THESIS

SINGLE-TRIAL P300 CLASSIFICATION USING PCA WITH LDA AND NEURAL NETWORKS

Submitted by

Nand Sharma

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2013

ABSTRACT

SINGLE-TRIAL P300 CLASSIFICATION USING PCA WITH LDA AND NEURAL NETWORKS

A brain-computer interface (BCI) is a device that uses brain signals to provide a non-muscular communication channel for motor-impaired patients. It is especially targeted at patients with 'locked-in' syndrome, a condition where the patient is awake and fully aware but cannot communicate with the outside world due to complete paralysis. The P300 event-related potential (ERP), evoked in scalp-recorded electroencephalography (EEG) by external stimuli, has proven to be a reliable response for controlling a BCI. The P300 component of an event related potential is thus widely used in brain-computer interfaces to translate the subjects' intent by mere thoughts into commands to control artificial devices. The main challenge in the classification of P300 trials in electroencephalographic (EEG) data is the low signal-to-noise ratio (SNR) of the P300 response. To overcome the low SNR of individual trials, it is common practice to average together many consecutive trials, which effectively diminishes the random noise. Unfortunately, when more repeated trials are required for applications such as the P300 speller, the communication rate is greatly reduced. This has resulted in a need for better methods to improve single-trial classification accuracy of P300 response. In this work, we use Principal Component Analysis (PCA) as a preprocessing method and use Linear Discriminant Analysis (LDA)and neural networks for classification. The results show that a combination of PCA with these methods provided as high as 13% accuracy gain while using only 3 to 4 principal components. So, PCA feature selection not only increased the classification accuracy but also reduced the execution time of the algorithms by the resulting dimensionality reduction. It was also observed that when treating each data sample from each EEG channel as a separate data sample, PCA successfully separates out the variance across channels.

ACKNOWLEDGEMENTS

I would like to thank Dr. Charles Anderson for his guidance, suggestions, and reviews and for introducing me to the exciting field of BCI. Thanks also to Dr. Michael Kirby for introducing me to many joys of exploring mathematical depths of the methods used in this work. I am also grateful to Dr. Chris Peterson for serving on my committee. Special thanks to the BCI group at CSU that collected the data used in this work. Last, but not the least, I'm thankful to all my friends, family and my wife Neelam who supported me during this work.

CHAPTER 1

Introduction

Various neurological diseases can disrupt the neuromuscular channels through which the brain communicates with the external world. Many such diseases can restrict both verbal and nonverbal communication. For patients suffering from such ailments, assistive technologies can restore some communication if they still have some residual motor ability.

But in certain cases like hemorrhage in the anterior brain stem or degenerative neuromuscular diseases like amyotrophic lateral scleriosis (ALS), the patients suffer from a total motor paralysis [5]. This results in a condition known as *locked-in syndrome*, wherein the patient is awake and fully aware but cannot communicate with the outside world due to complete paralysis. For such "locked-in" patients, there is a need for an assistive technology that needs no muscular activity whatsoever. It is for such patients that a brain-computer interface (BCI) is a need.

A brain-computer interface (BCI) is a device that uses brain signals to provide a direct, non-muscular communication channel between brain and the outside world [35, 34, 32]. The idea underlying BCIs (Figure 1.1) is to measure electric, magnetic, or other physical manifestations of the brain activity and to translate these into commands for a computer or other devices [24, 18].

For patients with locked-in syndrome,the P300 event-related potential (ERP), evoked in scalp-recorded electroencephalography (EEG) by external stimuli, has proven to be a reliable response for controlling a BCI [11]. Recent studies have demonstrated that a P300-based BCI trained on a limited amount of data can serve as an effective communication device [12, 16]. The P300 component of an event related potential is thus widely used in brain-computer

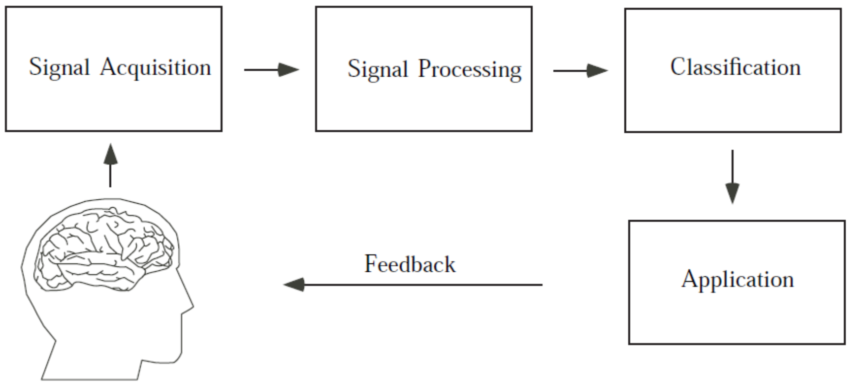

FIGURE 1.1. A typical BCI. The image is taken from [18].

interfaces (BCIs) to translate a subject's intent by mere thoughts into commands to control artificial devices. In this study we present comparison of some classification methods to classify an EEG signal based on the presence of P300 component.

1.1. BCI and P300

Types of BCIs can be broadly classified into two categories—those that use an external stimulus, and those that don't [24]. In the first method, the external stimuli cause changes in neurophysiologic signals called event-related potentials (ERPs) [18, 25] which are used to identify a user's response to the stimuli presented. In the second method, users generate certain detectable patterns of neurophysiologic signals by concentrating on a specific mental task. For example, imagination of hand movement can be used to modify activity in the motor cortex [18].

For recording the activity of the brain, the electroencephalogram (EEG) is the method of choice for a BCI due to their fast responsivity and covariation with cognitive processes [5]. Although invasive methods that use electrocorticography (ECoG) signals using implanted electrodes are more accurate, the non-invasive methods are more attractive because of their ease of use by patients; the non-invasive methods have also been shown to be comparable to implanted electrodes [33] when used with appropriate machine learning algorithms.

The EEG non-invasive recordings are done from a set of electrodes placed directly on the scalp using the International 10-20 system (Jasper, 1958) [28] as shown in Figure 1.2. Each electrode detects the electric potential of synchronized neuronal activity occurring in that area of the brain. This system is based on the relationship between the location of an electrode and the underlying area of cerebral cortex. The ”10” and ”20” refer to the fact that the actual distances between adjacent electrodes are either 10% or 20% of the total front-back or right-left distance of the skull. Each site has a letter to identify the lobe and a number to identify the hemisphere location. In this system, F, T, C, P and O stand for frontal, temporal, central, parietal, and occipital lobes; note that there exists no central lobe, the ”C” letter is used for identification purposes only. A ”z” (zero) refers to an electrode placed on the midline. Even numbers (2,4,6,8) refer to electrode positions on the right hemisphere, whereas odd numbers (1,3,5,7) refer to those on the left hemisphere.

In this work, the data is obtained through non-invasive electroencephalographic (EEG) recordings. Moreover, the experiments are run on EEG data using only a subset of 8 electrodes that have already been found to be meaningful for P300 classification [33]. A smaller number of electrodes is also better for practical reasons of lower cost and higher usability for target patients. These 8 electrodes used for this work are ’F3’,’F4’,’C3’,’C4’,’P3’,’P4’,’O1’,’O2’.

1.2. P300 and P300 Speller

ERPs are characterized by their voltage amplitude and their latency in relation to stimulus onset [5]. One of the most studied and used ERP components of the EEG is the so-called

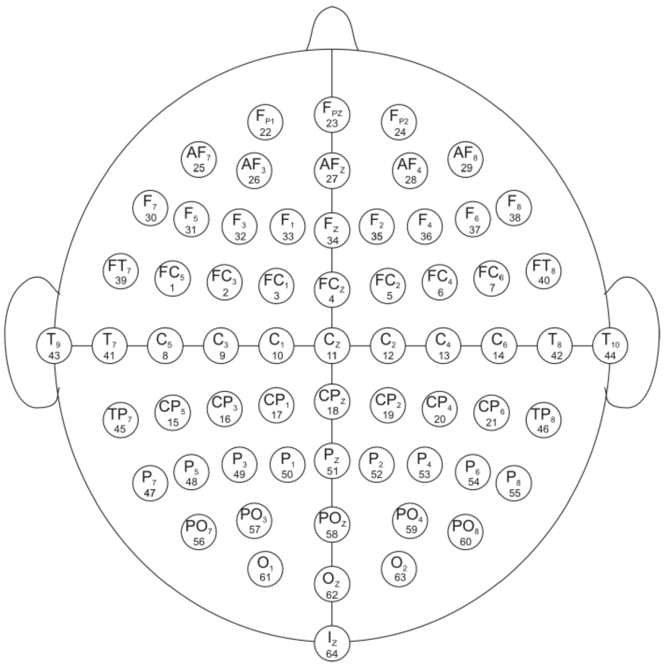

FIGURE 1.2. International 10-20 System. The image is taken from [7].

"P300". A P300 ERP is characterized by a positive peak about 300ms after the stimulus onset [5, 25] (Figure 1.3). It is elicited when subjects encounter a rarely occurring, but expected, stimulus among the presented stimuli. If subjects are assigned the task of assigning a category to each of the stimuli in a series of stimuli of two types, and if one of the two types occurs rarely, a P300 ERP is seen in the EEG [23] for the rare stimuli. This experimental paradigm based on extensive research has been called 'oddball' paradigm [23], the rarely occurring stimuli being the 'oddballs'.

Farwell and Donchin utilized this characteristic of the P300 to design a BCI in 1988 [23] that is called a 'P300 Speller'. This first BCI of its kind lets a user type one letter at a time using the EEG signals captured through an electrode cap, and needs no muscular movement. There has been a lot of research following their work on improving their setup and many variations of the setup have emerged. Their setup used a 6 X 6 grid containing

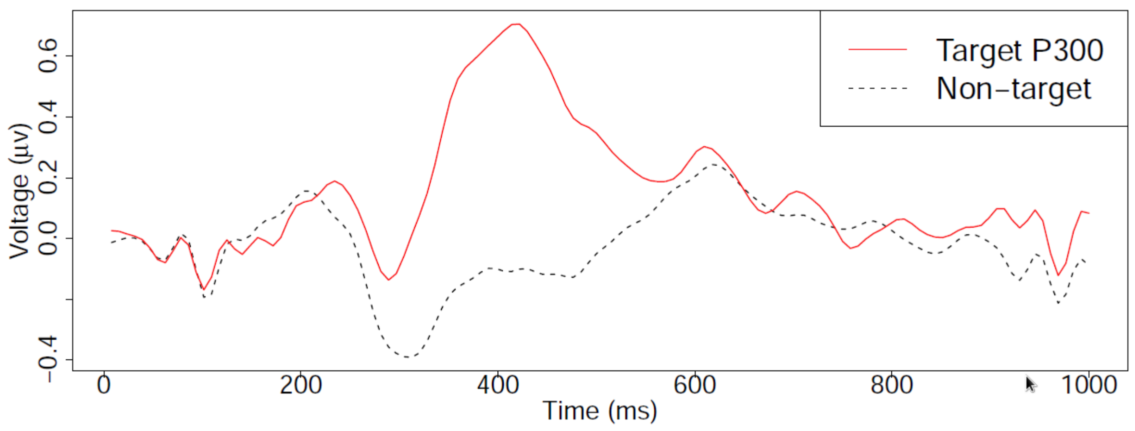

Figure 1.3. P300 ERP. The image is taken from [9].

letters of alphabet and some one-word commands, as shown in Figure 1.4, that is displayed to the users. The rows and columns of this grid flash at a constant rate, about eight times per second. The users need to focus their attention on the letter that they want to type. So intensification of each row/column containing the target letter becomes a relevant event while other rows/columns are not relevant. There are 12 possible events (6 rows and 6 columns), only 2 of which are relevant. A P300 response is elicited each time the row or column containing the target letter is intensified. This falls into the oddball paradigm because the random intensification is unpredictable but expected, and the probability of the target stimulus is only 1 in 6, i.e., about 16%. It is then possible to identify the target letter from the intersection of target row and target column each of which elicit a P300. Therefore, identifying an intended character is broken up into two distinct tasks: the classification of the target row, and the classification of the target column. To classify a row (column), a window of EEG (e.g., a one-second window) is taken starting at the flashing of each of the six rows (columns). As definitions for P300 speller paradigm vary in literature, it is made clear that for this work, each of these windows of EEG will be referred to as a 'subtrial' and a set of row (column) flashings (six in number which contain one target row(column)) will be called a trial [23].

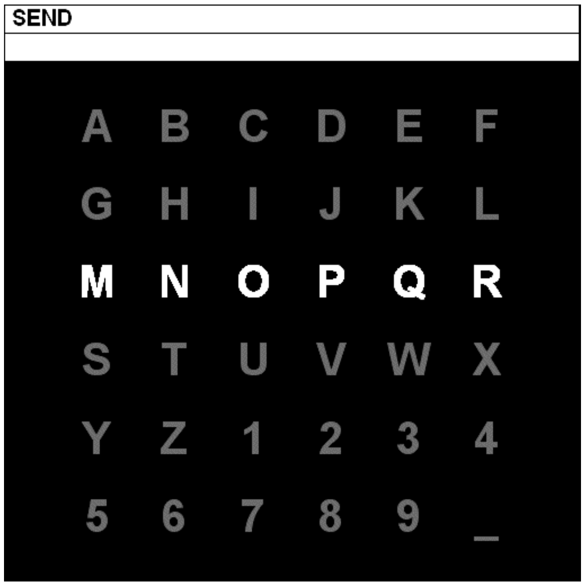

FIGURE 1.4. P300 speller grid. The image is taken from [21].

One of the main challenges in P300 classification is the low signal-to-noise ratio (SNR). As the EEG recorded from scalp contains a lot of noise from ongoing electrical activity in the brain, a P300 is hard to separate from the resulting noisy signal. The problem of low SNR is usually overcome by averaging together many subsequent trials which cancels out most of the noise, and makes P300 detection possible. But this approach of averaging comes at the cost of reduced communication rate. Farwell and Donchin reported in their pioneering work that it needed averaging of between 20 and 40 trials to achieve an accuracy of over 80% [23], with a communication rate of about 12 bits per min or about 2.3 characters per minute. Although their pioneering work did establish the feasibility of a P300 based speller, the communication rate was painfully slow for a practical use. This lead to much work in the last few decades which focused on improving the accuracy and communication speed of a P300-based BCI like P300 speller. So, one of the focus areas of this research has been the development of algorithms that can reduce the number of trials required to achieve a reliable P300-based BCI. But as it is amply demonstrated that averaging of trials stabilizes the P300 amplitude by removing noise, the challenge is also seen as improving single trial accuracy.

1.3. Related Work

The last few decades have seen a lot of research on P300 classification, with the eventual aim of achieving single-trial based reliable P300 BCI [24]. But a literature survey of this research does not show any single P300 classification method to be the state of the art. Krusienski, et al., [21] report the results of a comparison of different classifier algorithms, which shows that stepwise linear discriminant analysis (SWLDA) and support vector machines (SVMs) perform well compared to the other classifiers. In [22] by using SWLDA as the classification method, Krusienski, et al., achieved at least 60% accuracy for all participants. Three of the five participants performed above 90% accuracy with averaging about 15 trials. Sellers and Donchin [12] achieved comparable average results for healthy and ALS patients using SWLDA. Serby, et al., [16] used matched-filtering with independent component analysis (ICA) to achieve a communication rate of 5.45 symbols/min with an accuracy of 92.1% which averaged roughly 15 trials. When the detection was made in real-time by online testing with the same six subjects, the average communication rate achieved was 4.5 symbols/min with an accuracy of 79.5% which averaged roughly 18 trials. On the same lines, a survey of submissions to BCI Competition II and BCI Competition III shows that several very different approaches like SVM, ICA, LDA, peak-picking methods were able to achieve 100% accuracy using between 4 and 15 averaged trials [8, 19, 36] on the BCI Competition II data.

This short list shows that there are many different approaches that all work well, and also that one particular method is not clearly better than others. Then there are other challenges to P300 classification, namely subject-dependence of EEG and even a session-to-session variation in EEG responses of the same subject. A recent review of the BCI field by Mak, et al., [26] concludes that a lot more work is still needed to create a truly reliable P300 speller.

1.4. Contributions

In this work, we use Principal Component Analysis (PCA) as a preprocessing method and use Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA) and neural networks for classification. PCA has been shown to work well for P300 classification [9, 15]. In [9], the author compared various blind source separation methods as preprocessing methods for P300 classification, and it was seen that PCA generally worked better than the other methods like Independent Component Analysis (ICA) and Maximum Noise Fraction (MNF) for P300 classification. This work builds upon the work done by Cashero [9] and tests if classification methods other that Support Vector Machines (SVM) will work well in conjunction with PCA for P300 classification.

Among the classification methods, the choice of LDA is also driven by success of this method and its variants like Stepwise LDA (SWLDA) for P300 classification. Although SWLDA has been shown to work very well for higher accuracy, it is an expensive method with high processing time and is therefore not considered suitable for an online P300 speller system [24]. Considering that, we have preferred to consider LDA which is lightweight and provides good performance generally for P300 classification. QDA is chosen to compare LDA with, and to see whether a linear or a nonlinear method works better for P300 classification.

Our choice of Neural Networks (NN) as a method of study for P300 classification is primarily driven by our hypothesis that they should be a good choice for P300 classification. There have been studies employing Neural Networks for P300 classification [15, 29], and many have shown promise. There is very little work done using PCA and NN as a combination for P300 classification. As NN provide a lot of flexibility in terms of how the 'derived' data is created based on flexibility in the number of hidden units, it could be a useful method for noisy data like EEG, especially if PCA provides a good set of source components. SVM is another method which has been shown to work well for P300 classification. We don't include that in this study due to the fact that a study using PCA with SVM has already been covered in [9]. To keep our focus on the effect of PCA on these classification methods, we have tested only single trial accuracies.

1.5. Overview

The thesis is laid out as follows. Chapter 2 develops the mathematical background for all the algorithms that are used in the experiments for this work. The classification methods, LDA, QDA, Neural Networks, and the optimization algorithm used for NN and PCA are developed in this chapter. Chapter 3 describes the datasets used and details the steps used in the experiments. Chapter 4 explains the experiments as they were performed and what was learned along the way, and the results of the experiments. Chapter 5 concludes by providing a summary of the findings and identifies avenues for future work.

CHAPTER 2

Classification and Preprocessing Methods

2.1. LDA and QDA

LDA and QDA [17, 6] belong to a class of classification methods that model *discriminant functions* for each class and then classify a given data sample to the class with the largest value for its discriminant function. LDA and QDA model the posterior probabilities for the purpose of defining these discriminant functions. Representing a data sample by variable X and the class label by variable C, $X \in \mathbb{R}^d$ where d is the dimension of data sample X or equivalently represents the number of features or predictors in each sample X, and $C \in \{1, 2, ..., K\}$ where there are K classes. We thus need the class posteriors $p(C \mid X)$ for a given X and C. Suppose $f_k(x)$ is the class-conditional density of X in class $C = k$, and let π_k be the prior probability of class k, with $\sum_{k=1}^{K} \pi_k = 1$. Applying the Bayes theorem gives us

$$p(C = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l} \tag{2.1}$$

While many different models can be used to model the class-conditional density, LDA and QDA use multivariate Gaussians to model each class density so that

$$f_k(x) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} e^{-1/2(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)} \tag{2.2}$$

where Σ_k and μ_k are Covariance matrix and the mean of the Gaussian distribution. If we assume that the classes have a common covariance matrix given by $\Sigma = \sum_{k=1}^{K} \frac{N_k}{N}\Sigma_k$, where N_k is the class k samples and N is the total number of samples of all classes, then we get LDA. In comparing two classes k and l, it is sufficient to look at the log-ratio

$$\ln \frac{p(C = k \mid X = x)}{p(C = l \mid X = x)} = \ln \frac{f_k(x)}{f_l(x)} + \ln \frac{\pi_k}{\pi_l} \tag{2.3}$$

Plugging in the $f_k(x)$ and $f_l(x)$ as defined in (2.2), we get

$$\ln \frac{p(C = k \mid X = x)}{p(C = l \mid X = x)} = \ln \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k - \mu_l)^T \Sigma^{-1}(\mu_k - \mu_l) + x^T \Sigma^{-1}(\mu_k - \mu_l) \tag{2.4}$$

which is an equation linear in x. So, equation (2.4) gives us a decision rule for LDA—if the log ratio is positive, the sample x is classified as belonging to class k and to class l if it is negative. The value being zero implies the decision boundary which is linear in x. It's obvious that the *linear discriminant functions*

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + \ln \pi_k \tag{2.5}$$

are an equivalent description of the decision rule for LDA. So, the class of a new sample x is simply

$$C(x) = \arg\max_k \delta_k(x)$$

Now if we drop the assumption of a common covariance matrix for all classes, then the convenient cancellation of terms leading to (2.4) doesn't happen and we get the quadratic discriminant functions (QDA),

(2.6) $$\delta_k(x) = -\frac{1}{2}\ln|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \ln\pi_k$$

The decision boundary in this case between each pair of classes k and l is described by a quadratic equation given by $\delta_k(x) = \delta_l(x)$. Again, the class of a new sample x is obtained as

$$C(x) = \arg\max_k \delta_k(x)$$

The LDA and QDA are also called Generative models, as they make the assumption of Gaussian distribution of the data, and base the classification of samples on that assumption. The parameters of the Gaussian distributions are estimated using the training data. Class priors are calculated based on the number of samples of each class present in the training data, as

$$\pi_k = \frac{N_k}{N},$$

where N_k is the class k samples and N is the total number of samples of all classes. The class means are estimated as the means of the data of a particular class in the training data, so

$$\mu_k = \sum_{c_i=k} x_i/N_k$$

where c_i is the class of sample x_i. Similarly Σ_k is the covariance matrix for each class based on the data of that particular class in the training data.

2.2. Neural Networks

Neural Networks (NN) [17, 6] are primarily employed in machine learning as nonlinear regression and classification method. While there are many variations and flavors of NN like Recurrent NN, multi-layer perceptron, etc., [17, 6, 37], we use a single hidden layer NN for this work. This basic neural net, sometimes called the single hidden layer back-propagation network, or two layer perceptron is a two-stage regression or classification model. It consists of an input layer, a hidden layer and an output layer as shown in Figure 2.1.

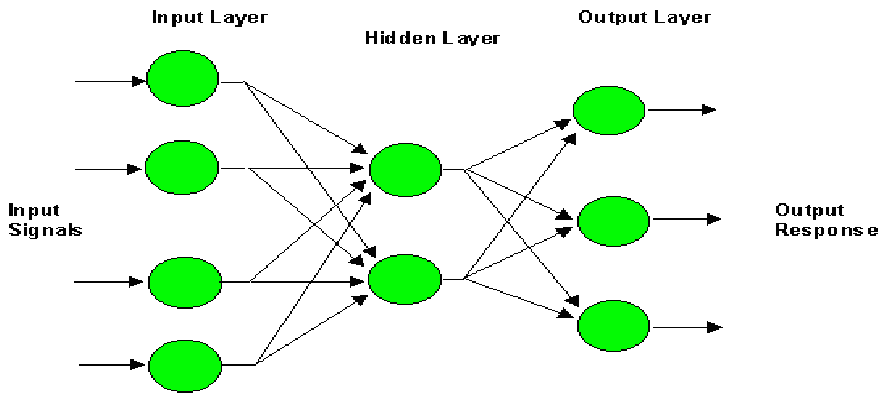

Figure 2.1. Basic configuration of a feedforward single-layer perceptron.

In the case of K-class classification, there are K output units, and each of the K output units models the probability of class k so that

$$Y_k \in [0,1] \ \forall \ Y_k, k = 1, ..., K$$

Derived features Z_m are created from linear combinations of the inputs, followed by a nonlinear activation function. The output Y_k is modeled as a function of linear combinations of the Z_m,

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X), \;\; m = 1, ..., M, \tag{2.7}$$

$$Y_k = \beta_{0k} + \beta_k^T Z, \;\; k = 1, ..., K, \tag{2.8}$$

$$f_k(X) = g_k(Y), \;\; k = 1, ..., K \tag{2.9}$$

where $Z = (Z_1, Z_2, ..., Z_M)$, and $Y = (Y_1, Y_2, ..., Y_K)$. $\sigma(v)$ is the activation function and is chosen to be a *sigmoid* defined as

$$\sigma(v) = \frac{1}{1 + e^{-v}}$$

The output function $g_k(Y)$ allows a final transformation of the vector of outputs Y. We choose the softmax function for this :

$$g_k(Y) = \frac{e^{Y_k}}{\sum_{l=1}^{K} e^{Y_l}} \tag{2.10}$$

which results in a multilogit model, and produces positive estimates that add to one. Treating these outcomes as probabilities for the corresponding class, we use negative log-likelihood as the objective function to minimize. This negative log-likelihood objective function is defined as

$$LL(\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K} T_{ik} log f_k(x_i) \tag{2.11}$$

where θ denotes the complete set of weights of the network, which consist of $\{\alpha_{0m}, \alpha_m,\ m = 1, ..., M\}$ and $\{\beta_{0k}, \beta_k,\ k = 1, ..., K\}$, and T_{ik} is the k^{th} component of the target indicator variable T_i for i^{th} training sample x_i. Finally, the corresponding classifier is defined as

$$C(x) = \arg\max_k f_k(x)$$

With the sigmoid activation function and log-likelihood error function, the neural network model works as a linear logistic regression model in the hidden units, and all the parameters are estimated by maximum likelihood. But by using the nonlinear transformation σ, it becomes a non-linear model of inputs X. Another interesting aspect of this model is that the number of hidden units can be varied to adjust the non-linearity of the model. If there are no hidden units, the model becomes a simple linear logistic regression model over input data. In this work we use both linear and non-linear versions of the neural network model and call them LR and NLR respectively.

The error function $LL(\theta)$ can be minimized by variety of approaches. One of the standard approaches is the gradient descent, called *back-propagation* in the neural network setting. But a conjugate gradient method called Scaled Conjugate Gradient has been more successful and faster for this optimization problem and we use that in this work.

2.3. Scaled Conjugate Gradient method

The Scaled Conjugate Gradient (SCG) algorithm [27] is based upon a class of optimization techniques, well known in numerical analysis as the Conjugate Gradient Methods (CGM). Conjugate direction Methods, upon which CGM are based, have the following properties [10]

(1) Solve quadratics of N variables in N steps;
(2) The usual implementation, the CGM, requires no Hessian matrix evaluations;
(3) No matrix inversion and no storage of $N \times N$ matrix required.

The CGMs typically perform better than the method of steepest descent, but not as well as Newton's method. But with Neural networks, CGMs are generally considered to be better suited as they are the class of methods most suited for large-scale problems.

As with steepest descent and Newton's method, the crucial factor in the efficiency of an iterative gradient-based method is the direction of search at each iteration. The direction chosen for search in the case of CGMs is the so-called Q-conjugate direction, where Q is the Hessian of the error function. Once the direction is selected, the step size is decided by the line-search in that direction.

The SCG builds upon the general CGMs to create a variant of CGMs that avoids the line-search per learning iteration by introducing a scaling factor to scale the step size. This leads to the resulting SCG algorithm that requires only $O(N)$ memory usage, where N is the number of weights in the network. SCG thus yields a speed-up when compared with standard CGMs and steepest descent. SCG also is fully automated including no user dependent parameters.

Let $E(w)$ be the error function to be minimized, where $w \in \mathbb{R}^N$ is the vector of Neural Network weights. $E(w)$ at a given point $(w+y)$, y being in the vicinity of w, can be expressed by the well known Taylor expansion:

$$E(w+y) \approx E(w) + E'(w)^T y + \frac{1}{2} y^T E''(w) y \tag{2.12}$$

Denote the quadratic approximation to E in a neighborhood of a point w by

$$E_{qw}(y) = E(w) + E'(w)^T y + \frac{1}{2} y^T E''(w) y \tag{2.13}$$

The standard CG algorithm with Hestenes-Stiefel formula [10], proceeds by iteratively finding the N conjugate directions $\{p_k : k = 1, .., N\}$ and updating the weight vector. The first conjugate direction is taken to be the negative gradient at w_1, the initial weight vector, so that

$$p_1 = r_1 = -E'(w_1) \tag{2.14}$$

And given p_k, and r_k, the k^{th} iteration negative gradient, the step size α_k is calculated as

$$\alpha_k = \frac{\mu_k}{\delta_k} \tag{2.15}$$

where $\mu_k = p_k^T r_k, \delta_k = p_k^T s_k$, where $s_k = E''(w_k)p_k$. The next Conjugate direction is calculated as

$$p_{k+1} = r_{k+1} + \beta_k p_k \tag{2.16}$$

where $\beta_k = \frac{|r_{k+1}|^2 - r_{k+1} r_k}{\mu_k}$ is given by the Hestenes-Stiefel formula. Furthermore, the same formula removes the need to calculate $E''(w_k)$ at each iteration by by estimating s_k as follows:

$$s_k = E''(w_k)p_k \approx \frac{E'(w_k + s_k p_k) - E'(w_k)}{\sigma_k}, 0 < \sigma_k << 1 \tag{2.17}$$

The method so far described is the standard CG algorithm with Hestenes-Stiefel formula, and works only for positive definite Hessian at each iteration. In the case where this is not the case, i.e., $E''(w_k)$ is not always positive definite, the CG algorithm takes the approach of line search to find the α_k. The SCG on the other hand avoids the line search by using Levenberg-Marquardt approach [13] and introduces a scalar λ_k to regulate the indefiniteness of $E''(w_k)$:

$$s_k = \frac{E'(w_k + s_k p_k) - E'(w_k)}{\sigma_k} + \lambda_k p_k \tag{2.18}$$

And for each iteration adjusting λ_k looking at the sign of δ_k, which reveals if $E''(w_k)$ is not positive definite. If $\delta_k \leq 0$ then λ_k is raised and s_k is estimated again. Furthermore, to ensure that this artificial way of making the Hessian matrix doesn't deteriorate the quadratic approximation too much, a comparison comparison parameter is introduced:

$$\Delta_k = \frac{E(w_k) - E(w_k + \alpha_k p_k)}{E(w_k) - E_{qw}(\alpha_k p_k)} \tag{2.19}$$

Then based on the value of Δ_k, the value of λ_k is lowered or raised at each iteration. The resulting algorithm is the final SCG algorithm that we use in this work for training the neural network. The interested reader is referred to [27] for a detailed study of SCG.

2.4. PCA

Principal Component Analysis, or PCA, [6, 31, 20] (also known as *Karhunen-Loeve* transform) is a technique that is widely used in pattern recognition and machine learning for dimensionality reduction and feature extraction. There are two commonly used derivations of PCA—one that maximizes variance of data, and the one that minimizes the projection error. We develop the maximum variance formulation.

Given a set of data samples $\{x_n, n = 1, ..., N\}, x_n \in \mathbb{R}^D$, the goal of PCA is to project the data onto a space with dimensionality $M \leq D$, while maximizing the variance of the projected data. If v_1 is the first direction of projection, the variance of the data projected on v_1 is given by

$$\frac{1}{N}\sum_{n=1}^{N}\{v_1^T x_n - v_1^T \bar{x}\}^2 = v_1^T S v_1 \tag{2.20}$$

where $\bar{x} = \frac{1}{N}\sum_{n=1}^{N}(x_n)$ is the sample set mean, and S is the data covariance matrix defined by

$$S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T = v_1^T S v_1 \tag{2.21}$$

So, the optimization problem becomes that of maximizing 2.20, or equivalently

$$\max_{v_1} \; v_1^T X^T X v_1, \; v_1^T v_1 = 1 \tag{2.22}$$

which is a constrained optimization problem. Using a Lagrange multiplier λ, we get

$$\max_{v_1,\lambda} \; v_1^T X^T X v_1 + \lambda(1 - v_1^T v_1) \tag{2.23}$$

By setting the derivative of 2.23 with respect to v_1 equal to zero, we get

$$2Xv_1 - 2\lambda v_1 = 0 \tag{2.24}$$

$$X^T X v_1 = \lambda v_1 \tag{2.25}$$

which means that v_1 must be an eigenvector of $X^T X$ with eigenvalue λ, which also turns out to be a measure of the variance. So, v_1 turns out to be a direction of projection that results in maximum variance in the projected data, and is called the first 'Principal Component'. The subsequent directions can be found inductively as follows. Given that we have up to

v_k Principal Components, the next Principal Component v can be found by solving the following constrained optimization problem :

$$\max_{v, v \perp v_1, .., v_k} v^T X^T X v, \ v^T v = 1 \tag{2.26}$$

And this problem can be solved by creating the Lagrangian. Solving Equation 2.26 for all eigenvectors is equivalent to computing the singular value decomposition (SVD) of X [31, 20, 14]:

$$X = U \Sigma V^T \tag{2.27}$$

where $U \in \mathbb{R}^{N \times N}$, $V \in \mathbb{R}^{D \times D}$ and $\Sigma \in \mathbb{R}^{N \times D}$. In this decomposition, the columns of V, the right singular vectors, are the eigenvectors of $X^T X$ [31, 20, 14] which provide us the required 'Principal Components'. Additionally, these column vectors are ordered by the variance they produce when data is projected onto them so that the first column of V is the first Principal Component, the second one the second and so on. In this work, we thus use SVD to derive the Principal Components for our experiments.

CHAPTER 3

Data Acquisition and Representation

This chapter describes the datasets used in this study, as also the methods and EEG recording equipment used. As data representation is an important part of any signal processing method, we also describe the data representation used in this study. A novel way of using channel-subtrials is also defined and explained.

3.1. Datasets

There are four subjects in the study. The EEG data for subjects 1 and 2 was recorded by BCI laboratory at Computer Science department at Colorado State University [2]. The data were recorded using the g.Tec g.GAMMAsys system [4] with a 8-electrode cap with electrodes located at Fz, Cz, Pz, Oz, P3, P4, O1, O2. This subset of electrodes has been found to be meaningful for P300 classification [33, 30]. Such small subsets of electrodes are also easier to use for a practical and easy-to-use BCI. Subject 2 was an able-bodied participant in the study, and subject 1 was a subject with C4 complete Spinal cord injury. C4 is a level of Cervical (neck) injury that results in significant loss of function at the biceps and shoulders. While the data for subject 2 was collected in the laboratory, the data for subject 1 was recorded at home.

Three sessions of data collection were performed with both subjects 1 and 2. The subjects count one of three target letters ('b', 'd', 'p') during a session as various other non-target letters are randomly flashed on a screen. This data collection is done as per 'odd-ball' paradigm and the probability of occurrence for the target letter in each session was 0.25. The data were sampled at 256 Hz, with an inter-stimulus-interval (ISI) of 1 second. One session consisted in recording 20 target and 60 non-target subtrials of 1000ms each of EEG data at 8 electrodes.

Data for subjects 3 and 4 is taken from BCI competition III (dataset II) [1]. These experiments being based on the P300 speller as proposed by Farwell and Donchin [23], the subjects were presented with a 6 by 6 matrix of characters. The subject's task was to focus attention on characters in a word that was prescribed by the investigator (i.e., one character at a time). All rows and columns are successively and randomly intensified at a rate of 5.7Hz. The objective in this contest was to predict the correct character in each of the provided character selection 'epochs' that consisted of 15 sequences—each sequence consisting of 12 flashings—6 rows and 6 columns. The data was collected and bandpass filtered from 0.1-60Hz and digitized at 240Hz. After intensification of a row/column, the matrix was blank for 75ms. Row/column intensifications were block randomized in blocks of 12. The sets of 12 intensifications were repeated 15 times for each character epoch (i.e., any specific row/column was intensified 15 times and thus there were 180 total intensifications for each character epoch). Each character epoch was followed by a 2.5 s period, and during this time the matrix was blank. While data for subjects 3 and 4 is recorded at 64 electrode locations, this work uses data only from 8 locations which is the same set of electrodes used for subjects 1 and 2—Fz, Cz, Pz, Oz, P3, P4, O1, O2.

3.2. Data Processing and Representation

The original data for all the four subjects comes as a continuous EEG for an entire recording. This data for subject 1 is shown in Figure 3.1. The data from each channel is separated on the vertical axis and the topmost dummy channel in the shape of 'box function' captures the start of stimulus, the positive 'box' signifying the target letter, and the negative 'box' signifying the non-target letter.

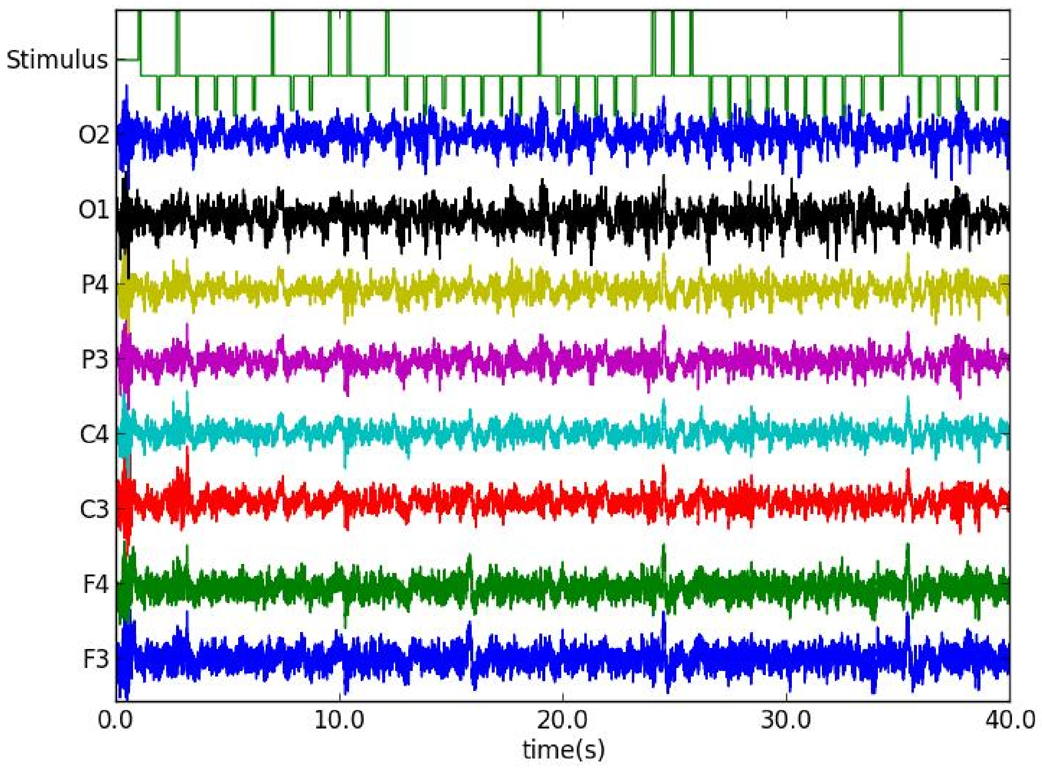

FIGURE 3.1. Original Data sub1. The top channel 'Stimulus' captures the stimulus being a target or a non-target letter.

All the data was then bandpass-filtered from 0.23 Hz to 30 Hz. The data were then normalized for zero mean and unit variance. Figures 3.2 and 3.3 show a 4 second window of data before and after the bandpass filtering. The bandpass filtering was done using Butterworth bandpass filter [3]. This data is then sliced using the 'box function' dummy channel to separate the target and non-target subtrials for each channel. Each Dataset was thus reshaped into a matrix with each row representing a channel-subtrial- 256 datapoints as a time series for subjects 1 (sub1) and 2 (sub2), and 240 for subjects 3(sub3) and 4(sub4).

The channel-subtrials used for classification therefore consist of one-second long windows after each stimulus onset that are extracted from the continuous signal in each data segment.

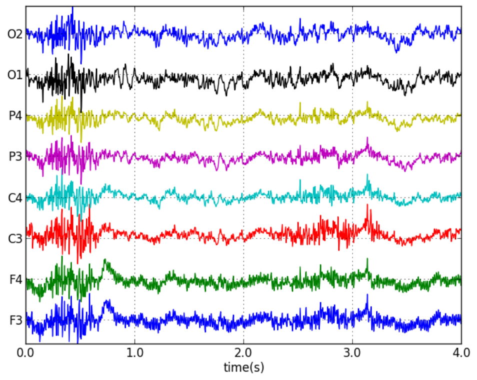

FIGURE 3.2. 4s Original Data sub1.

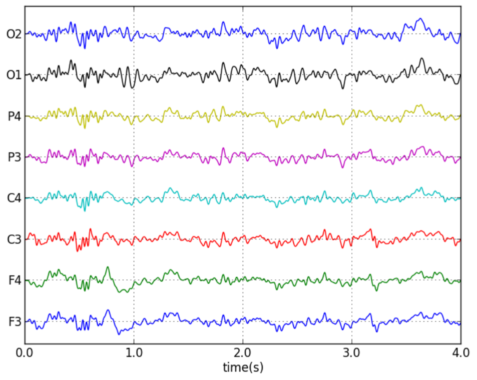

FIGURE 3.3. 4s Data sub1 after bandpassing.

Data representation is an important part of any signal processing method. There are different ways to represent data, which can broadly be classified into spatio-temporal domain and frequency domain representations. For this work, we stick with the more standard approach [23, 21] of representing the data in the spatio-temporal domain. A common approach used in data representation is to concatenate all the channel specific EEG for a particular subtrial [21]. Here we have taken a different approach which does not seem to have been explored in literature. This approach involves considering a time series from each of the eight channels as a separate subtrial for the purpose of initial classification. What this means is that a subtrial that in the usual sense consists of eight different time series, one for each channel, is split into eight different subtrials in our data matrix. This is done with the view that P300 response is present in each of the eight channels considered although it varies in degree and has some phase difference, and that our preprocessing method should be able to get common sources to these responses at different channels. We call these channel subtrials. Once the training of the algorithm is done, and results for the test set are collected for each channel sub trial, these results are aggregated to arrive at a classification for the 'overall' subtrial. This aggregation is done using a voting method.

The P300 data is generally unbalanced in positive and negative examples, as every trial contains many more negative examples than the positive ones in accordance with the 'odd-ball' paradigm. This usually tends to bias a classifier in favor of the negative examples. So, an equal number of subtrials of target and non-target EEG were used for these experiments (20 for sub1 and sub2, and 30 for sub3 and sub4).

As a next step, we visualize the subtrials and can clearly see that as more subtrials are averaged, a P300 amplitude starts appearing in the averaged data for subjects 1 and 2. This can be seen in Figures 3.4, 3.5, 3.6, and 3.7 for subject 1 and Figures 3.8, 3.9 for subject 2. The noise in the signal is canceled when more trials are averaged. The same pattern of averaging resulting in noise cancellation is observed for subjects 3 and 4 also as shown in Figures 3.10 through 3.13. But a clear P300 is not clearly observable for these subjects. The reason for this seems to be the different data acquisition method used for these two subjects. While sub1 and sub2 data was collected with an ISI (Inter Stimulus Interval, which is the time interval between consecutive stimuli) of 1000ms, data for sub3 and sub4 was collected with an ISI of just 175ms. The short ISI results in the overlapping of signals following a stimulus which in turn suppresses the P300 amplitude.

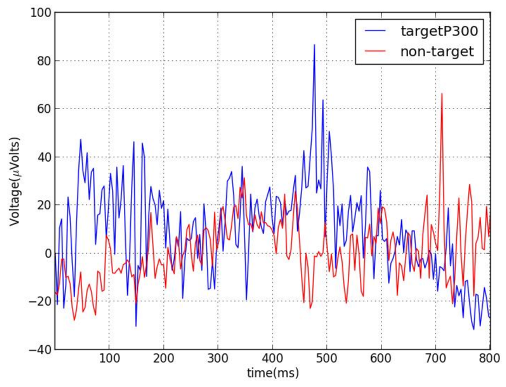

FIGURE 3.4. Two random sub1 single subtrials

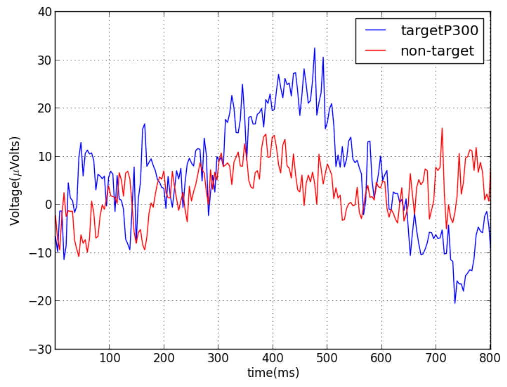

FIGURE 3.5. sub1 5 averaged subtrials

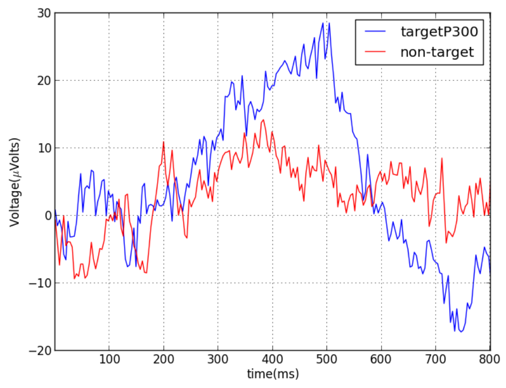

FIGURE 3.6. sub1 10 averaged subtrials

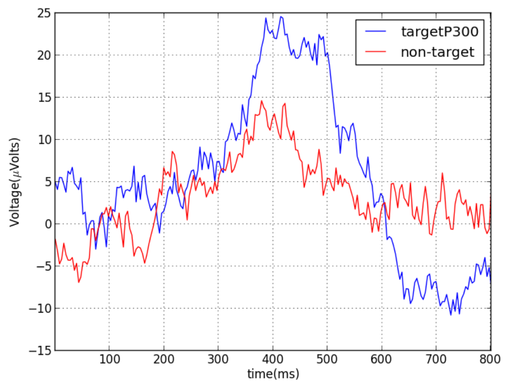

FIGURE 3.7. sub1 20 averaged subtrials

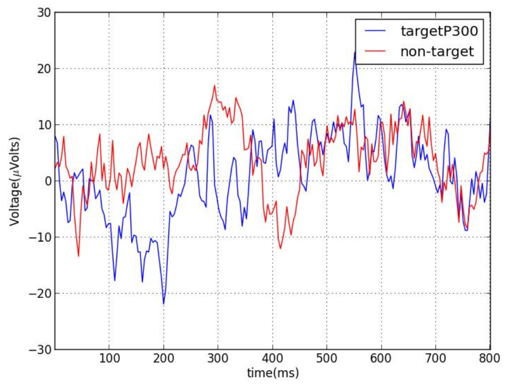

FIGURE 3.8. Two random sub2 single subtrials

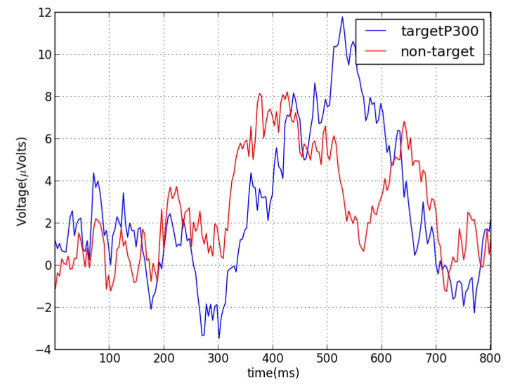

FIGURE 3.9. sub2 20 averaged subtrials

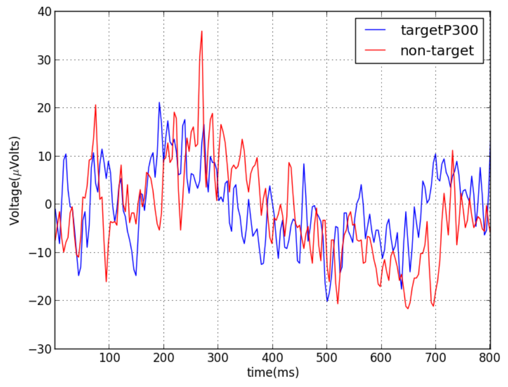

FIGURE 3.10. Two random sub3 single subtrials

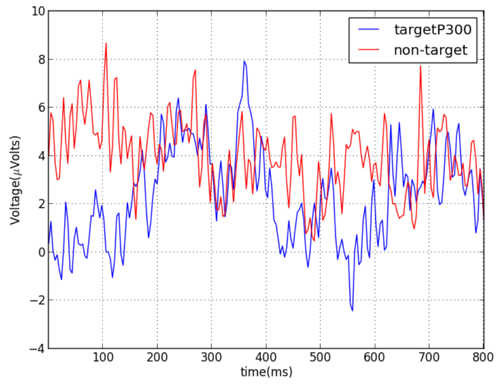

FIGURE 3.11. sub3 20 averaged subtrials

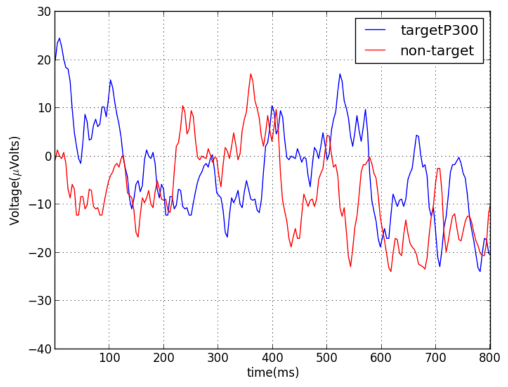

FIGURE 3.12. Two random sub4 single subtrials

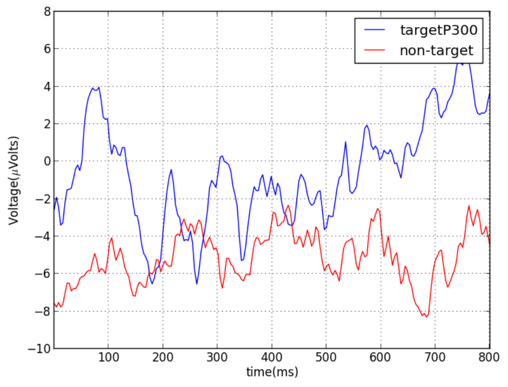

FIGURE 3.13. sub4 20 averaged subtrials

CHAPTER 4

Experiments and Results

In this chapter, we provide details of all the experiments that were performed to compare the performance of different methods used. The detailed workflows of these experiments are explained. The results from the experiments are provided and discussed.

4.1. Experiment Design

Experiments were designed to test and compare the various approaches selected for this work with and without PCA. So, the following workflow was followed.

(1) Classification performance of the four classifiers—LDA, QDA, LR and NLR—was recorded on the raw data, i.e., the data that has not been transformed with PCA. In this, no feature selection is done, so that all the original features are used.
(2) Then classification performance is tested after performing PCA on the data. There are two procedures in this set of experiments—one with forward selection of principal components across all the components, and one where forward selection is done on selected number of top components as per the magnitude of the singular values.

4.2. Without PCA

We start experiments by testing our four methods on the raw data, i.e., the data without PCA transformation for the four subjects. This would serve as a benchmark for the experiments to follow using the PCA. For these experiments and that follow, the following common approach was followed.

(1) The datasets were randomly partitioned into training and test sets in the ratio of 80:20. As we consider channel-subtrials as separate subtrials during training and

classification before voting, the data was partitioned to ensure we place complete set of 8 channel-subtrials belonging to a subtrial together either in the training or test sets.

(2) The algorithms are trained on the data comprising the channel-subtrials, and tested on the test channel-subtrials.

(3) The accuracies on the channel-subtrials from the 8 channels are aggregated together using the voting method to decide on the final class of the actual subtrial.

(4) The above process is repeated 20 times and the accuracies over the 20 runs are averaged to obtain the final accuracies.

For NLR, the training set was partitioned to create a validation set. The accuracies on the validation set were used to choose the number of hidden units. In the pilot studies on all four subjects, a range of number of hidden units from 2 to the number of features was tested on the validation set. It was observed that the validation accuracies generally were the best at number of hidden units equal to the number of features in the dataset. So, throughout the experiments, number of hidden units for a dataset have been taken as number of features used for classification.

The accuracies obtained on the raw data for the four subjects is shown in Table 4.1. First thing to note is that QDA did not work for any of the subjects, the problem being that the covariance matrices became singular, that gave a runtime error. So, the accuracies for QDA have been left blank for raw data. The problem of sample covariance matrices being singular occurs when sample size is less than the number of features, which happens to be the case when using QDA on raw data. The same problem was not usually encountered in the case of LDA as the averaging of the sample covariance matrices of the two classes resulted in the average covariance matrix being non-singular. In the subsequent experiments, a small subset of features was always used, which ensured that this problem is not encountered.

Analyzing the results, we observe as generally expected in P300 classification that no single method is the best across all 4 subjects. LR works the best for subjects sub1 and sub4, but NLR also works the best for two subjects sub3 and sub4. It's only for sub2 that neither of the NN versions, linear and nonlinear, work well. And it is LDA that works the best for sub2. So, across subjects, we can not generalize if linear or nonlinear methods are a clear winner. But for sub1, the performance of the linear version of NN, the LR, is far superior than the other methods. The best accuracies obtained for all the subjects are near 50% which is the expected accuracy for a random classifier for single trials [16, 9] except sub1 which gets a very high accuracy for single trials.

TABLE 4.1. Accuracies with all four methods (without PCA)

Subjects $\longrightarrow$	sub1	sub2	sub3	sub4
Algorithms $\downarrow$				
LDA	47.33	**51.25**	49.58	44.58
QDA	-	-	-	-
LR	**65.62**	48.25	47.08	**51.25**
NLR	58.75	42.12	**50.83**	**51.25**

4.3. With PCA

The next step in our experiments was to use PCA for feature extraction. We start our experiments with PCA by getting the Principal Components (PCs) of the training data using SVD. Then we plot the projection of training data onto first 10 components for each of the four datasets. The projections of sub1 dataset are shown in the Figures 4.1 through 4.4.

This is done to visually see how each of the individual PC does for discriminating the target and non-target class data.

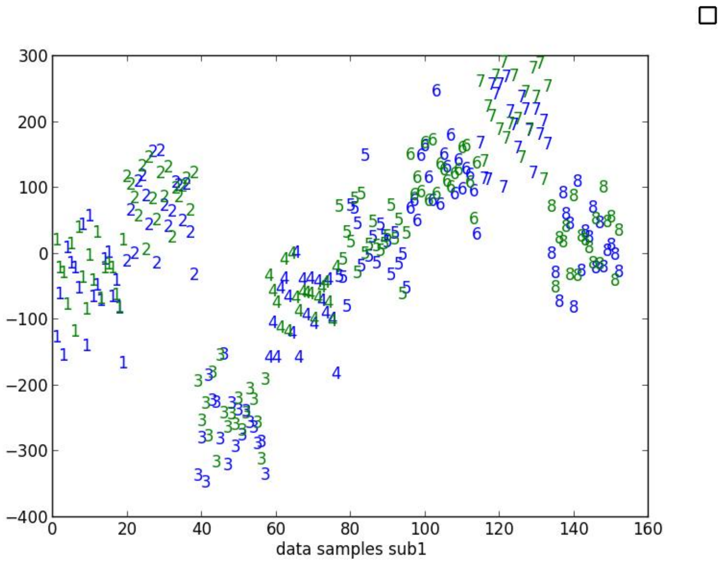

FIGURE 4.1. sub1 data projected on 1st Principal Component. Each digit represents data from one of the 8 channels. Blue digits represent the target and the green no-target data.

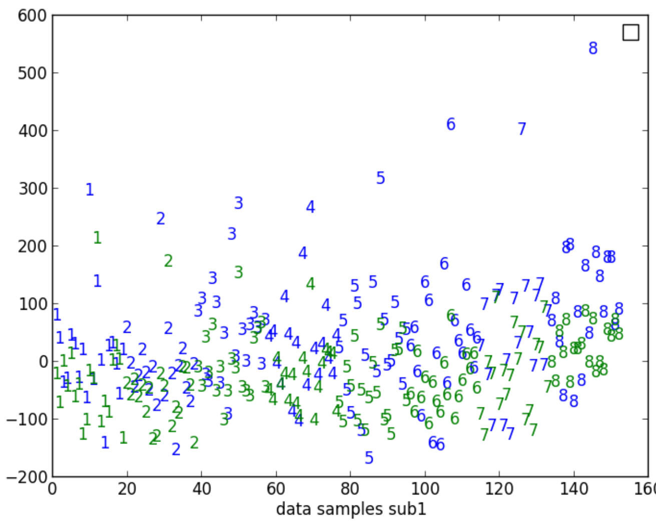

FIGURE 4.2. sub1 data projected on 2nd Principal Component. Each digit represents data from one of the 8 channels. Blue digits represent the target and the green no-target data.

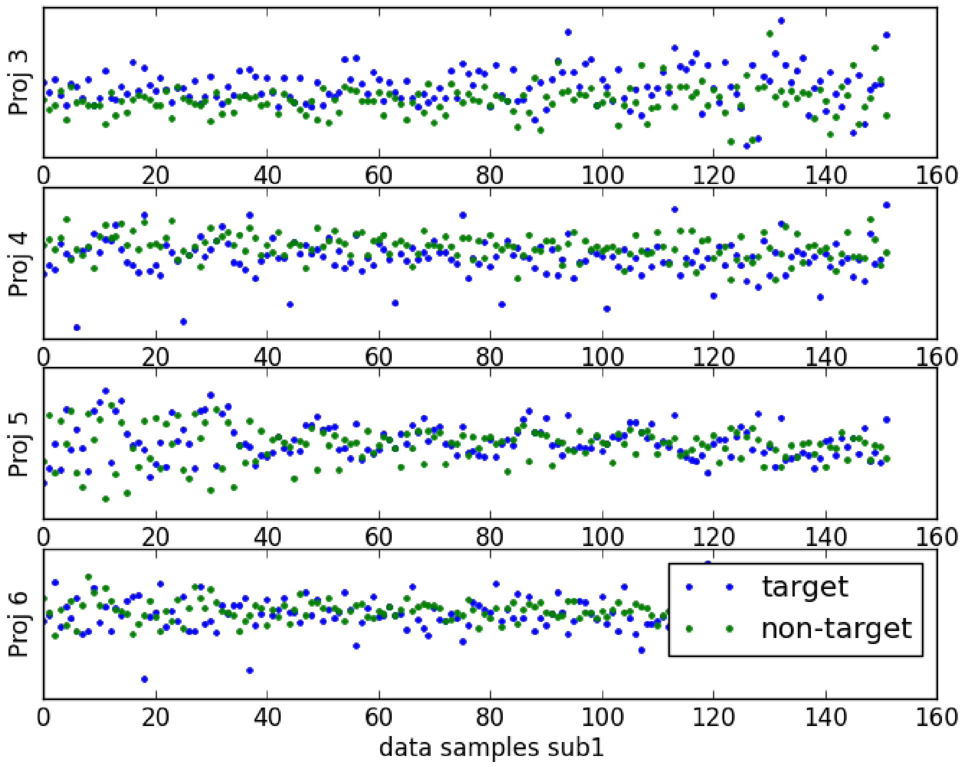

FIGURE 4.3. sub1 data projected on Principal Components 3,4,5,6

As we can see, the first PC captures the variance among the 8 channels as all the same-channel data are clumped together in the direction of this PC. This does not help in discriminating the two classes. On the other hand, the second PC seems to do a good job of separating the two classes. Similar analysis of the remaining top 8 PCs shows some PCs that do better than others in separating the two classes. Projections 3 and 4 also seem to work well. After this analysis which was done for the top 20 PCs, we choose the top 3 PCs through visual inspection—for sub1 these turned out to PCs 2,3, and 4. Projection of sub1 data onto a 3-dimensional subspace of these 3 PCs is shown in Figures 4.5 and 4.6.

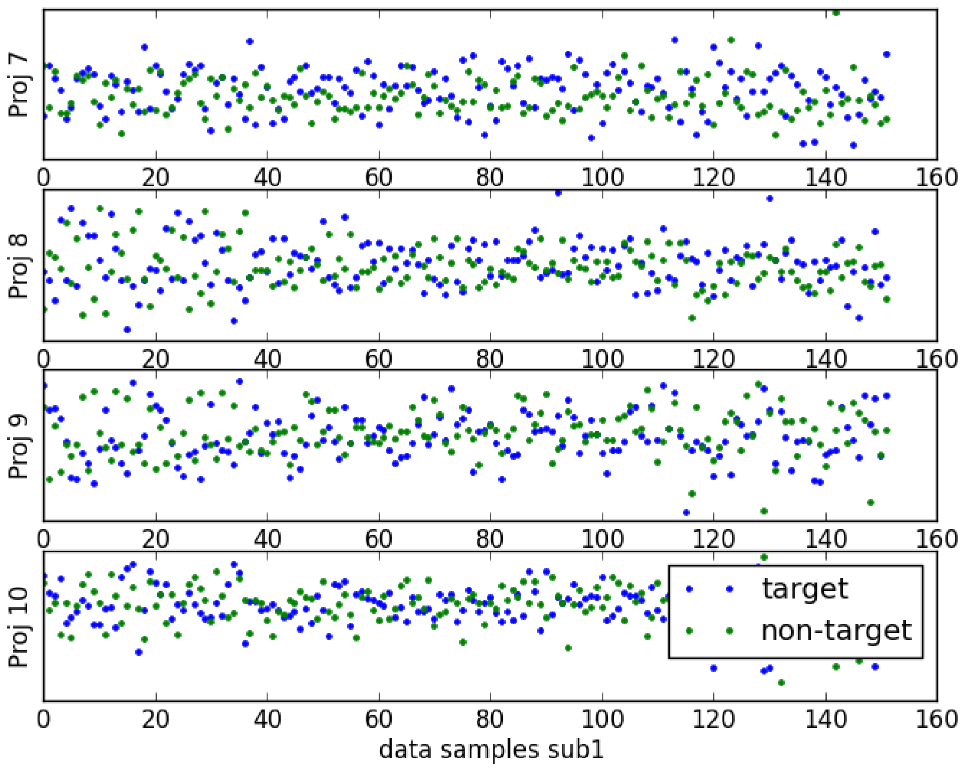

FIGURE 4.4. sub1 data projected on Principal Components 7,8,9,10

After a similar analysis for subjects sub2, sub3 and sub4, we similarly choose the top 3 components for each of them. Then using the chosen three components, we run our classification experiments. The results of these experiments are shown in Table 4.2.

TABLE 4.2. Accuracies for PCA-transformed data using only 3 visually chosen PCs

Subjects $\longrightarrow$ Algorithms $\downarrow$	sub1	sub2	sub3	sub4
PCA+LDA	**55.0**	55.0	**49.16**	**55.00**
PCA+QDA	52.5	46.0	**49.16**	51.6
PCA+LR	41.6	40.8	45.0	50.83
PCA+NLR	41.6	**56.6**	**49.16**	53.33

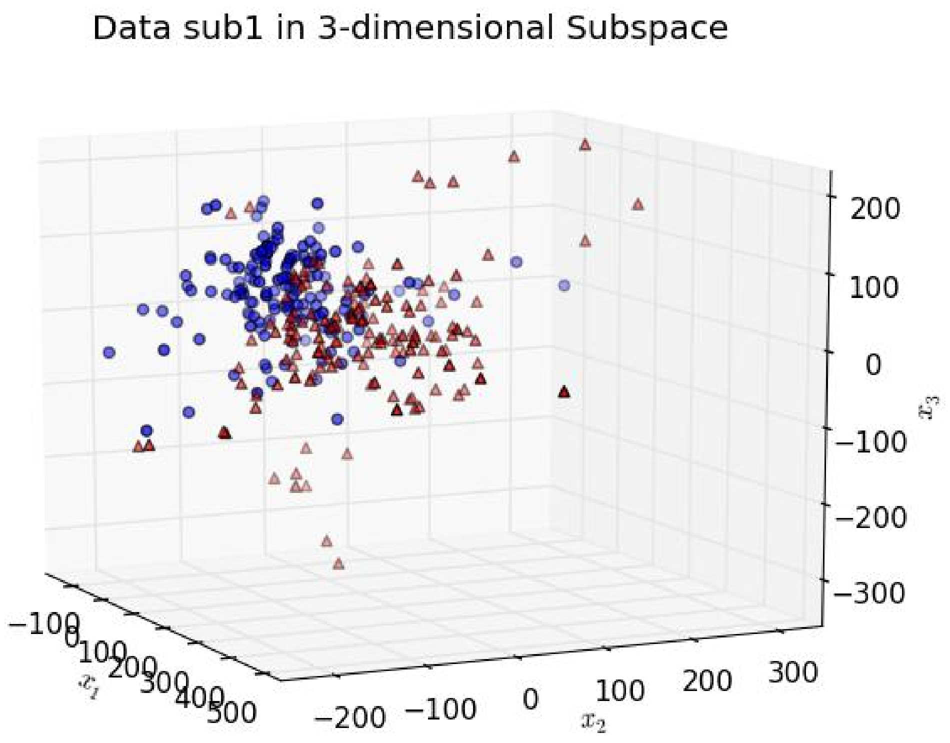

FIGURE 4.5. sub1 data projected on 3-D subspace - view 1

While this method improved the best accuracy for sub2 and sub4, for sub1 and sub3 the accuracies decreased with almost all the methods. Although the accuracy gains for sub2 and sub4 are impressive, this method is neither scientific nor desirable for a BCI because of its reliance on human intervention and visual judgment. Moreover, there is a limit to the number of PCs that we can inspect visually. But, it does show that if we are able to pick the right components, we could get accuracy gains for some subjects.

To make the process of choosing the PCs thorough and to automate the process, we use the method of 'forward selection' (FS) of the PCs in each of the four algorithms. The forward

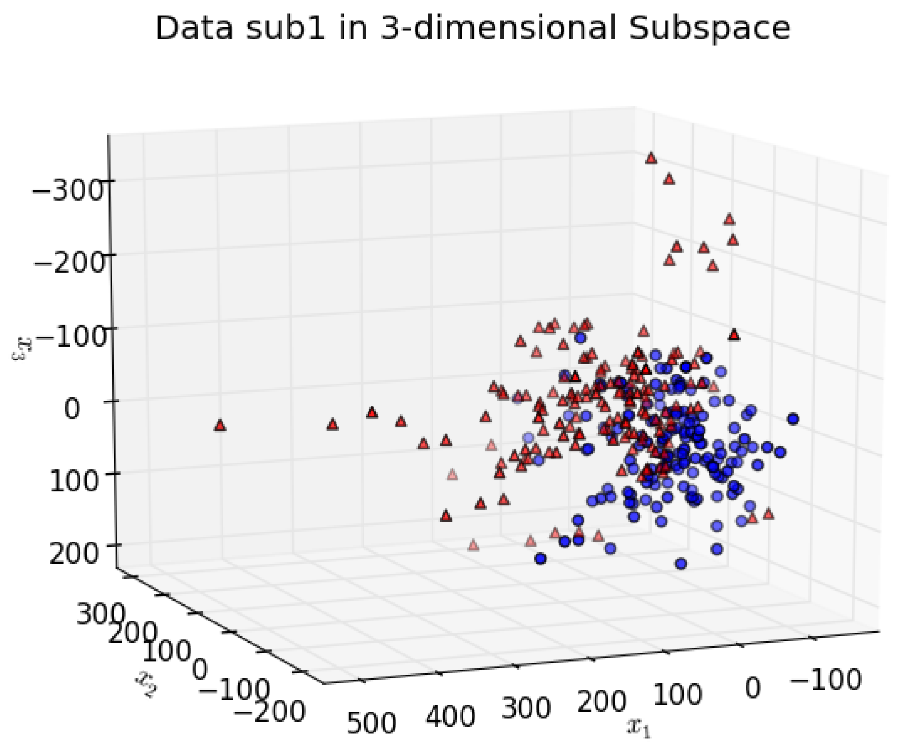

FIGURE 4.6. sub1 data projected on 3-D subspace - view 2

selection is done using 3-fold cross-validation to choose the PCs that give the best accuracy for the validation set. A new PC is thus added to the set of PCs at each iteration. Finally, the minimal set that gives the best accuracy on the validation set is chosen. As this method is very expensive, we pick and test validation accuracies till only the top 50 components are chosen. This also ensures that data sample size being less than the dimensionality doesn't give singular covariance matrices in the case of QDA. This method thus iterates through all the PCs at each iteration and picks the best PC at each iteration. The results of this method are shown in table 4.3.

TABLE 4.3. Accuracies for PCA-transformed data with number of PCs chosen using forward selection(Number of Principal Components in braces)

Subjects ⟶ Algorithms ↓	sub1	sub2	sub3	sub4
PCA+LDA	54.37(11)	53.12(14)	49.58(16)	48.75(14)
PCA+QDA	49.37(18)	**55.12**(18)	50.83(15)	47.91(12)
PCA+LR	42.50(6)	50.46(5)	47.08(3)	50.00(5)
PCA+NLR	**55.00**(4)	48.03(5)	**51.25**(20)	**50.83**(5)

The results using forward selection of Principal components are not found to be as encouraging as expected. The challenge lies in selecting the number of top components based on the validation accuracies. The method chooses the number of components that give the best validation accuracy, but this doesn't necessarily work for the test data. The Figures 4.7, 4.8 below show validation and test accuracies for a single run of this method with LDA and NLR respectively as classifiers.

As can be seen in these figures, the validation accuracies do not generalize well to the test data. The validation process chooses the optimal number of components as the number that gives the best accuracy across all components, but that same number of components do not provide the best accuracy for the test set—in fact as can been seen in the figures, the two are way off in both experiments. In the case of LDA, we get the best validation accuracy for 10 components, whereas for the test set the accuracy at 10 components is way less than say for 12 components or for 20 components. Similarly, for NLR, the best validation accuracy occurs for 12 components, whereas the test accuracy at 12 components is way lower than for, say, 3 and 9 components. The reason for that is the extreme noise in the data due to which the validation set accuracies don't generalize well to test data. The FS would pick such components that might be modeling the noise of the validation set which obviously will

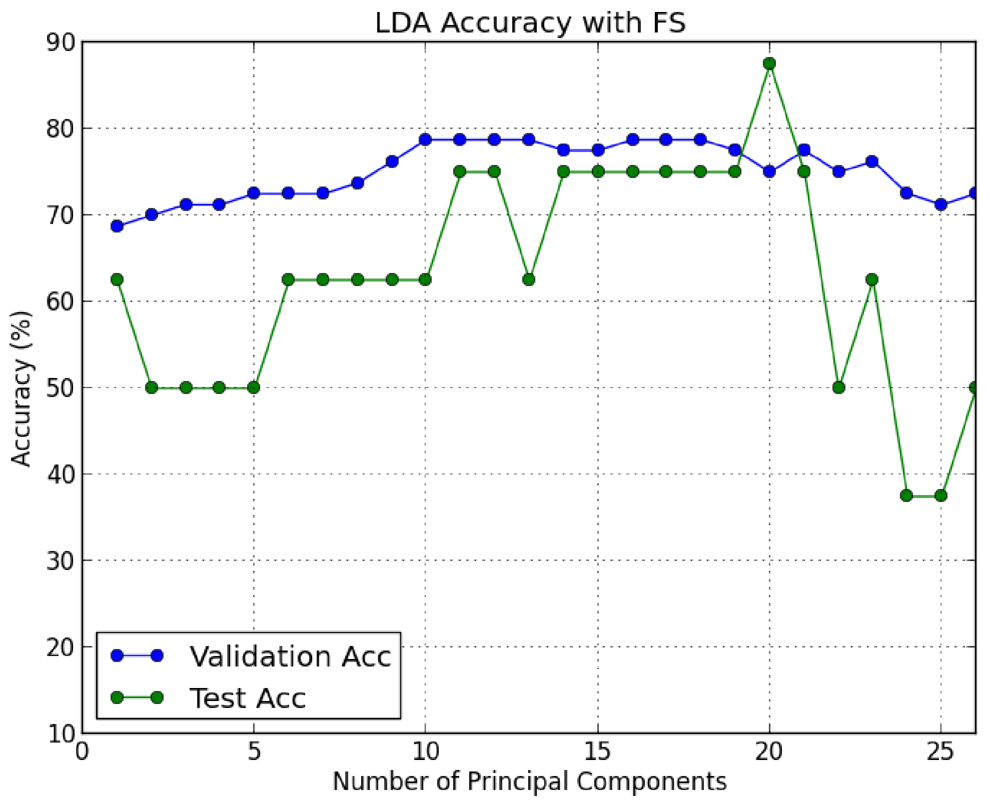

FIGURE 4.7. LDA accuracies for sub1

throw the classifier off on the test data. Then there is also over-fitting of training data in the case of non-linear methods as can be seen for NLR. In addition to these problems, there is the matter of complexity and runtime. The FS is a very expensive method as each iteration of choosing a new component has to iterate through all the remaining set of components.

Considering the above pitfalls of the FS method, we need to try a different method for selecting a good subset of PCs. Looking back at the results of using top 3 PCs chosen visually from top 20 components, we try modifying the forward selection algorithm to select the components only from a set of an empirically chosen number, say n, of top PCs based on the magnitude of their singular values. This amounts to just considering the top n PCs as they are ordered already by magnitude of their singular values. As visually chosen 3

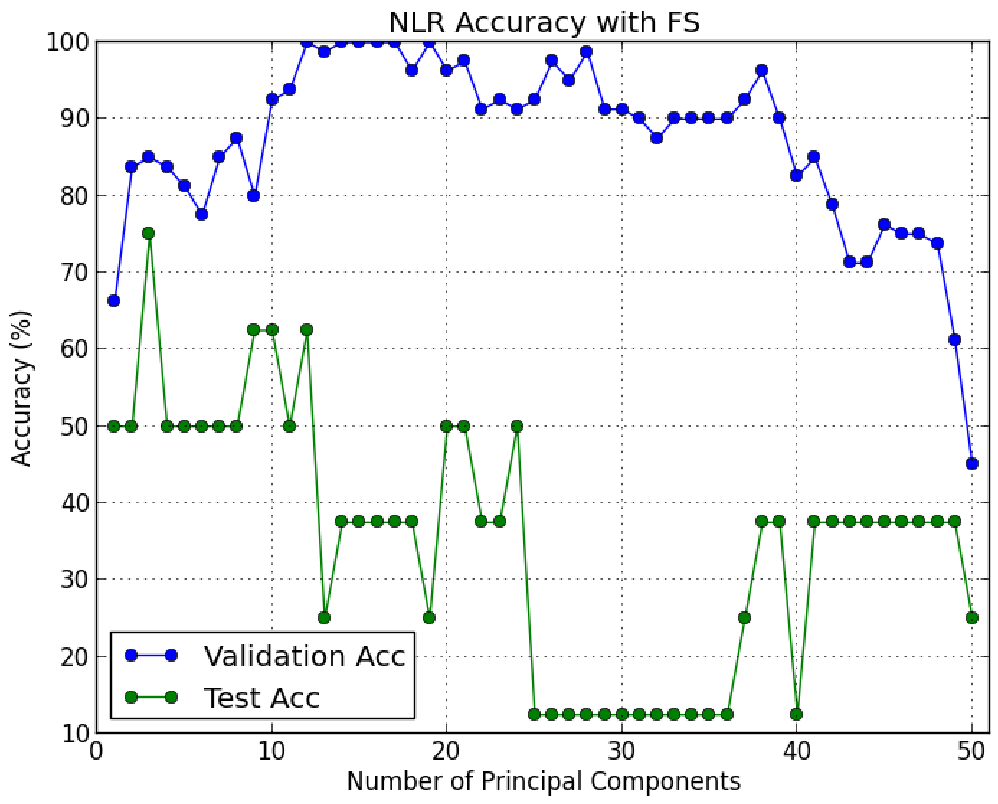

FIGURE 4.8. NLR accuracies for sub1

components had shown good results for 2 out of 4 subjects, we try $n = \{5, 10, 15, 20\}$. After looking at the results thus obtained, it was observed that using $n = 5$ works the best. The results of using FS on a restricted set of top 5 PCs is shown in table 4.4.

TABLE 4.4. Accuracies for PCA-transformed data choosing the best among only top 5 PCs (Number of Principal Components in braces)

Subjects $\longrightarrow$	sub1	sub2	sub3	sub4
Algorithms $\downarrow$				
PCA+LDA	55.0(2)	48.0(3)	45.0(2)	47.5(3)
PCA+QDA	55.0(2)	52.7(3)	40.83(2)	**53.3**(1)
PCA+LR	41.66(3)	**57.5**(2)	56.9(4)	46.66(2)
PCA+NLR	**60.0(3)**	40.0(3)	**63.05**(4)	50.83(4)

The results obtained using FS on the restricted set of top 5 PCs show an improvement in accuracy for all the four subjects when compared with regular FS. The method gives overall best accuracies for sub2 and sub3. For sub4 also, the accuracy is better than all methods except visual selection of PCs which can be discounted as that's not a practical method for an online BCI. So, discounting the visual selection method, we get best accuracies for three subjects out of four using this method of restricted FS. In addition, this method comes with greatly reduced data dimensionality. While FS selected as high as 18 PCs, this method needed no more than 4 components to achieve much better results.

CHAPTER 5

Conclusions

5.1. Summary of Results

As the experiments have shown and as is supported by the literature, the success of a method for P300 classification depends a lot on the subjects. While LR gave exceedingly good results for sub1 without PCA, sub3 had the best accuracy of all the methods with NLR + PCA (with restricted FS). NLR + PCA (with restricted FS) also gave the second best accuracy for sub1 at 60%.

For sub2, on the other hand, it was the linear version of NN—the LR (with restricted FS)—that gave the best accuracy. For sub4 alone, the visually chosen 3 PCs gave the best accuracy. But overall it's clear that PCA did help in improving the accuracy of classification of single trials in all subjects but sub1. The best results except in the case of sub1 all came with a much reduced dimensionality of data—the dimensionality was a maximum of 4 for all these subjects. So, PCA feature selection not only increased the classification accuracy but also reduced the execution time of the algorithms by the resulting dimensionality reduction.

Among the wider distinction between linear and nonlinear methods, the results were split. While sub1 and sub2 got the best accuracies using linear LR, sub3 and sub4 got the best results with QDA and NLR—both nonlinear methods. The reason for this seems to be different data acquisition methods used for the two datasets. While sub1 and sub2 data was collected with an ISI of 1000ms, data for sub3 and sub4 was collected with an ISI of just 175ms. The short ISI resulted in the overlapping of P300 amplitude of the EEG which happens about 300ms with the new stimulus induced signal, which also explains why the grand average signal for sub3 and sub4 was not as similar to a P300 as it was for sub1 and sub2. The overlapping of signals from many stimuli seems like the reason that makes these two datasets more difficult to be linearly separable. Another observation is that the method of treating each channel-subtrial as a separate subtrial for the purpose of classification works well for P300. The hypothesis of PCA being able to extract the relevant source components from this channel-subtrial dataset is also validated by the results. Also for sub1 and sub2, the PCA captures the variance across channels into a component (it was the first PC for sub1). Then the classifier would ignore that component as it won't give it any significant discriminatory value for the purpose of classification. Such variance was not that pronounced in the case of sub3 and sub4.

5.2. Future Work

This work has many avenues for future work. As discussed earlier, as the P300 classification depends a lot on the subjects, it would be useful to run these same experiments on data from more subjects to see how well the results generalize across subjects. Then for the same subjects other classification methods especially variants of LDA, such as Fisher's linear discriminant (FLD), stepwise linear discriminant analysis (SWLDA), and regularized Discriminant Analysis, can be tried to see if the results could be further improved. Another interesting thing to try would be to compare our method of channel-subtrial based data with other methods of data representation in spatio-temporal and frequency domains. Also, the methods used in this work can be studied in more depth. For example, it's well known that LDA and QDA require a certain minimum number of data samples for optimal performance. It would be interesting to see if and by how much could the performance further improve if more number of samples are used for training the classifiers used in this work. For NLR, we could also try multiple hidden layers to see how that would work for P300 classification.

BIBLIOGRAPHY

[1] Bci competition iii challenge 2004 (http://www.bbci.de/competition/iii/desc_ii.pdf).

[2] Brain-computer interfaces laboratory, computer science department, colorado state university, usa (http://www.cs.colostate.edu/eeg/main/data/2011-12_bci_csu).

[3] Butterworth filter design, mathworks (http://www.mathworks.com/help/signal/ref/butter.html).

[4] g.tec g.gammasys active electrode system (http://www.gtec.at/products/electrodes-and-sensors/g.gammasys-specs-features).

[5] Kubler A, Kotchoubey B, Kaiser J, Wolpaw JR, and Birbaumer N. Brain-computer communication: unlocking the locked in. *Psychological Bulletin, American Psychological Association, vol 127, pp. 358-375*, 2001.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[7] Benjamin Blankertz. Documentation, wadsworth bci dataset (p300 evoked potentials), bci competition iii challenge (http://www.bbci.de/competition/iii/desc_ii.pdf), 2004.

[8] V. Bostanov. Bci competition 2003-data sets ib and iib: feature extraction from eventrelated brain potentials with the continuous wavelet transform and the t-value scalogram. *IEEE Trans. Biomed. Eng., vol. 51, pp. 1057-1061*, 2004.

[9] Zachary Cashero. *Comparison of EEG preprocessing methods to improve the classification of P300 trials, M.S Thesis*. Department of Computer Science, Colorado State University, USA, 2011.

[10] Edwin KP Chong and Stanislaw H Zak. *An introduction to optimization*. John Wiley & Sons, 2013.

[11] Donchin E, Spencer KM, and Wijesinghe R. The mental prosthesis: Assessing the speed

of a p300-based brain-computer interface. *Rehabilitation Engineering, IEEE Transactions on , vol.8, no.2, pp. 174-179*, 2000.

[12] Sellers EW and Donchin E. A p300-based brain-computer interface: Initial tests by als patients. *Clinical Neurophysiology 117, pp. 538-548*, 2006.

[13] R. Fletcher. *Practical Methods of Optimization*. John Wiley, Sons, 1975.

[14] Gene H. Golub and Charles F. Van Loan. *Matrix computations (4th ed.)*. Johns Hopkins University Press Baltimore, MD, USA, 2013.

[15] Mirghasemi H, Fazel-Rezai R, and Shamsollahi MB. Analysis of p300 classifiers in brain computer interface speller. *In the Proceedings of Conf Proc IEEE Eng Med Biol Soc. 2006;1:6205-8*, 2006.

[16] Serby H, Yom-Tov E, and Inbar GF. An improved p300-based brain-computer interface. *IEEE Trans Neural Syst Rehabil Eng, vol. 13, pp. 89-98*, 2005.

[17] Trevor. Hastie, Robert. Tibshirani, and J Jerome H Friedman. *The Elements of Statistical Learning*. Springer New York, 2001.

[18] U. Hoffmann, J. Vesin, and T Ebrahimi. Recent advances in brain-computer interfaces. *Multimedia Signal Processing, 2007. MMSP 2007. IEEE 9th Workshop on Recent advances in brain-computer interfaces, pp. 17-17*, 2007.

[19] M. Kaper, P. Meinicke, U. Grossekathoefer, T. Lingner, , and H. Ritter. Bci competition 2003- data set iib: support vector machines for the p300 speller paradigm. *IEEE Transactions on Biomedical Engineering, vol. 51, pp. 1073-1076*, 2004.

[20] Michael Kirby. *Geometric Data Analysis: An Empirical Approach to Dimensionality Reduction and the Study of Patterns*. John Wiley & Sons, 2000.

[21] D. Krusienski, E. Sellers, F. Cabestaing, S. Bayoudh, D. McFarland, T. Vaughan, and J. Wolpaw. A comparison of classification techniques for the p300 speller. *Journal of Neural Engineering, vol. 3, no. 4, pp. 299-305*, 2006.

[22] D.J. Krusienski, E.W. Sellers, D.J. McFarland, T.M. Vaughan, and J.R. Wolpaw. Toward enhanced p300 speller performance. *Journal of Neuroscience Methods, vol. 167, pp. 15-21*, 2008.

[23] Farwell LA and Donchin E. Talking off the top of your head: Toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology, vol. 70, Issue 6, pp 510-523*, 1988.

[24] Kun Li, Vanitha Narayan Raju, Ravi Sankar, Yael Arbel, and Emanuel Donchin. Advances and challenges in signal analysis for single trial p300-bci. *Springer-Verlag, Berlin, pp. 88-94*, 2011.

[25] Steven J. Luck. An introduction to the event-related potential technique. *MIT Press, Cambridge, USA*, 2005.

[26] J. N. Mak, Y. Arbel, J. W. Minett, L. M. McCane, B. Yuksel, D. Ryan, D. Thompson, L. Bianchi, and D. Erdogmus. Optimizing the p300-based brain-computer interface: current status, limitations and future directions. *Journal of Neural Engineering, vol. 8, no. 2*, 2011.

[27] Martin F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks, vol. 6, pp. 525-533*, 1993.

[28] Ernst Niedermeyer and Fernando Lopes da Silva. *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields.* Lippincott Williams and Wilkins, 2004.

[29] F. Piccione, F. Giorgi, P. Tonin, K. Priftis, S. Giove, S. Silvoni, G. Palmas, and F. Beverina. P300-based brain computer interface: Reliability and performance in healthy and paralysed participants. *Clinical Neurophysiology 117, pp. 531-537*, 2006.

[30] E.W. Sellers, D.J. Krusienski, D.J. McFarland, and J.R. Wolpaw. Towards brain-computer interfacing. *Cambridge, MA: The MIT Press, pp. 31-42*, 2007.

[31] Gilbert Strang. *Introduction to Linear Algebra*, volume 1. Wellesley-Cambridge Press,

2009.

[32] Vaughan TM, Heetderks WJ, Trejo LJ, Rymer WZ, Weinrich M, Moore MM, Kubler A, Dobkin BH, Birbaumer N, Donchin E, Wolpaw EW, and Wolpaw JR. Brain-computer interface technology: A review of the second international meeting. *IEEE Transactions on Rehabilitation Engineering, vol. 11(2), pp. 94-109*, 2003.

[33] J. R. Wolpaw and D. J. McFarland. Control of a two-dimensional movement signal by a non-invasive brain-computer interface in humans. *Proceedings of the National Academy of Sciences of the United States of America, vol. 101(51), pp. 17849-17854*, 2004.

[34] Jonathan R. Wolpaw, Niels Birbaumer, William J. Heetderks, Dennis J. McFarland, P. Hunter Peckham, Gerwin Schalk, Emanuel Donchin, Louis A. Quatrano, Charles J. Robinson, , and Theresa M. Vaughan. Brain-computer interface technology: A review of the first international meeting. *IEEE Transactions on Rehabilitation Engineering, vol. 8(2), pp. 164-73*, 2000.

[35] Jonathan R. Wolpaw, Niels Birbaumer, Dennis J. McFarland, Gert Pfurtscheller, and Theresa M. Vaughan. Braincomputer interfaces for communication and control. *Neural Networks, vol. 6(4), pp. 525–533*, 1993.

[36] N. Xu, X. Gao, B. Hong, X. Miao, S. Gao, , and F. Yang. Bci competition 2003-data set iib: Enhancing p300 wave detection using ica-based subspace projections for bci applications. *IEEE Transactions on Biomedical Engineering, vol. 3, pp. 1067-1072*, 2004.

[37] Guoqiang Peter Zhang. Neural networks for classification: a survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol.30, no.4, pp. 451-462*, 2000.