3 layers the GMDH algorithm grew for m = 5 inputs and F = 4 is shown.

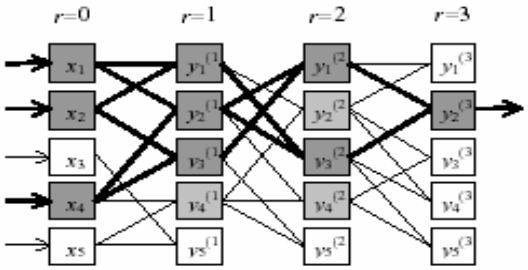


Fig. 3 Polynomial Neural Network trained by GMDH Algorithm

The neuron-candidates that are selected at each layer are depicted as grey boxes. A neuron $y_2^{(3)}$ that provides the best classification accuracy assigns to be an output neuron. A resulting polynomial network, as shown in Figure 3 is a 3 layer network consisting of 6 neurons and 3 input nodes. This network is described by a set of the following polynomials:

$$\begin{aligned} y_2^{(3)} &= g_1(y_1^{(2)}, y_3^{(2)}), \\ y_1^{(2)} &= g_2(y_2^{(1)}, y_3^{(1)}), \\ y_3^{(2)} &= g_3(y_1^{(1)}, y_2^{(1)}), \\ y_1^{(1)} &= g_4(x_1, x_2), \\ y_2^{(1)} &= g_5(x_1, x_4), \\ y_3^{(1)} &= g_6(x_2, x_4), \end{aligned} \tag{2}$$

where $g_1, \ldots . g_6$ are the transfer functions of the neurons. To realize the selection procedure, a dataset are beforehand divided into two subsets. The first of them is used to train the neuron weights and the second to evaluate the classification accuracy of the neuron. Thus the value of the selection criterion depends on the behavior of the neuron on the examples that have not been included in the training subset. This kind of criteria called exterior allows to prevent GMDH type networks from over-fitting [16,18,19]. The transfer function (1), the number of best neurons F, as well as the selection criterion is predefined by users. Setting

these parameters, the users can experimentally search for the best polynomial network.

TRAINING OF GMDH TYPE NETWORKS:

The algorithm for training GMDH type networks is explained below [11]: Let X be a n x m matrix of input data that includes n training samples presented by m features, and y_i^0 is a target vector: $y_i^0 = (y_1^0,....,y_n^0)^T$, $y_i^0 \in \{0,1\}$ and let $D = (X, y_i^0)$ be the dataset. Let the transfer function of neurons be a short-term polynomial of two variables x_1 and x_2 as given in equation 1. In the first layer r = 1, the neurons are connected to the input nodes. The neuron inputs are given by pairs from m variables $x_1,.....,x_m$.

$$x = (1, x_{i1}, x_{i2}, x_{i1}x_{i2}), i_1 \neq i_2 = 1,...,m \quad (3)$$

In the next layers r = 2, 3, …, the input of the neurons are connected to the outputs y_{i1} and y_{i2} of the neurons from the previous (r - 1) layer.

$$x = (1, y_{i1}, y_{i2}, y_{i1}y_{i2}), i_1 \neq i_2 = 1,...,F \quad (4)$$

F is the number of the best neurons selected from the previous layer. For the weight vector w and the k-th example for the input $x^{(k)}$ the output y of the neuron is given by equation 5.

$$y = g(x^{(k)}, w), k = 1,....n \quad (5)$$

For selecting F best neurons, the GMDH uses the exterior criterion calculated on the unseen examples that have not been used for fitting the weights of the neurons. The unseen examples are reserved by dividing the dataset D into two non intersecting subsets $D_A = (X_A, y^0_A)$ and $D_B = (X_B, y^0_B)$ named the training and examining datasets. The user defines the sizes n_A and n_B of these subsets, usually $n_A = n_B$, and $n_A + n_B = n$. A weight vector w* is found that minimizes the sum square error e of the neuron is calculated on a subset D_A.

$$e = \sum_k (g(x^{(k)}, w) - y_k^0)^2, k = 1,...., n_A \quad (6)$$

To find out a desirable minimum, the GMDH fits the neuron weights to a subset D_A by using a Least Square Method (LSM). Once a desirable weight vector w* that minimizes the error e on a subset D_A is found for all L_r neuron candidates of the layer r then the values CRi of the exterior criterion on a subset D_B that has not been used to fit the weights is calculated.

$$CR_i = \Sigma_k (g_i(x^{(k)}, w^*) - y_k^0)^2, k=1,....,n_B, i=1,...L_r \quad (7)$$

The calculated value of CR_i depends on the behavior of the i-th neuron-candidate on the unseen examples of the subset D_B. Therefore the value of CR calculated on entire set D will be high for the neurons with small generalization ability. The values CR_i calculated at the r-th layer are arranged in ascending order:

$$CR_{i1} \leq CR_{i2} .... \leq CR_{iF} \leq .... \leq CR_{iL} \quad (8)$$

So the first F neurons are the best. For each layer r it is defined a minimal value CR_m corresponding to the best neuron, i.e., $CR_m^{(r)} = CR_{i1}$. The first F best neurons are then used at the next r + 1 layer. The outputs of F selected neurons in accordance with equation 4 feed the neuron-candidates at the r + 1 layer. The training and selection of the neurons of this layer performed with the equations 6, 7 and 8 are repeated. The value of $CR_m^{(r)}$ is step-by-step decreased while the number of layers r is increased and the network grows. The value of CR reaches to a minimal point at r=3 and then starts to increase, as shown in Figure 4. In Figure 4 the value of CR_m is minimum at r=3 (third layer) of the polynomial network, i.e., $CR_m^{(1)} > CR_m^{(2)} > CR_m^{(3)}$. At r=3, the value of CR_m becomes minimum. At the next layer r=4 the value of $CR_m^{(4)}$ is increased, therefore in accordance with the exterior criterion the polynomial network has been over-fitted. Since a minimum CR was reached at the third layer the training algorithm is stopped and concluded that a desirable polynomial network has been grown at the r = 3 layer.

(3.4)

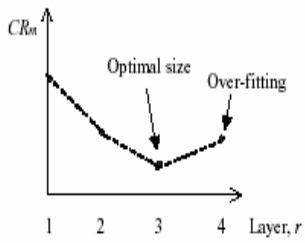


Fig. 4 The value of criterion CR_m at various layers

IV. FEED FORWARD NEURAL NETWORK

Feed forward neural networks (FNN) are composed of layers of neurons, in which the input layer of neurons are

connected to the output layer of neurons. The training process of FNN is undertaken by changing the weights such that a desired input-output relationship is realised. In feed forward architectures, the activations of the input units are set and then propagated through the network until the values of the output units are determined. The network acts as a vector-valued function taking one vector on the input and returning another vector on the output. In this network, the information moves only in one direction, forward, from the input nodes, through the hidden nodes (if any) to the output nodes. Each neuron in one layer is connected to every neuron on the next layer. There are no cycles or loops in the network. Hence information is constantly "fed forward" from one layer to the next. An example of three layer feed-forward neural network is shown in Figure 5.

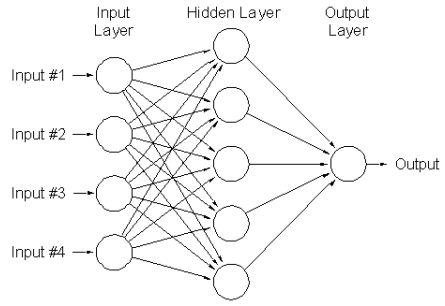


Fig. 5 Basic structure of a Feed Forward Network

By varying the number of nodes in the hidden layer, the number of layers, and the number of input and output nodes, one can classify the given points in arbitrary dimension into an arbitrary number of groups. Hence feed-forward networks are commonly used for classification. Once the user has determined the number of neurons in each layer and the numbers of layers have been decided on, the network's weights must be adjusted to minimize the delta error. A training algorithm is used for this purpose. The most common and widely used algorithm for training multi-layer feed-forward neural networks is the back-propagation algorithm.

BACK PROPAGATION TRAINING ALGORITHM.

Back propagation is a supervised learning technique used for training feed-forward networks. Backpropagation requires that the transfer function used by the artificial neurons be differentiable. Input vectors and the corresponding target vectors are used to train a network until it can approximate a function, associate input vectors with specific output vectors, or classify input vectors in an appropriate way as it is defined. Standard backpropagation is a gradient descent algorithm, as is the Widrow-Hoff learning rule, in which the network weights are moved along the negative of the gradient of the performance function. The term backpropagation refers to the manner in which the gradient is computed for nonlinear multilayer networks. There are a number of variations on the basic algorithm that are based on other standard optimization techniques, such as conjugate gradient and Newton methods. Backpropagation usually allows quick convergence on satisfactory local minima for error in the kind of networks to which it is suited.

A subset of training samples is presented to the feed forward network. In back propagation learning, every time an input vector of a training sample is presented, the output vector o is compared to the desired value d. The comparison is done by calculating the squared difference of the two [20]:

$$Err = (d - o)^2 \tag{9}$$

The value of Err tells us how far away obtained values are from the desired value for a particular input. The goal of back propagation is to minimize the sum of Err for all the training samples, so that the network behaves in the most "desirable" way.

$$Minimize \sum Err = (d - o)^2 \tag{10}$$

Err can be expressed in terms of the input vector (i), the weight vectors (w), and the threshold function of the neurons. Given the fact that decreasing the value of w in the direction of the gradient leads to the most rapid decrease in Err, the weight vectors can be updated every time a sample is presented using the following formula:

$$w_{new} = w_{old} - n\frac{\delta Err}{\delta w} \tag{11}$$

where n is the learning rate. Using this algorithm, the weight vectors are modified so that the value of Err for a particular input sample decreases a little bit every time the sample is presented. When all the samples are presented in turns for many cycles, the sum of Err gradually decreases to a minimum value, and best fitting weights are obtained for the samples.

V. FEATURE EXTRACTION

Neural Network requires input features that exactly represent the true character of the input signals so that the neural network could classify the signals based on those key characters that differentiate between various signals. The main purpose of feature extraction is to reduce the data by measuring certain features that capture the relevant information. So the selection of input feature to the neural network is an important criterion for proper classification of signal.

In this work, Auto Regressive (AR) coefficients are used as the input features for classification. FFT generated spectra can also be used for classification. However, AR coefficients contain most of the information of the signal that would be in the spectrum and therefore a classifier should be able to discriminate between sets of AR coefficients calculated from signals with different spectral properties. The advantage of classifying with raw AR coefficients is that one does not need to search for specific frequency components that contain the information. Not putting limitations on where in the frequency domain the features may be located is inherent when using the

raw AR coefficients. AR coefficients are the better alternative to FFT frequency bands for extracting features when no prior spectral information is assumed [21].

A real valued, zero mean, stationary, nondeterministic, autoregressive process of order p is given by

$$x(n) = -\sum_{k=1}^{p} a_k x(n-k) + e(n) \quad (12)$$

where p is the model order, $x(n)$ is the signal at the sampled point n, a_k is the real valued AR coefficients and $e(n)$ represent the error term independent of past samples [22]. The term autoregressive implies that the process $x(n)$ is seen to be regressed upon previous samples of itself. The error term is assumed to be a zero mean noise with finite variance. In applications, the values of a_k have to be estimated from finite samples of data. In this application AR coefficients are estimated using Burg's method [23,24]. This method is more accurate as compared to other methods like Levinson-Durbin as it uses the data point directly. Furthermore, Burg algorithm uses more data points by minimizing both forward error and backward error. So in this paper to classify EEG and EOG where the significant spectrum of both is (0-16 Hz), AR coefficient using Burg's method of order 6 is used as input feature to the Neural Network rather than power spectrum at various bands [25]. 10 second epochs of raw EEG data consisting of 7 frontal channel recordings are taken and Independent Components are obtained using the JADE algorithm. Each Independent component is divided into 1 second segments that overlap each ½ second. Hence totally 19 one second segments will be available and each one second segment is represented by 6 AR coefficients. Thus the number of input features for one independent component will be equal to 19*6 = 114. The calculated AR coefficients represent the feature vector and are used as inputs to the neural network classifier. The classifier operates under the feature vector and leads to reach a decision of classification.

VI. IMPLEMENTATION

EEG data with ocular artifacts are taken from [26] for testing the proposed algorithm. The scalp EEG is obtained using electrodes placed at locations defined by the 10-20 system and is sampled at a rate of 128 samples/second. EOG interference will be dominant in the EEG recorded from the electrodes F3, Fz, F4, etc., placed on the patient's forehead. Hence, samples from these frontal channels FP1, FP2, F3, F4, F7, F8 and FZ are taken for analysis. Independent Components are obtained by applying the JADE algorithm to the blocks of data, 10 seconds in length, which gives seven independent components containing both ocular and neural components. Then the features (Auto Regressive coefficients) are extracted from each of the Independent Components. The classifier is trained using the features obtained, and it classifies the independent components into ocular and neural components.

Polynomial Neural Network:

The polynomial neural network must be trained with EEG and EOG samples to automatically recognize the Independent component which corresponds to EOG. Samples are collected from the independent components obtained using the JADE algorithm and they are visually inspected by an EEG expert to identify the EEG and EOG Samples. GMDH algorithm requires two non-intersecting subsets as training dataset and examining dataset along with testing dataset. In this experiment the training set consists of 229 samples containing 99 EOG components and 130 EEG components and examining set consists of 150 samples containing 65 EOG components and 85 EEG components and the testing set consists of 2154 samples containing 208 EOG samples and 1946 EEG samples. As discussed in section V each sample was represented by 114 input features i.e., AR coefficients. These features have been used as the input nodes of the Polynomial Neural Network.

During training the weights are updated such that the sum of squares of error of the neuron over all training samples is minimum. delta rule is used for updating the weights [27]. The delta rule is also referred to as Widrow-Hoff rule or Least Mean Square (LMS) rule. According to this rule, the change in weight is directly proportional to the error signal and the input. The error signal is equal to the difference between the desired output value and actual output value of the neuron. Target for EOG is given as 0 and for EEG as 1. Change in weight for the ith input for single output neuron is given by

$$\Delta w_i = \frac{2\alpha}{P} \sum_{P=1}^{P} \delta_p x_i \quad (13)$$

$$\delta_p = t_P - y_{inP} \quad (14)$$

where

α - learning rate

P - number of samples

t_P - target of Pth sample

y_{inP} - obtained input to the neuron

x_i - input to the neuron

Using this delta rule the weights for all the neurons in each layer is calculated and the best neurons in that layer are selected by simulating the neurons with examining datasets. With 114 input features 6441 neurons are obtained in the first layer. Setting F=60, sixty best neurons are selected and outputs of those neurons are fed as input to the next layer. The other layers in the network will have 1770 neurons from 60 inputs from its previous layers. The same procedure is repeated for evaluating the weights and selecting the best neurons in the other layers with the same training and examining datasets. The network is grown layer by layer as

per the GMDH training algorithm. The error value is decreased step-by-step while the number of layers is increased and the network grows. In the 4th layer the error value becomes minimal. In the next layer the value of error is increased, therefore in accordance with the exterior criterion the polynomial network has been over-fitted at layer 5. Because a minimum is reached in the 4th layer, the training algorithm is stopped and is concluded that a desirable polynomial network has been grown at the 4th layer. The errors of the best neurons from layer 1 to layer 5 are:

$$26.354 > 22.07 > 20.273 > 18.75 < 18.76 \quad (15)$$

So a desirable polynomial network for the proposed application has been grown at 4 layers as shown in Figure 6. A polynomial neural network trained by GMDH algorithm has learnt from the training dataset, a classification rule that is described by a set of 15 polynomials given below:

z11=0.75827*1+ 0.50852 *X(97)+ 0.85907 *X(98)-0.10723*X (97)* X (98);
z12=0.70742*1+1.3931 * X (2) +0.4726* X (85) + 0.38296 * X (2)* X (85);
z13=0.75827*1+ 0.50852 *X(97)+0.85907 *X(98)-0.10723* X (97)* X (98);
z14=0.57238 *1+1.377 * X (2) + 0.26579 * X (61) + 0.4594 * X (2)* X (61);
z15=0.70742*1+1.3931 * X (2) +0.4726* X (85) + 0.38296 * X (2)* X (85);
z16=0.65842*1+ 0.35681*X(67) +0.86528*X(92)-0.033001*X (67)* X (92);
z17=0.73729*1+ 0.48345 *X (97)+0.96379 *X(104)-0.1949*X(97)*X (104);
z18=0.74838*1+ 0.52138*X(25) +1.0585* X (56)- 0.11204* X (25)* X (56);
z21= -0.17655 *1 + 0.58637 *z11 + 0.30952 *z12 + 0.70452 *z11*z12;
z22= -0.21672 *1 + 0.64903 *z13 + 0.29864 *z14 + 0.74661 *z13*z14;
z23= -0.20438 *1 + 0.62977 *z15 + 0.31258 *z16 + 0.71985 *z15*z16;
z24= -0.14459 *1 + 0.55598 *z17 + 0.31728 *z18 + 0.63401 *z17*z18;
z31=-0.019766 *1 + 0.48808 *z21 + 0.43042 *z22 + 0.18239 *z21*z22;
z32=-0.052048 *1 + 0.47732 *z23 + 0.50716 *z24 + 0.17377 *z23*z24;
z41(i)= -0.16397 *1 + 0.83886 *z31 + 0.75138 *z32 - 0.42587 *z31*z32;

Figure 6 depicts an appropriate structure of the trained PNN, which consists of input nodes and 15 neurons whose transfer function is described by equation 1. Note that the extracted polynomial rule used only 10 from 114 input variables for classification. The polynomial rule given by a set of 15 polynomials is used to classify the independent components as follows: If the output Z < 0.5, the Independent Component is classified as EOG, otherwise it is EEG. The polynomial network is tested with a testing dataset which is an entirely new dataset which consists of 208 EOG samples and 1946 EEG samples. The performance of the classifier was evaluated by its sensitivity, specificity and average detection rate [28] and is given in Table 3.1. Sensitivity is a measure of the ability of the classifier to detect EOG components and Specificity is a measure of the ability of the classifier to specify EEG components. The sensitivity is very poor for testing dataset. The performance is very much below the expected level.

TABLE I
RESULTS OF POLYNOMIAL NEURAL NETWORK

Data Set	Sensitivity	Specificity	Average Detection Rate
Training data	76 %	79 %	77.5 %
Testing data	55 %	78 %	66.5 %

Feed Forward Network:

The feed-forward neural network (FNN) used in this work contains one hidden layer and one output neuron. The transfer function used in the hidden layer is log-sigmoid function which is given in equation 3.16.

$$y = \frac{1}{1+e^{-w_o - \sum_i^m w_i x_i}} \quad (16)$$

where

x_i - the ith input variable

y - the output of neuron

w_o - bias term

w_i - synaptic weights of neuron

m - number of input variables

The sigmoid transfer function takes the input, which may have any value between plus and minus infinity, and adjusts the output into the range 0 to 1. In the output layer a linear transfer function is used. The neuron weights are initialized by random values. A structure of fully connected FNN is defined by the user. The user must assign the input nodes and preset the number of hidden neurons h.

Fletcher-Reeves conjugate gradient algorithm provided by MATLAB is used for training the network. The basic back propagation algorithm adjusts the weights in the steepest descent direction (negative of the gradient). This is the direction in which the performance function is decreasing most rapidly. It turns out that, although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence. In the conjugate gradient algorithms a search is performed along conjugate directions, which produces generally faster convergence than steepest descent directions. In conjugate gradient algorithms, the step size is adjusted at each iteration. A search is made along the conjugate gradient direction to determine the step size, which minimizes the performance function along that line. If the vicinity of the minimum has the shape of a long, narrow valley, the minimum is reached in far fewer steps than would be the case using the method of steepest descent. To prevent over-fitting the network used regularization method. This improves generalization ability of the network. This involves modifying the performance function, which is

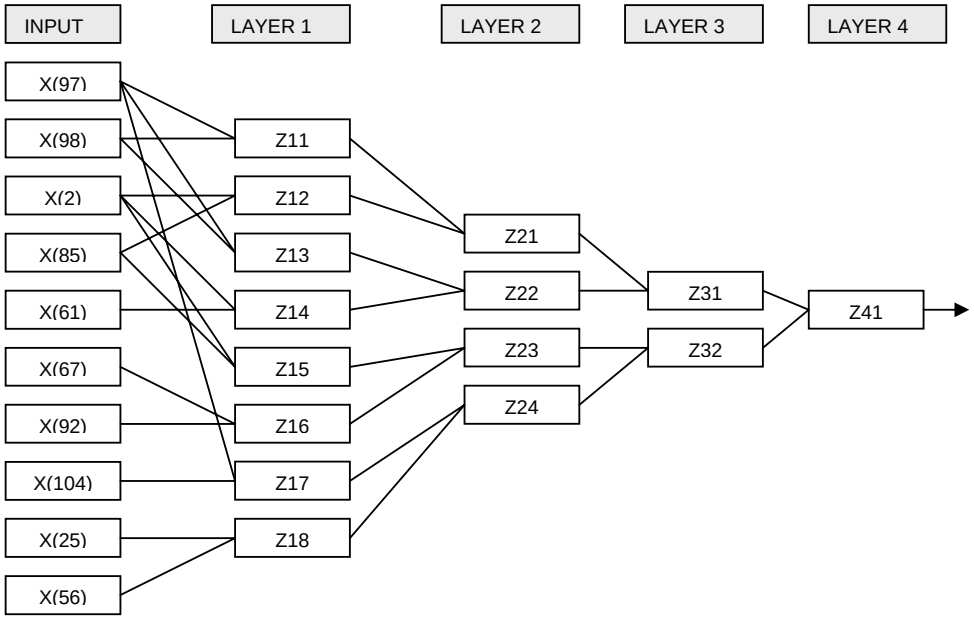


Fig. 6 Polynomial Neural Network to classify EEG and EOG components

normally chosen to be the sum of squares of the network errors on the training set. The typical performance function that is used for training feed-forward neural networks is the mean sum of squares of the network errors. It is possible to improve generalization if the performance function is modified by adding a term that consists of the mean of the sum of squares of the network weights and biases. Using this performance function will cause the network to have smaller weights and biases, and this will force the network response to be smoother and less likely to over fit.

The number of neurons in hidden layer is varied from 15 to 30. The network is trained for various initial weights for 1000 epochs with a training rate of 0.01. The performance ratio is set to 0.5, which gives equal weight to the mean square errors and the mean square weights for obtaining performance function. Significant result is obtained for neural network with 25 neurons in the hidden layer. The dataset used for training and testing the PNN is used feed-forward network. Training dataset consists of 164 EOG samples and 215 EEG samples and the testing set consists of 2154 samples containing 208 EOG samples and 1946 EEG samples. The performance of the feed-forward network in classifying the independent components corresponding to EOG is shown in Table II.

The results show that the Feed-Forward network performs better than Polynomial network. Though the structure of polynomial network is simpler than feed-forward network its successful classification rate is very less than feed-forward network. The performance of feed-forward network is acceptable level for classification of EEG and EOG components

TABLE II
RESULTS OF FEED FORWARD NEURAL NETWORK

Data Set	Sensitivity	Specificity	Average Detection Rate
Training data	96 %	98 %	97 %
Testing data	90 %	94 %	92 %

VII. RESULTS AND DISCUSSION

Samples of EEG data with ocular artifacts available in [26] are analyzed using both JADE-PNN and JADE-FNN. The results obtained for 'cba1ff01.set' file is reported in the paper. EEG data corrupted with ocular artifact is shown in Figure 7. The frontal channels FP1, FP2, F3, F4, F7, F8, and FZ are considered for artifact removal since ocular artifacts are dominant in the frontal electrodes. The independent components obtained from JADE algorithm are shown in Figure 8. Features are extracted from the Independent components and are given as input to the PNN and FNN and the component which corresponds to ocular artifact is identified by the neural classifier. The ocular artifacts are then removed by discarding the identified artifact component and reconstructing the corrected EEG using the remaining components. Figures 9a and 9b shows a segment of EEG recordings contaminated by the eye blink artifact from one of the subjects at the selected electrodes along with the corrected EEG using JADE-PNN and JADE-FNN. To test the performance of the algorithm, the original EEG data and the corrected one using the proposed algorithm is compared by inspecting their visual appearance for each subject's data. The results show that eye blink artifacts were removed with no obvious distortions to the original underlying brain signals using the JADE-FNN algorithm compared to JADE-PNN.

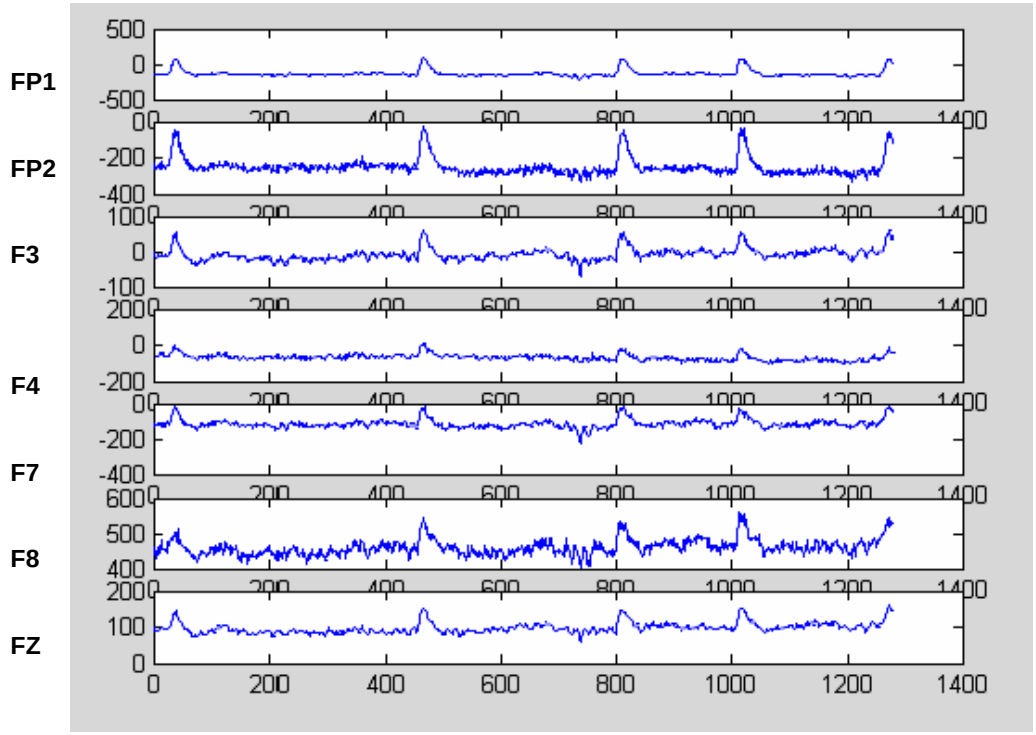


Fig. 7 10s segment of the ocular artifact contaminated EEG recordings

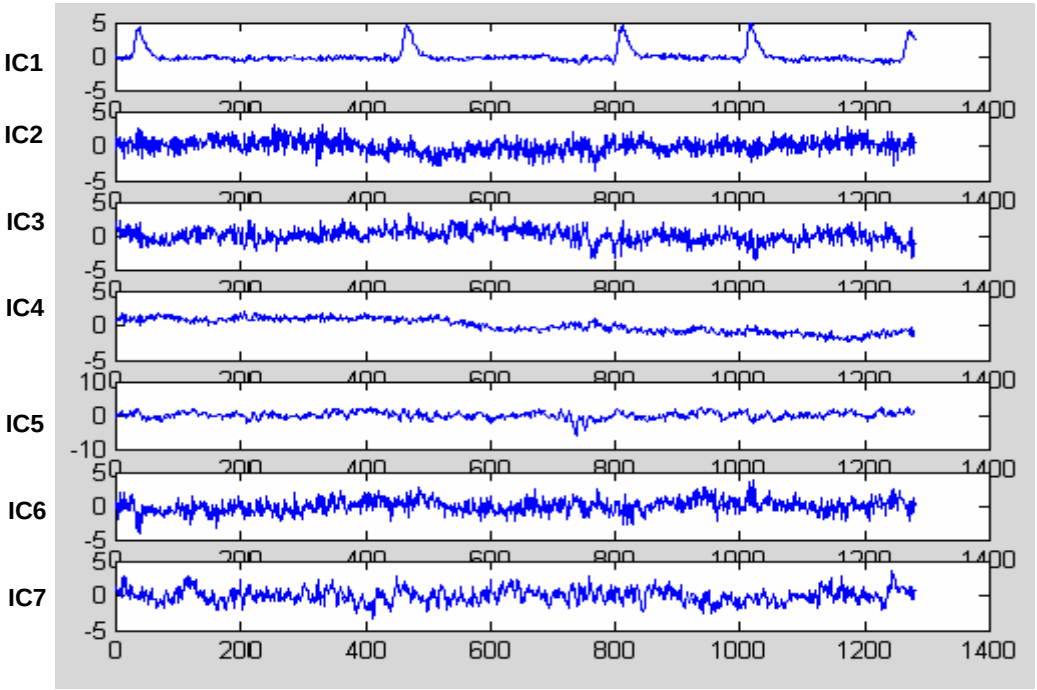
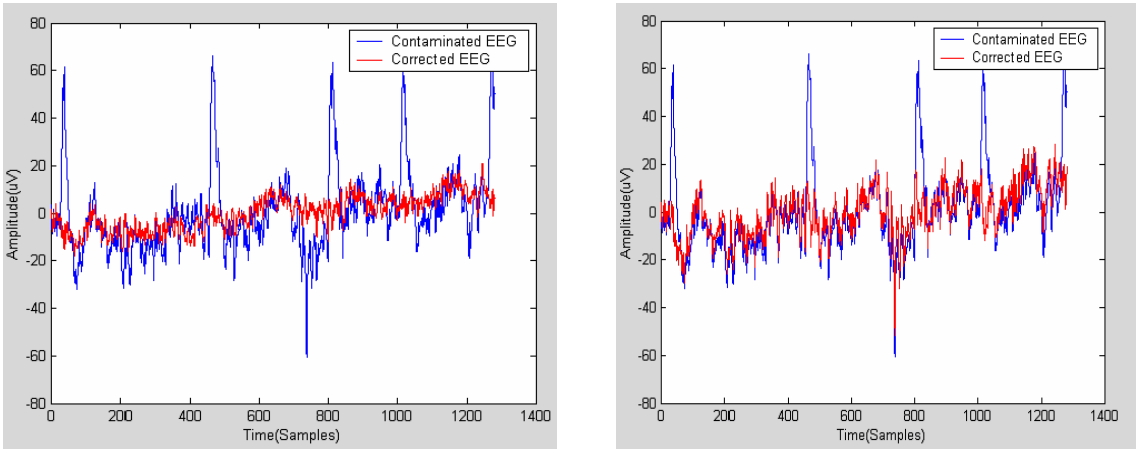
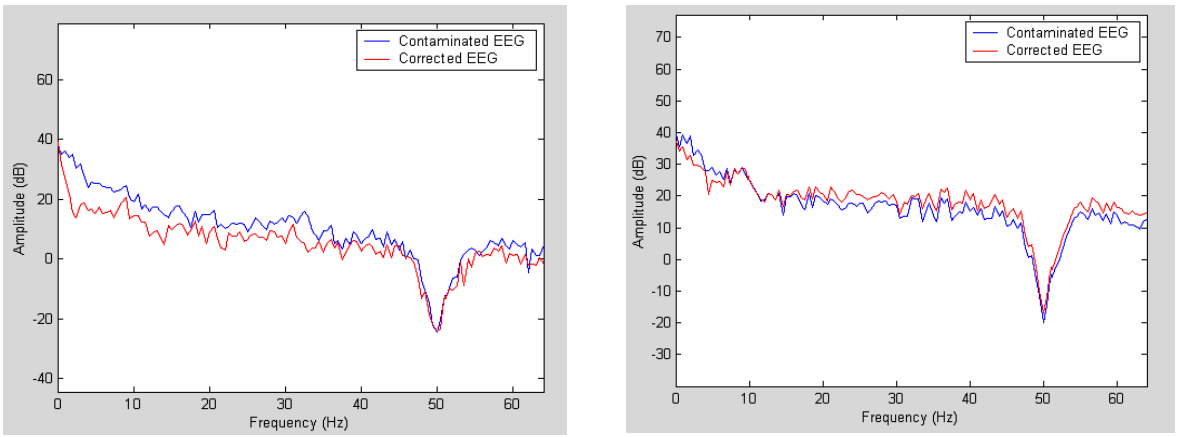


Fig. 8 independent components obtained using JADE algorithm



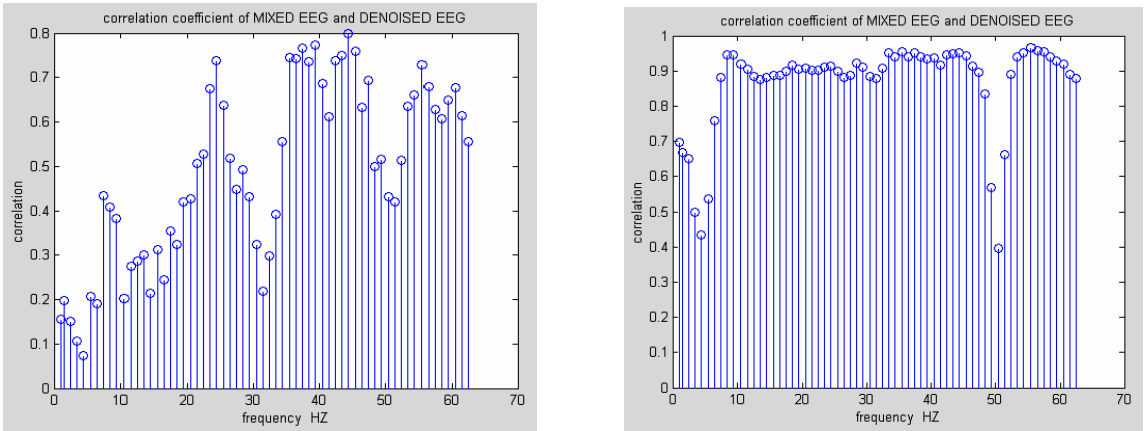
(a) (b)

(a) JADE-PNN (b) JADE-FNN
Fig. 9 Time domain plot for F3 channel



(a) (b)

(a) JADE-PNN (b) JADE-FNN
Fig. 10 Power Spectral Density plot for F3 channel



(a) (b)

(a) JADE-PNN (b) JADE-FNN
Fig. 11 Frequency Correlation plot for F3 channel

Power Spectral Density (PSD) plots are shown in Figure 10. The power spectral density of the signals is found using the window Blackman-Tukey method [24]. Here periodogram smoothing was obtained by applying the Blackman window to the autocorrelation estimate and then taking Fourier transform. The periodogram averaging was done by segmenting the data to obtain several records followed by windowing spectral leakage and finally by averaging the periodogram to reduce variance.

Yet another performance metric for validating the noisy data and denoised data is correlation in the frequency domain. The calculation of the correlation in the frequency domain is equivalent to the correlation in time domain after filtering the time series with the corresponding frequency filter [28]. The frequency correlation between two signals x and y can be calculated using the formula given below:

$$C_{x,y} = \frac{.5 * \sum_{w1}^{w2} \tilde{x}^* \tilde{y} + \tilde{x}\tilde{y}^*}{\sqrt{\sum_{w1}^{w2} \tilde{x}\tilde{x}^* * \sum_{w1}^{w2} \tilde{y}\tilde{y}^*}} \quad (17)$$

where,

w1 and w2 are the window limits

$\tilde{x}$ and $\tilde{y}$ are the Fourier coefficients of x and y.

$\tilde{x}^*$ and $\tilde{y}^*$ are the complex conjugate of $\tilde{x}$ and $\tilde{y}$

Here x and y are noisy EEG and denoised EEG signals respectively. It turns out, that the definition of the correlation of frequency filtered time series is equivalent to calculating the correlation between the (complex) fourier coefficients in the corresponding frequency window. Here the window is chosen with 3 Hz that covers 25 fourier transform coefficients. The 3 Hz window is then moved through the entire spectrum of 64 HZ and correlation coefficient at the corresponding centre frequencies (1.5, 2.5…62.5) are found. The above mentioned formula calculates 125% correlation coefficient assuming the mean of the two signals as zero. So the mean of EEG signal is made zero by subtracting the mean of the entire signal with each value of the signal. The frequency correlation between noisy data shown in Figure 9 and denoised data obtained using JADE-PNN and JADE-FNN is found and the plots are shown in Figure 11. It is evident from Figure 11 that compared to JADE-PNN, JADE-FNN performs better in removing the retaining the high frequency components (13Hz-64 Hz) at the same time removing the ocular artifacts which belong to low frequency range (0 Hz – 13 Hz). Hence from the time domain plots, PSD's and frequency correlation plots, it is clear that JADE-FNN minimizes the amplitude of the ocular artifact and retains the background EEG, much better than JADE-PNN.

VIII. CONCLUSION

Considering the classification rate performance of the automatic detection methods is increasing in the order of polynomial neural network and feed-forward neural network. Even though the classification rate of FNN is better than PNN, the success of neural network based classifier solely depend on the training samples. If the training samples are carefully selected such that those types of EEG and EOG sample patterns occur frequently in decomposed EEG recordings, this technique proves to be efficient. Hence in order to make the automated process independent of training samples, Kalman predictor can be used for classifying EEG and EOG components.

REFERENCES

[1] Lagerlund TD, Sharbrough FW, Busacker NE, "Spatial filtering of multichannel electroencephalographic recordings through principal component analysis by singular value decomposition", Clinical Neurophysiology, 14(1), 1997, pp 73 – 82.

[2] Scott Makeig, Tzyy-Ping Jung, Anthony J Bell, Terrence J Sejnowski, "Independent Component Analysis of Electroencephalographic data", Advances in Neural Information Processing Systems 8 MIT Press, Cambridge MA, Vol (8), 1996, pp 145-151.

[3] Tzyy-Ping Jung, Scott Makeig, Colin Humphries, Te-won Lee, Martin J Mckeown, Vincent Iragui and Terrence J Sejnowski, "Extended ICA removes Artifacts from Electroencephalographic recordings", Advances in Neural Information Processing Systems 10, MIT Press, Cambridge, MA, 1998, pp 894-900.

[4] Vigario R, Jaakko Sarela, Veikko Jousmaki, Matti Hamalainen, Erkki Oja, "Independent Component Approach to the Analysis of EEG and MEG Recordings", IEEE Transactions on Biomedical Engineering, Vol 47, No.5, May 2000 pp 589-593.

[5] Delorme.A, Makeig.. S & Sejnowski T, "Automatic artifact rejection for EEG data using high-order statistics and independent component analysis", Proceedings of the Third International ICA Conference, 2001, pp 9-12 .

[6] N.Nicolaou and S.J.Nasuto, "Temporal Independent Component Analysis for automatic artefact removal from EEG", 2nd International Conference on Medical Signal and Information Processing, Malta, 2004, pp 5-8.

[7] Carrie A.Joyce, Irina F Gorodnitsky and Marta Kutas, "Automatic removal of eye movement and blink artifacts from EEG data using blind component separation", Psychophysiology, 41 Issue 2, 2004, pp 313-325

[8] Shoker L, Sanei S and Chambers J, "Artifact removal from electroencephalograms using a hybrid BSS–SVM algorithm" IEEE Signal Process. Lett. 12, 2005 pp 721–4

[9] Yandong Li, Zhongwei Ma, Wenkai Lu and Yanda Li, "Automatic removal of the eye blink artifact from EEG using an ICA-based template matching approach" Physiological Measurement 27, 2006, pp 425-436

[10] V Krishnaveni, S Jayaraman, Chaitanya Mathi, N Malmurugan, K Ramadoss, "Quantitative Evaluation of Signal Separation Algorithms for removal of ocular artifacts from EEG", National Journal of Technology, No.2, Vol.1, 2005, pp 47-53.

[11] Vitaly Schetinin Theorie Labor, Friedrich-Schiller, "Polynomial Neural Networks Learnt to Classify EEG Signals," NIMIA-SC October 2001.

[12] S.C Satapathy, P.K. Dash, G.Panda, B.B.Mishra, "Polynomial Neural Swarm Classifier" MMU International Symposium on Information and Communication Technologies MUSIC 2004.

[13] X.Wang, L.Li, D.Lockington, D.Pullar, D.S.Jeng, "Self-Organizing Polynomial Neural Network for Modelling Complex Hydrological Processes" Research Report No R861, University of Sydney, December 2005.

[14] Nalimov V.C. and Chernova N. A., "Statistical methods of planning the extremum experiments, Moscow, 1965.

[15] Ivakheneko A G, "Polynomial Theory of Complex System, IEEE Transactions on Systems, Man, and Cybernetics, SMC-1(4), 1971, pp 364-378.

[16] Farlow, S.J., "Self-organizing Methods in Modeling: GMDH Type Algorithms", Marcel Dekker, New York, 1984.

[17] Chang, F. J. and Hwang, Y.Y., "A self-organization algorithm for real-time flood forecast, Hydrological processes", 13, 1999, pp 123-138.

[18] H.R. Madala, A.G. Ivakhnenko. "Inductive Learning Algorithms for Complex Systems Modeling," 1994
[19] J.A. Müller, F. Lemke, A.G. Ivakhnenko., "GMDH Algorithms for Complex Systems Modeling. Mathematical and Computer Modeling of Dynamical Systems," Vol. 4, 1998, pp. 275-315.
[20] http://cse.stanford.edu/class/sophomorecollege/projects00/neural/networks/Architecture /feedforward/html
[21] Mark Polak Aleksandar Kostov, "Feature extraction in development of brain-computer feature extraction in development of brain-computer," Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Vol. 20, No.4 1998.
[22] Nai-Jen, Huan and Ramaswamy Palaniappan , "Classification of mental tasks using fixed and adaptive autoregressive models of EEG signals," Proceedings of the 26th Annual International Conference of the IEEE EMBS, September 2004.
[23] Burg, J.P., "A new analysis technique for time series data," NATO Adv. Study Inst. Signal Processing With Emphasis on Underwater Acoust., Aug. 1968
[24] Monson H Hayes, "Statistical Digital Signal Processing and Modeling" John Wiley & Sons, Inc, 1996
[25] Keirn, Z.A., and Aunon, J.I, "A new mode of communication between man and his surroundings," IEEE Transactions on Biomedical Engineering, Vol. 37, No.12 December 1990, pp. 1209-1214.
[26] http://www.sccn.ucsd.edu/~arno/famzdata/publicly_available_EEG_data.html
[27] Sivanandam S.N. Paulraj M., "Introduction to artificial neural networks," Vikas publishers, 2004, pp. 39-41.
[28] L. Tarassenko, Y.U.Khan, M.R.G Holt, "Identification of inter-ictal spikes in the EEG using neural network analysis" Inst. Elect. Eng._Proc. Sci Meas. Technol., Vol 145, No.6, 1998, pp 270-278.
[29] Andreas Jung, Dissertation on "Statistical analysis of biomedical data" University of Regensburg, 2003.