FEATURE SELECTION & EXTRACTION ALGORITHMS FOR BRAIN COMPUTER INTERFACE

By

Anas Abdulqader Mohammed Hadi

A thesis submitted for the requirements of the degree

of Master of Computer Science

Faculty of Computing and Information Technology

King AbdulAziz University - Jeddah

FEATURE SELECTION & EXTRACTION ALGORITHMS FOR BRAIN COMPUTER INTERFACE

By

Anas Abdulqader Mohammed Hadi

A thesis submitted for the requirements of the degree

of Master of Computer Science

Supervised By

Dr. Mahmoud Ibrahim Kamel Ali

Faculty of Computing and Information Technology

King AbdulAziz University - Jeddah

Rabi' al-thani 1434H – February 2013


بسم الله الرحمن الرحيم

FEATURE SELECTION & EXTRACTION ALGORITHMS FOR BRAIN COMPUTER INTERFACE

By

Anas Abdulqader Mohammed Hadi

This thesis has been approved and accepted in partial fulfillment of the requirements for the degree of Master of Computer Science

EXAMINATION COMMITTEE

	Name	Rank	Field	Signature
External Examiner	**Prof. Yasser M. Kadah**	**Professor**	**Biomedical Engineering**	
Internal Examiner	**Dr. Mohammad J. Alhaddad**	**Associate Professor**	**Information Technology**	
Advisor	**Dr. Mahmoud I. Kamel**	**Assistant Professor**	**Computer Science**	

KING ABDULAZIZ UNIVERSITY

Rabi' al-thani 1434H – February 2013

Dedicated to my family:

My Mother

Safa, and Hamzah

ACKNOWLEDGMENTS

The author wishes to express sincere thanks and appreciation to Allah who give me the light to understand and give me the power to do.

Many thanks to Dr. Hussain M. Malibary and Dr. Mohammad J. Al-Haddad for sharing their invaluable experience with me before and during writing this thesis. I should not forget to thank Dr. Ali Wagdi for his valuable comments.

I am also grateful to BCI research team at KAU, to EEG technicians: Marlorline and Soseat, and to the volunteers: Maan H. Harbi, Mota'b Al-Malki, Kahlid Asiri, and Mohammad Kaa'bi, who took part in the experiments and played an important role in making this thesis possible.

My deepest gratitude goes to my family: my mother, my wife and my whole family who inspired and supported me from the beginning to the end.

Finally, This research is considered as part of the main BCI project in the King AbdulAziz University that is funded by (King AbdulAziz City for Science and Technology) KACST, 8-NAN106-3.

FEATURE SELECTION & EXTRACTION ALGORITHMS FOR BRAIN COMPUTER INTERFACE

Anas Abdulqader Mohammed Hadi

Abstract

A brain-computer interface (BCI) is a direct communication pathway between a human brain and an external device. In other words, a BCI allows users to act on their environment by using only brain activity, without using peripheral nerves and muscles. In BCI there are many paradigms; one of them is P300 which occurs in response to a significant but low-probability event. BCI data is considered to be high in their dimensionalities which reduce the system performance.

Feature selection is a dimensionality reduction technique. Feature selection techniques study how to select a subset of features that enhance the performance of the system. The reason behind using feature selection techniques include reducing dimensionality, removing irrelevant and redundant features, reducing the amount of data needed for learning, and improving algorithms' predictive accuracy.

In this thesis, three types of feature selection techniques are compared and applied. These types are filter, wrapper, and hybrid. Fisher score, Determination Coefficient (r^2), Regularized Fisher Linear Discriminant (RFLD), and Bayesian Linear Discriminant Analysis (BLDA) were used as evaluation functions. Differential Evolution (DE) optimization technique was used as searching technique. Two datasets were used to evaluate the results.

Filter types were the preferred to be selected as feature selection method for P300 based BCI, in particular r^2. This is due to the good reduction in dimension 64.8% and low computational cost 6.75ms. The time required for training and testing the classifier was improved by 83.62%.

LIST OF PUBLICATIONS

This thesis has resulted in the following article:

- Mohammed J. Alhaddad, Mahmoud Kamel, Hussein Malibary, Khalid Thabit, Foud Dahlwi, and **Anas Hadi** "*P300 Speller Efficiency with Common Average Reference*", AIS 2012, LNCS 7326, pp. 234–241, 2012

- Mahmoud I. Kamel , Hussein M. Malibary , Mohammed J. Alhaddad , Khalid Thabit, Foud Dahlwi, Ebtehal A. Alsaggaf, **Anas A. Hadi**, "*EEG based Autism Diagnosis Using Regularized Fisher Linear Discriminant Analysis*", I.J. Image, Graphics and Signal Processing, 2012, 3, 35-41, Published Online April 2012 in MECS (http://www.mecs-press.org/), DOI: 10.5815/ijigsp.2012.03.06

- Mohammed J. Alhaddad, Mahmoud I. Kamel , Hussein M. Malibary, Ebtehal A. Alsaggaf, Khalid Thabit, Foud Dahlwi and **Anas A. Hadi**, "*Diagnosis Autism by Fisher Linear Discriminant Analysis FLDA via EEG*", International Journal of Bio-Science and Bio-Technology, Vol. 4, No. 2, June, 2012

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND TERMINOLOGY

AAR: Adaptive Auto Regression.

AR: Auto Regression.

BCI: Brain Computer Interface.

BCI2000: BCI2000 is a general-purpose system for brain-computer interface (BCI) research. It can also be used for data acquisition, stimulus presentation, and brain monitoring applications.

BLDA: Bayesian Linear Discriminant Analysis.

CAR: Common Average Reference.

CWT: Continuous Wavelet Transform.

DE: Differential Evolution.

DWT: Discrete Wavelet Transform.

EEG: Electroencephalogram.

ERPs: Event-Related Potentials.

FFT: Fast Fourier Transform.

FLDA: Fisher's linear Discriminant Analysis.

GA: Genetic Algorithm.

ISI: Inter-Stimulus Interval.

MALAB: (MATrix LABoratory) is a numerical computing environment and fourth-generation programming language. It is currently one of the most popular languages for scientific and numerical computing.

MRPs: Motor-Related Potentials.

P300: It is an endogenous component of ERPs with a latency of about 300 ms which is elicited by rare and/or significant stimuli

PSD: Power Spectral Density.

PSO: Particle Swarm Optimization.

r^2: Determination Coefficient.

RFLD: Regularized Fisher Linear Discriminant.

SBE: Sequential backward elimination

SFFS: Sequential forward floating search.

SFS: Sequential forward selection.

SSVEPs: Steady-State Visual Evoked Potentials.

Chapter 1 Introduction

1.1 Introduction

One of the main factors that make the life of any human being enjoyable is the ability to communicate with other persons. Individuals suffering from the so-called locked-in syndrome do not have the ability of such ability. The locked-in syndrome is a condition in which patients are fully conscious and aware of what is happening in their environment but are not able to communicate or move.

Brain-computer interface (BCI) is a productive research program that aims to resolve such a problem. It has been an expanding field of research and development in recent years. The last two decades innovated BCI's emerged. The idea underlying BCI systems is to measure electrical activity of the brain, and translate them to commands for a computer or other devices. Consequently, users will be able to act on their environment by using their brainwaves instead of their muscles.

This thesis focuses on P300 based Brain Computer Interface (BCI) using EEG signals. In particular, the algorithms of selecting the most appropriate features.

1.2 Motivations

The Motivations for this thesis can be explained by investigating three questions: Why BCI? Why P300 based BCI? And why Feature selection?

BCI Motivations: All over the world, the healthcare quality is increased. The methods to save the life of people who make accidents are optimized. Although the life of those people is saved, they might become handicapped and need more help to live as natural as possible.

The advances in neuroscience, computational technology, component miniaturization, biocompatibility of materials, and sensor technology can be integrated to interpret the intentions of the paralyzed people to achieve their desires. A BCI is one of the most promising outcomes of this integration especially for severely paralyzed persons.

P300 based BCI Motivations: P300 is relatively well understood from a neurophysiologic point of view and can be evoked robustly across different subjects. Moreover, feedback training is not necessary in P300 based BCI systems, as the P300 appears "automatically" whenever subjects concentrate onto one out of several stimuli presented in random order [1].

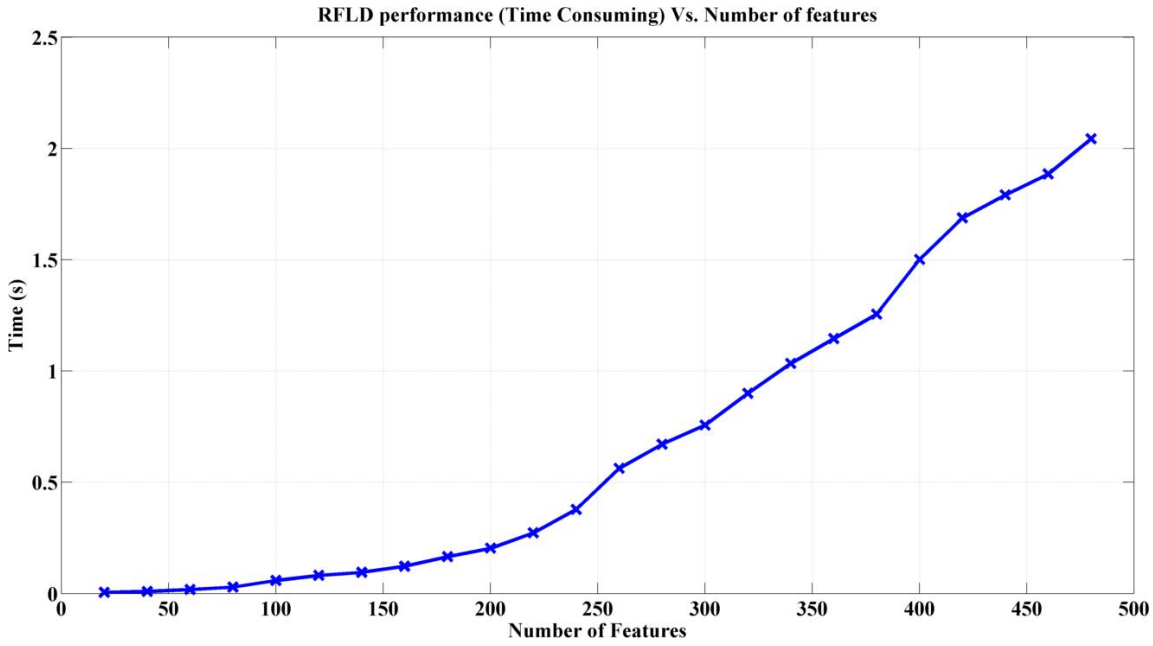

Figure 1-1 Classifier's performance and the number of features

Feature Selection Motivations: In P300 based BCI we have hundreds to thousands of features with many irrelevant or redundant ones. Irrelevant and redundant features can confuse the classifier. Moreover, the classifier's performance is decreased if the number of features is increased. Figure1-1 illustrates this problem from time consuming point of view using RFLD. Finally, Feature Selection is one of the two methods which can solve the curse of dimensionality problem. The other method is feature extraction.

1.3 Objectives

1. Design and Implementation of an offline P300 based BCI system using EEG signals using machine learning algorithms and signal processing techniques.
2. Improving P300 based BCI performance by selecting the most appropriate features.
3. Improving the infrastructure for EEG lab for further BCI research at King Abdul-Aziz Hospital (KAUH).
4. Transfer this technology P300 based BCI to our community.

1.4 Scope of the Thesis

The scope of this thesis is a dimensionality reduction problem using feature selection algorithms. The application of this problem is P300 based BCI which is considered to be a binary classification problem. Two datasets were used. One of them is from our BCI lab at KAUH, and the other is from U. Hoffmann et al [1].

1.5 Thesis Problem Definition

Considering our binary classification problem, Let $X \in R^k$ be the input EEG vectors. And let $Y \in \{1,0\}$ be the corresponding class. Let $F = \{f_1, \ldots., f_i, \ldots., f_n\}$ be the set of all features under examination, and let:

$$Tr = \{(X(l), Y(l)) \mid l = 1, 2, \ldots, N\} = \{[x1(l)\ x2(l) \ldots xk(l)]T, Y(l) \mid l = 1, \ldots, N\}$$

Denotes the training set containing N training pairs, where $x_i(l)$ is the numerical value of feature f_i for the lth training sample.

The goal of feature selection is to find a minimal set of features

$$Fs = \{fs_1, \ldots., fs_i, \ldots., fs_d\}$$

To represent the input vector X in a lower dimensional feature space as

$$Xs = \{Xs_1, \ldots., Xs_i, \ldots., Xs_d\}$$

Where $d < n$ and while the classifier obtained in the low dimensional representation still yields the acceptable classification accuracy.

1.6 Outline of the Thesis

The rest of this thesis is organized into five chapters:

Chapters 2 and 3 contain background material and literature review, Chapter 2 will be an introduction to Brain Computer Interface. This will include BCI definition, brief background about BCI, and neurophysiologic signals used in BCI. After that P300 based BCI is described in details. Chapter 3 will be dedicated to feature extraction and selection for BCI. The definition and the difference between them are introduced. Feature extraction methods are explained then the algorithms used for feature selection are described. Although the literature review is covered through the whole two chapters, a literature review for feature selection in BCI is given separately by the end of chapter 3.

Chapters 4 and 5 mainly describe research specific to this thesis. Chapter 4 describes the methodology and the materials used for feature selection for P300 based BCI, as well as, the implementation details of the proposed model. Chapter 5 will present and discuss the results. Conclusion and an outlook on future work will be in Chapter 6.

Chapter 2 Introduction to Brain Computer Interfaces

2.1 Brief Background

In 1875, the existence of electrical currents in the brain was discovered by a Liverpool surgeon named Richard Caton. He studied action potentials from the exposed brains of rabbits and monkeys [2]. In 1929, Hans Berger used his ordinary radio equipment to amplify the brain's electrical activity measured on the human scalp. This electrical activity is called the electroencephalogram, or EEG, which was the first EEG recording of humans. He showed that weak electric currents generated in the brain can be recorded without opening the skull, and depicted graphically on a strip of paper [3]. Berger's observations were confirmed by other respected physiologists and led to the acceptance of the EEG as a real phenomenon in 1935 [3]. In 1964 Grey Walter and his colleagues reported the first cognitive ERP component, which they called the contingent negative variation or CNV. This negative voltage—the CNV—was clearly not just a sensory response. Instead, it appeared to reflect the subject's preparation for the upcoming target. This exciting new finding led many researchers to begin exploring cognitive ERP components [2, 3]. The next major advance was the discovery of the P3 component by Sutton et al. (1965). They found that when subjects could not predict whether the next stimulus would be auditory or visual, the stimulus elicited a large positive P3 component that peaked around 300 ms post-stimulus; this component was much smaller when the modality of the stimulus was perfectly predictable [3].

Over the past two decades, productive BCI research groups have arisen to explore this new potential communication modality. Facilitated and encouraged by new understanding of brain function, by the advent of powerful low-cost computer equipment, and by growing recognition

of the social needs of people with disabilities, these groups focused their attention on developing a new alternative communication and control technology to exploit this medium. Today, BCI systems propose to offer humans a new non-muscular modality through which to communicate directly with their environment. They rely on the acquisition and interpretation of the user-controlled commands encoded in the neurophysiologic signals [2].

2.2 What is Brain Computer Interface (BCI)?

In basic terms, a BCI involves monitoring brain activity (via a brain imaging technology) and detecting characteristic brain pattern alterations that the user controls in order to communicate (via digital signal processing algorithms) with the outside world. In a BCI, messages and commands are expressed not by muscle contractions as with conventional communication means, but rather by electrophysiological signals generated within the brain [2] Figure 2-1 shows the basic design of any BCI system.

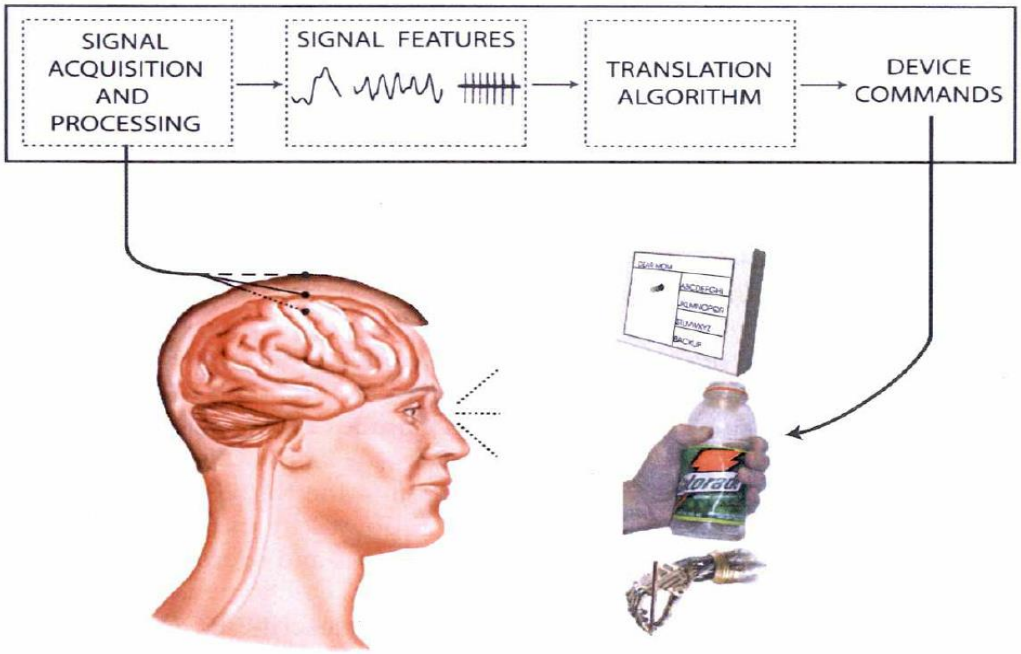

Figure 2-1 Basic design and operation of any BCI system

BCI systems can be classified into invasive and non-invasive. The invasive BCI involves attaching electrodes directly to the brain tissue. Non-invasive BCI involves putting electrodes on the scalp of the patient and record the signals. Non-invasive BCI have poorer spatial resolution, but it is preferable to their safety and cost issues. The main focus of this research will be on non-invasive BCI. *Mason et al.* [4] present a comprehensive survey of BCI technology published prior to January 2006.

2.2.1 Neurophysiologic Signals

Neurophysiologic signals used in BCI can be classified as:

1. Event-Related Potentials (ERPs)
    a. P300
    b. Steady-State Visual Evoked Potentials (SSVEPs)
    c. Motor-Related Potentials (MRPs)
2. Oscillatory Brain Activity
    a. Sensorimotor Rhythms
    b. Other Oscillatory Activity
3. Slow Cortical Potentials
4. Neuronal Ensemble Activity

In this Thesis, only Event-Related Potentials (ERPs) will be considered.

2.2.1.1 Event – Related Potentials (ERPs)

ERPs are spatial-temporal patterns of the brain activity, acquiring time-locked to an event – i.e. after presentation of a stimulus, before execution of a movement, or after the detection of a novel stimulus. Traditionally, ERPs are recorded with EEG. Here is a list of the most ERP signals used:

P300

It is an endogenous component of ERPs with a latency of about 300 ms which is elicited by rare and/or significant stimuli (visual, or auditory). It was found in 1965 by (Sutton et al., 1965; Walter, 1965). The amplitude of the P300 potential is largest at the parietal electrode sites, see Figure 2-1. Effectively P300 potentials are ERP components whose presence depends on whether or not a user attends a rare, unexpected target stimulus. This is what makes it possible to use them in BCI systems to determine user intentions [5].

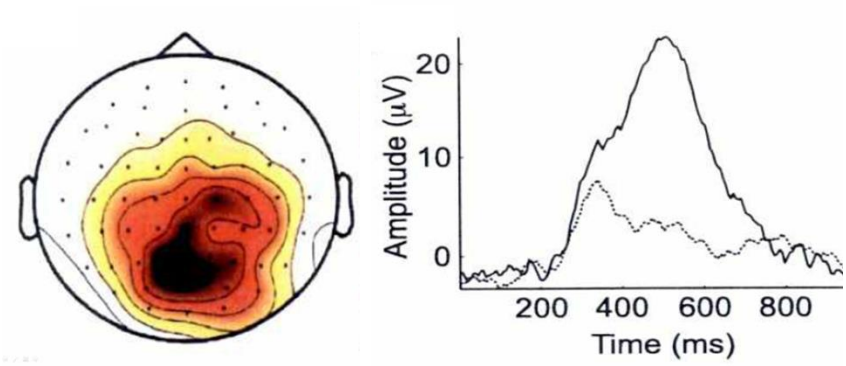

Figure 2-2 Spatial and temporal location of P300

The characteristics of the P300 component (mainly its amplitude and its latency) vary depending on several factors. Some factors are related to the psychophysical state of the subject, such as food intake, fatigue, assumption of drugs. Others factors depend on the physical layout of the stimuli, such as number of symbols, their size, their relative spacing. Other important factors are related to the sequence of stimuli. For example, several studies have reported that the P300 amplitude increases as target probability decreases. The P300 amplitude seems also to be positively correlated with the inter-stimulus interval (ISI) [1]. ISI is the amount of time between two consecutive stimuli.

In this thesis, P300 will be used to construct the P300-based BCI system.

Steady-State Visual Evoked Potentials (SSVEPs)

SSVEPs are oscillations observable at occipital electrodes, induced by repetitive visual stimulation. Stimulation at certain frequencies leads to oscillations at the same frequency and at harmonics and sub-harmonics of the stimulation frequency. Figure 2-2 illustrates this phenomenon using two targets.

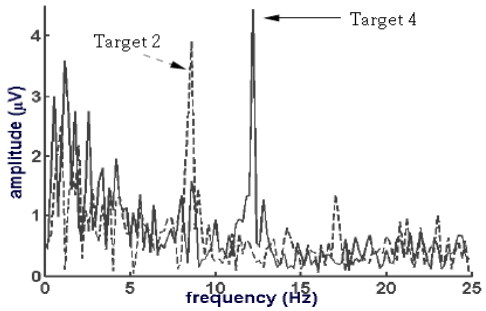

Figure 2-3 SSVEP frequencies for two targets

Motor-related potentials (MRPs)

Other than the previously described signals MRPs are independent of the perception or processing of stimuli. The event which MRPs are related to is the preparation or imagination of movement. They are observable over the sensorimotor cortex before movements' onset or during movement imagination. Figure 2-3 illustrates this phenomenon.

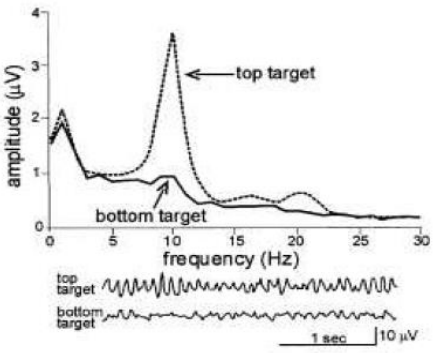

Figure 2-4 MRP frequencies target and non-target

2.3 P300 based BCI System

Figure 2-4 shows the main blocks in the general P300 based BCI System. Here the EEG signals are recorded from scalp using brain activity monitoring device. Then, EEG signals are fed to Preprocessing and Feature selection extraction blocks were pre-processing, feature selection, and extraction are done on the EEG signals, and the output will be fed to the Learning block during the training phase or to the Classification block during the testing phase. The Classification result command will be translated to feedback using Device Controller.

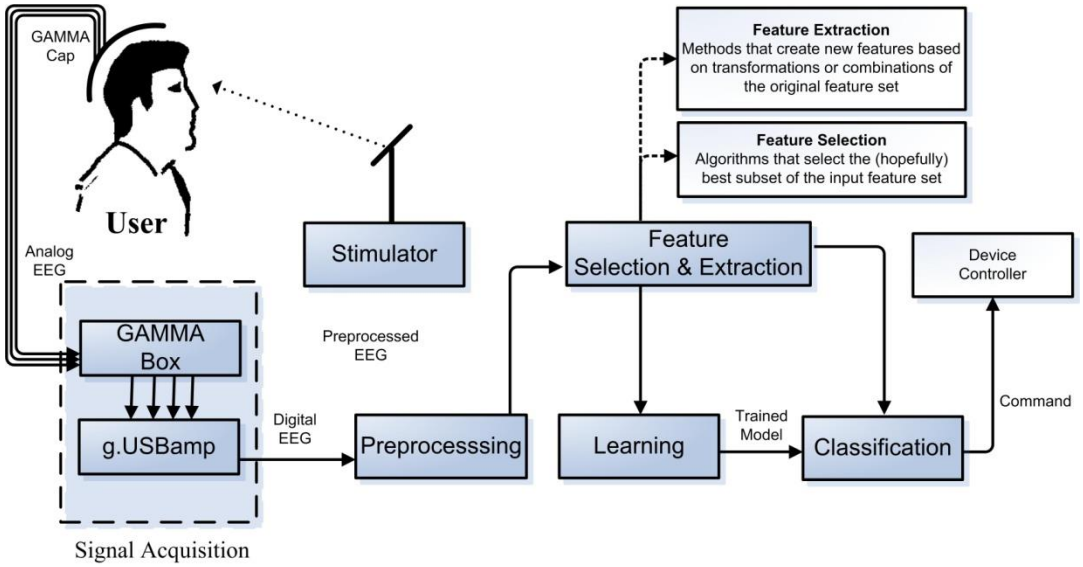

Figure 2-5 Functional model of a BCI System

2.3.1 Signal Acquisition

To enable the communication via BCI, first brain signals have to be recorded; different recording methods can be used. The main functions of this module are:

1. Recording the data from the brain.
2. Doing some low level filtering.
3. Pass the data to be interpreted.

The recorded data could be EEG, ECog, MEG, fMRI, or NIRS. Among them EEG were one of the most widely used non-invasive technique for recording electrical brain activity. It has been employed to answer many different questions about the brain activity, and has served as a diagnostic tool on clinical practice. EEG is a popular signal acquisition technique because the required devices are simple and cheap, and the preparation of the measurement takes only a small amount of time. On the other hand it has a very good temporal resolution. The electrodes used are placed in the scalp in standardized positions, usually 10-20 scheme, see Figure 2-5. In this thesis, the P300 based BCI will be constructed using EEG signals.

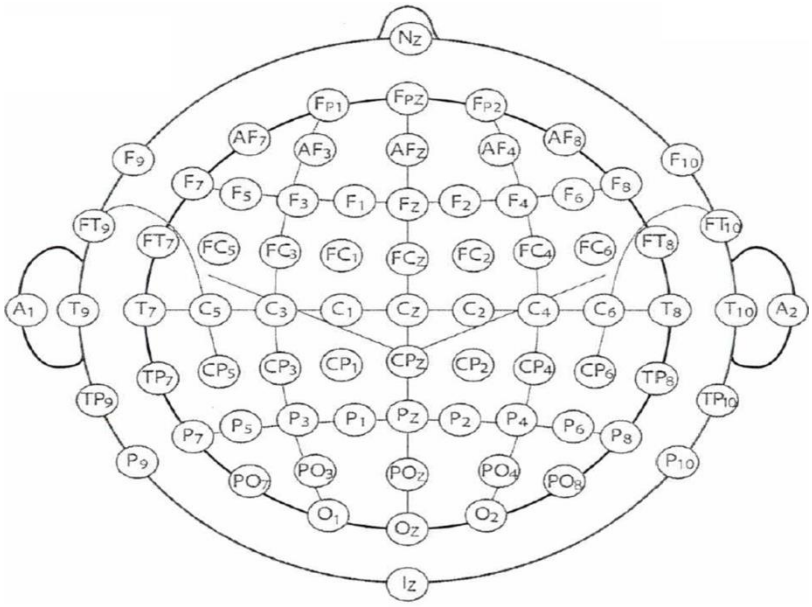

Figure 2-6 Electrodes locations in the 10-20scheme

2.3.2 Feature Extraction and selection

One major challenge in optimizing the performance of the P300 based BCI is enhancing the real-time detection of the P300 elicited by the chosen stimuli. The process of real time detection consists of extraction and selection of P300 features which best represent the user's intentions and classification of the extracted features into an appropriate output by the selected algorithm [6]. In order to select the most appropriate classifier for a given BCI system, it is essential to

clearly understand what features are used, what their properties are and how they are used. Concerning the design of a BCI system, some critical properties of these features must be considered:

- **Noise and outliers:** BCI features are noisy or contain outliers because EEG signals have a poor signal-to-noise ratio.
- **High dimensionality:** In BCI systems, feature vectors are often of high dimensionality. Indeed, several features are generally extracted from several channels and from several time segments before being concatenated into a single feature vector.
- **Time information:** BCI features should contain time information as brain activity patterns are generally related to specific time variations of EEG.
- **Non-stationarity:** BCI features are non-stationary since EEG signals may rapidly vary over time and more especially over sessions [7].

Chapter 3 will be dedicated to cover Feature Selection and Extraction methods.

2.3.3 Machine learning and classification

After feature extraction and selection, supervised machine learning algorithms are applied to learn the classifiers. Supervised classifiers can be categorized in general in to linear and non-linear methods. Over the last decade several more sophisticated non-linear classification methods, like support vector machines and random forests, have been proposed, but it is wise to try linear ones first (of course using shrinkage estimation). For BCI, it was agreed that simplicity is generally best and, therefore, the use of linear methods is recommended wherever it possible [8]. Linear methods are discriminant algorithms that use linear functions to distinguish classes. They are probably the most popular algorithms for BCI applications. Two main kinds of linear classifier have been used for BCI design, namely, Linear Discriminant Analysis (LDA) and Support Vector Machine (SVM) [7]. In this thesis our concern is going to be

dedicated to LDA and in particular Fisher's Linear Discriminant Analysis (FLDA) and the regularized versions of it Regularized Fisher Linear Discriminant (RFLD) and Bayesian Linear Discriminant Analysis (BLDA). The regularized version of FLDA may give better results for BCI than the non-regularized version. Surprisingly, it is much less used than LDA for BCI applications [7].

2.3.3.1 Fisher's Linear Discriminant Analysis (FLDA)

Fisher's linear discriminant is a method used in statistics, pattern recognition and machine learning to find a linear combination of features which separate two or more classes of objects or events. This linear combination of the measured variables is easy to interpret [9]. The resulting combination may be used as a linear classifier or, sometimes, for dimensionality reduction before later classification.

Consider a set of features $\vec{x}$ (also called observations, attributes, variables or measurements) for each sample of an object or event with known class *y*. This set of samples is called the training set. The classification problem is then to find a good predictor for the class *y* of any sample of the same distribution (not necessarily from the training set) given only a features $\vec{x}$.

Suppose we have two classes of features with means $\vec{\mu}_{y=0}, \vec{\mu}_{y=1}$ and covariances $\Sigma_{y=0}$, $\Sigma_{y=1}$. Then the linear combination of features $\vec{w} \cdot \vec{x}$ will have means $\vec{w} \cdot \vec{\mu}_{y=i}$ and variances $\vec{w}^T \Sigma_{y=i} \vec{w}$ for $i = 0, 1$.

Fisher will choose w, which maximize:

$$J(w) = \frac{W^T S_B W}{W^T S_w W}$$

Where $\boldsymbol{S_B}$ is the between classes scatter matrix and $\boldsymbol{S_W}$ is the within classes scatter matrix and:

$$S_B = (\mu_{y=1} - \mu_{y=0})(\mu_{y=1} - \mu_{y=0})^T$$

$$S_w = S_1 + S_2$$

It can be shown that the maximum separation occurs when [60]:

$$\vec{w} = (\Sigma_{y=0} + \Sigma_{y=1})^{-1}(\vec{\mu}_{y=1} - \vec{\mu}_{y=0})$$

Generally, the data points to be discriminated are projected onto $\vec{w}$; then the threshold that best separates the data is chosen. There is no general rule for the threshold [61]. However, if projections of points from both classes exhibit approximately the same, the good choice would be the middle between projections of the two means, $\vec{w} \cdot \vec{\mu}_{y=0}$ and $\vec{w} \cdot \vec{\mu}_{y=1}$. In this the threshold *c* can be found explicitly:

$$c = \vec{w} \cdot (\vec{\mu}_{y=0} + \vec{\mu}_{y=1})/2$$

2.3.3.2 Regularized Fisher Linear Discriminant (RFLD)

In FLDA, The standard estimator for a covariance matrix is the empirical covariance. This estimator is unbiased and has under usual conditions good properties. But for extreme cases of high-dimensional data with only a few data points given- as our case - the estimation may become imprecise. This leads to a systematic error: Large eigen values of the original covariance matrix are estimated too large, and small eigen values are estimated too small; see figure 3-1. This error in the estimation degrades classification performance.

Regularization is a common remedy for the systematic bias of the estimated covariance matrices. For the RFLD we use:

$$\tilde{\boldsymbol{\Sigma}}(\gamma) := (1-\gamma)\hat{\boldsymbol{\Sigma}} + \gamma\nu\mathbf{I}$$

Where $\boldsymbol{\gamma}$ is a tuning parameter $\boldsymbol{\gamma} \in [0, 1]$ and ν defined as average eigenvalue trace $\bar{\boldsymbol{\Sigma}}/\boldsymbol{d}$ of $\bar{\boldsymbol{\Sigma}}$ with $\boldsymbol{d}$ being the dimensionality of the feature space and $\boldsymbol{I}$ being the identity matrix. There are many methods to estimate optimum $\boldsymbol{\gamma}$ [59], we test three of with toy example and we choose $\boldsymbol{\gamma}$ using B. Blankertz et al. [10] method:

$$\gamma^* = \frac{n}{(n-1)^2}\frac{\sum_{i,j=1}^{d} var_k(z_{ii}(k))}{\sum_{i\neq j} s_{ij}^2 + \sum_i (s_{ii}-v)^2}$$

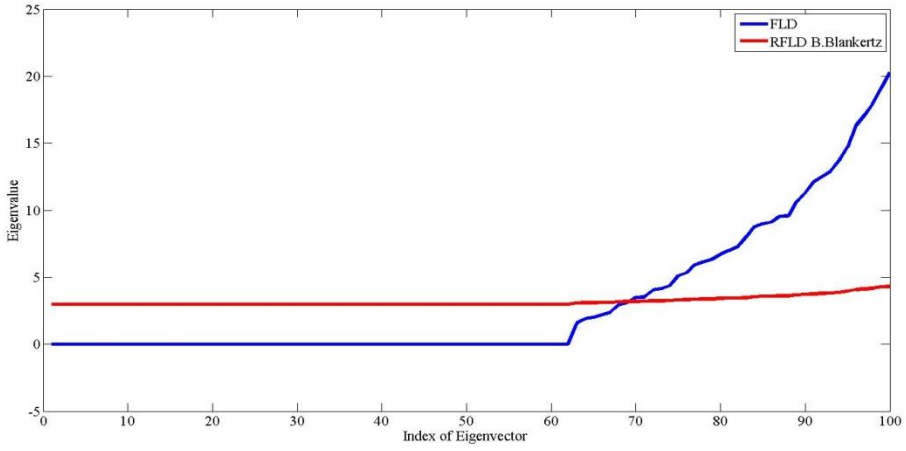

Figure 2-7 Eigenvalues of a given covariance matrix using FLDA (blue line) and RFLD (red line)

2.3.3.3 Bayesian Linear Discriminant Analysis (BLDA)

BLDA can be seen as an extension of Fisher's Linear Discriminant Analysis (FLDA). In contrast to FLDA, in BLDA regularization is used to prevent over-fitting to high dimensional and possibly noisy datasets. Through a Bayesian analysis the degree of regularization can be estimated automatically and quickly from training data. This method is similar to the effect of the regularization term used in regularized FLDA.

It has been proven [60, 63] that least squares regression is equivalent to FLDA if regression targets are set to N/N_1 for instances from class 1 and to$-N/N_2$ for instances from class-1 (where N is the total number of training instances, N_1 the number of instances from class 1, and N_2 the number of instances from class -1). According to this, BLDA is equivalent to performing Bayesian regression and setting target values to N/N_1 and $-N/N_2$for instances from class 1 and class 2 respectively.

The assumption in Bayesian regression is that targets *y* and feature vectors *x* are linearly related with additive white Gaussian noise *n*.

$$y = w^T x + n$$

According to Bayesian rule:

$$posterior = \frac{likelihood \times prior}{evidence}$$

The proper normalized expression for the *likelihood* function is:

$$p(D|\beta, w) = (\frac{\beta}{2\pi})^{N/2} \exp(-\frac{\beta}{2}||x^T w - y||^2)$$

Denoting by β the inverse variance of the noise and by *D* the pair *{x, t}*.The expression for the normalized *prior* distribution is:

$$p(w|\alpha) = (\frac{\alpha}{2\pi})^{D/2}(\frac{\in}{2\pi})^{1/2} \exp(-\frac{1}{2} w^T I'(\alpha) w)$$

Where *D* is the number of features and $I'(\alpha)$ is a square *D* + 1 dimensional, diagonal matrix:

$$I'(\alpha) = \begin{bmatrix} \alpha & 0 & \cdots & \cdots & 0 \\ 0 & \alpha & \cdots & \cdots & 0 \\ \vdots & \vdots & \ddots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & \in \end{bmatrix}$$

The *prior* for the weights is thus a zero-mean Gaussian distribution with variance$\frac{1}{\alpha}$. Given *likelihood* and *prior*, the *posterior* distribution of *w* can be computed using Bayesian rule:

$$p(w|\alpha,\beta,D) = \frac{p(D|\beta,w)p(w|\alpha)}{\int p(D|\beta,w)p(w|\alpha)\,dw}$$

Since both *prior* and *likelihood* are Gaussian, the *posterior* is also Gaussian and its parameters can be derived from *likelihood* and *prior* by completing the square [63]. The mean ***m*** and covariance **C** of the *posterior* satisfy the following equations:

$$m = \beta A^{-1} xy$$

$$C = A^{-1}$$

Where"

$$A = \beta x x^T + I'(\alpha)$$

By multiplying the *likelihood* function for a new input vector *x* with the *posterior* distribution and integrating over *w* we obtain the *predictive distribution*. The *predictive distribution* is again Gaussian and can be characterized by its mean μ and its variance σ^2:

$$\mu = m^T x$$

$$\sigma^2 = \frac{1}{\beta} + x^T C x$$

In the P300-based BCI described in [1], only the mean value of the predictive distribution was used for taking decisions. Both the *posterior* distribution of *w* and the *predictive distribution* depend on the hyper parameters α and β. Bayesian regression framework offers a more elegant and less time-consuming solution for the problem of choosing these hyper parameters. The idea is to write down the *likelihood* function for

the hyper parameters and then maximize the *likelihood* with respect to the hyper parameters. The maximum *likelihood* was described in details in [63].

Defining the following eigen vector equation:

$$x^T x\, \varphi = \lambda\, \varphi$$

Where φ is the eigen vector and λ is the eigen value. According to this, α and β can be computed by:

$$\alpha = \frac{\gamma}{m^T m}$$

$$\beta = \frac{N - \gamma}{error}$$

Where γ is defined as:

$$\gamma = \sum_i \frac{\beta \lambda i}{\alpha + \beta \lambda i}$$

And *error* is the minimum squares error or least squares regression and defined as:

$$error = \sum_i^N (y - m^T x(i))^2$$

Both α and β depend on the $\boldsymbol{m}$ which itself depends on α and β. This represents an implicit solution for the hyper parameters. Thus, to maximize it an iterative scheme is used in which first $\boldsymbol{m}$ is computed for an initialization setting of α and β and then they are updated according to equations above. After few iterations the values for α and β converge to the maximum-likelihood solution. A MATLAB implementation of BLDA can be downloaded from the webpage of the EPFL BCI group (http://bci.epfl.ch/p300)

2.4 Applications

In theory any device that can be connected to a computer or to a microcontroller could be controlled with a BCI. In practice however, the set of devices and applications that can be controlled with a BCI is limited. To understand this, one has to consider that the amount of information which can be transmitted with present day BCI systems is limited. Some of the applications possible with current BCIs are:

- Spelling devices
- Environment Control.
- Wheelchair Control.
- Games and Virtual Reality.

Chapter 3 Feature Extraction and Selection for BCI

3.1 Introduction

There are two ways to reduce the dimensionality and avoid the curse of dimensionality in BCI, one of them is ***feature selection*** and the other is ***feature extraction*** [11].

Feature Selection and Feature Extraction terms are used interchangeably in the literature of BCI. So it's important here to distinguish between them. The term *feature selection* refers to algorithms that select the (hopefully) best subset of the input feature set. On the other hand, Methods that create new features based on transformations or combinations of the original feature set are called *feature extraction* methods [12]. Feature extractor was defined in [4] as the component of a Brain-Interface Transducer that translates the (artifact-free) input brain signal into a value correlated to the neurological phenomenon.

3.1.1 Feature Selection – Definition

Given a set of features:

$$F = \{f_1, \dots, f_i, \dots, f_n\}$$

Feature Selection problem is to find a subset $F' \subset F$ that maximizes the classifiers ability to classify patterns, see Figure 3-1. The chosen subset of features F' should be neither *irrelevant* nor *redundant* to the target concept.

A feature *Xi* is said to be *relevant* to a concept *C* if *Xi* appears in every Boolean formula that represents *C* and irrelevant otherwise. In [70] the relevance is divided into strong relevance and weak relevance. Strong relevance implies that the feature is indispensable in the sense that it

cannot be removed without loss of prediction accuracy. Weak relevance implies that the feature can sometimes contribute to prediction accuracy. Features are relevant if they are either strongly or weakly relevant and are irrelevant otherwise. Irrelevant feature does not affect the target concept in any way.

Redundant feature does not add anything new to the target concept [13]. In feature selection, it is important to choose features that are relevant for prediction, but at the same time it is important to have a set of features which is not redundant in order to increase robustness [71].

3.1.2 Feature Extraction – Definition

Given a set of features:

$$F = \{f_1, \ldots., f_i, \ldots., f_n\}$$

Feature Extraction problem is to map F to some feature set F'' that maximizes the classifier's ability to classify patterns, see Figure 3-1. This general definition subsumes feature selection; i.e. a feature selection algorithm also performs a mapping but can only map to subsets of the input variables.

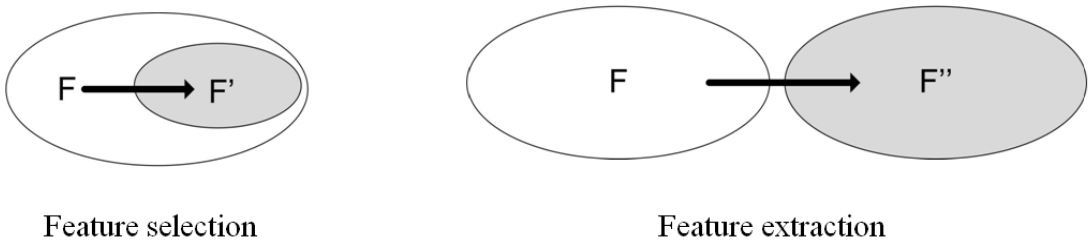

Figure 3-1 Feature selection and Feature extraction

Usually, feature extraction precedes feature selection; first, features are extracted from the data and then some of the extracted features are selected according to their discriminative ability. Feature selection leads to savings in measurement cost and the selected features retain their

original physical interpretation. On the other hand, transformed features generated by feature extraction may provide a better discriminative ability than the best subset of given features, but these new features may not have a clear physical meaning [12].

3.2 Feature Extraction methods

According to their domain, features can be grouped as Time domain, Frequency Domain, and Spatial Domain.

3.2.1 Time Domain Features

Time domain features are related to changes in the amplitude of neurophysiologic signals, occurring time-locked to the presentation of stimuli or time-locked to actions of the user of a BCI system. P300 are one of the signals that can be characterized with the help of time domain features. A strategy that is often used to separate these signals from background activity and noise is lowpass or bandpass filtering, optionally followed by downsampling. This allows to remove unimportant information from high frequency bands and to reduce the dimensionality of the signals. Examples for systems in which filtering and downsampling have been employed are the P300 BCI described in [1, 14]. An alternative to filtering is to use the wavelet transform of the signals. Systems based on the discrete wavelet transform (DWT), as well as systems based on the continuous wavelet transform (CWT) have been described in the literature [15, 16].

3.2.2 Frequency Domain Features

Frequency domain features are related to changes in oscillatory activity. In systems based on motor imagery, the band power in the mu and beta frequency bands at electrodes located over the sensorimotor cortex is used as a feature. In the case of SSVEPs, band power in the harmonics of the visual stimulation frequency at occipital electrodes can be used as a feature. To estimate band power, different methods have been used. These include Fast Fourier Transform (FFT) and

Power Spectral Density (PSD) methods [17], adaptive autoregressive (AAR) models [18], and Morlet wavelets [19] which are classified as *time-frequency features*.

3.2.3 Spatial Domain Features

In many BCI systems, data from more than one electrode is available. Hence, the features extracted from several electrodes have to be combined in an efficient way. Finding efficient combinations of features from more than one electrode is the goal of spatial feature extraction methods. A spatial feature extraction method consists in applying spatial filtering algorithms before further processing takes place. Spatial filtering corresponds to building linear combinations of the signals measured at several electrodes. Different spatial filtrations methods have been used in the literature [20, 21].

Another classification of feature selection algorithms in ECoG based MRP were performed in [6], five types of *MRP feature* were discussed, these types are:

Type1- Statistical features:

a. Moments between all 64 channels,

b. Commulant between 2,3 and 4 channels,

c. Correlation between all 64 channels,

d. Form Factor for each channel separately,

e. Statistical Variance for each channel separately.

Type2- Entropy based features:

a. Approximated Neural Complexity Measure between all 64 channels ,

b. Lample-Ziv Complexity Measure for each channel separately ,

c. Shannon, Renyi and Tsallis Entropies (for α and q = -5, -2, -1, 0.5, 1.5, 2, 3, 5)

d. Approximation Entropy for each channel separately ,

Type3- Signal's Energy in different frequency ranges:

a. δ(0-3.5Hz), θ(3.5-7.5Hz), α(7.5-13.5Hz), β(13.5-20Hz),

b. δ(0-3Hz), θ(4-7Hz), α(8-13Hz), β(14-20Hz),

c. δ(0.5-3.5Hz), θ(3.5-7Hz), α(7.5-13Hz), β1(12-15Hz), β2(15-17.5Hz), β3(18-25Hz), β4(25.5-30Hz),

d. δ(1.5-6Hz), θ(6-8Hz), α1(8.5-10Hz),α2=(10.5-12Hz), β1(12.5-18.5Hz), β2(18.5-21Hz), β3(21-30Hz), β4(30-40Hz).

Type4- Parameters estimated for AR (Autoregressive) model:

a. AR model, order 4,

b. AR model, order 8,

c. AR model, order 16,

d. AR model, order 32.

Type5-Frequency based features:

a. Coefficients of the Discrete Cosine and Sinusoid Transform,

b. Coefficients of the Wavelet Transform (Haar, Daubechies2, Daubechies3, Daubechies4 andDaubechies5).

3.3 Feature Selection algorithms

3.3.1 Feature selection and BCI domains

The features extracted to be used in BCI systems will provide a better or worse separability for the classes depending on where on the scalp they are coming from and where in its domain they are, e.g. P300 peaks do not occur uniformly at all electrodes but are usually stronger at electrodes over Cz. On the other hand, features found in the motor cortex are more likely to provide better separability for classes of a BCI systems based on motor imagery than the features found on the visual cortex, where features found on the visual cortex are more likely to provide better separability for classes of a BCI system based on SSVEP than features found on the motor cortex. Also, for motor imagery based BCI systems, features found on the alpha and beta band

are more likely to provide better separability of classes than features found in other frequency bands. So beside the extraction of features using a specific method is also necessary to select the appropriate features to be used for classification.

3.3.2 Manual and automatic feature selection

The probably simplest way for performing feature selection is to use only electrodes that carry useful information for discrimination of a given set of cognitive tasks. Electrodes can be selected manually or by using an algorithm that automatically selects an optimal electrode subset. Due to its simplicity the former approach has been used in almost all types of BCIs. The latter, more complex approach has been used for classification of data recorded with a sensorimotor rhythm paradigm such as [22, 29, 34, 35, 37, 38, 39, 40, 44, 45, 46, 47, 48, 49], and classification of P300 data such as [23, 24, 41, 42, 50, 51, 52].

3.3.3 Supervised, Unsupervised, and Semi-Supervised Feature Selection

Feature selection algorithms can be categorized – according to availability of class information – to Supervised, Unsupervised, and Semi-Supervised Feature Selection [66].

In supervised feature selection algorithms, measures are based on the class information. Supervised feature selection algorithms try to find features that help separate data of different classes. If a feature has no effect on class-based separation, it can be removed. A good feature should, therefore, help enhance class-based separation. A well-known measure is information gain, which is widely used in feature selection. ReliefF, r^2…etc are other examples of this algorithm.

In order to deal with data without class labels, unsupervised feature selection was needed. It is closely related to unsupervised learning [66]. One widely used clustering algorithm is k-means. Unsupervised feature selection is more difficult to deal with than supervised feature selection. However, it also is a very useful tool if the majority of data are unlabeled.

When a small number of instances are labeled but the majorities are not, semi-supervised feature selection is designed to take advantage of both the large number of unlabeled instances and the labeling information as in semi-supervised learning. Intuitively, the additional labeling information should help constrain the search space of unsupervised feature selection. In other words, semi-supervised feature selection attempts to align locality-based separation and class-based separations. Since there are a large number of unlabeled data and a small number of labeled instances, it is reasonable to use unlabeled data to form some potential clusters and then employ labeled data to find those clusters that can achieve both locality-based and class-based separations.

A comprehensive introduction and review of Supervised, Unsupervised, and Semi-Supervised Feature Selection is presented in [66]. For my research, supervised feature selection algorithm is selected. Although unsupervised feature selection can be used to skip the training session, this will be at the cost of complexity and accuracy.

3.3.4 Single-Objective (SO) and Multi-Objective (MO) Optimization

Many real-world optimization problems need to achieve several objectives. The main goal of single- objective optimization is to find the "best" solution, which corresponds to the minimum or maximum value of a single objective function that lumps all different objectives into one. This type of optimization is useful as a tool but usually cannot provide a set of alternative solutions that trade different objectives against each other. On the contrary, in a multi-objective optimization with conflicting objectives, there is no single optimal solution. The interaction among different objectives gives rise to a set of compromised solutions, largely known as the trade-off, non-dominated, non-inferior or Pareto-optimal solutions [67].

In our problem we have two objectives: maximize the accuracy and minimize the number of features. We will convert this MO problem into a SO problem. This conversion could be done by

aggregating all objectives in a weighted objective function [68]. Section 4.4.2.2 describes this conversion.

3.3.5 Typical Feature Selection Method

Feature selection can be considered as a searching problem. Here we are searching for the best subset of feature among the competing 2^N candidate subsets. This search is subjective to some evaluation function. However this procedure is exhaustive as it tries to find only the best one. Finding best subset of features is intractable and many problems related to feature selection have been shown to be NP-hard [44]. Other methods try to reduce the computational complexity by selecting efficient searching algorithms such as random or heuristic ones. These methods need a stopping criterion to prevent an exhaustive search of subsets. Liu and Yu [25] as in Figure 3-2 suggest four steps in a typical feature selection method. These four steps are:

- Generation procedure.
- Evaluation function.
- Stopping criterion.
- Validation procedure.

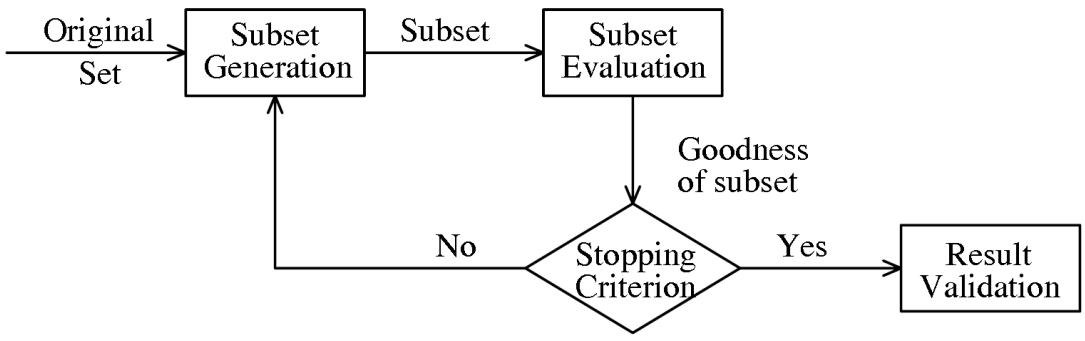

Figure 3-2 Typical feature selection method

The generation procedure generates subsets of features for evaluation. This procedure can be initialized with no features, with all features, or with a random subset of features. According to the first two cases, features are added or removed iteratively, whereas in the last case, features are either iteratively added or removed or produced randomly thereafter.

An evaluation function evaluates the subset produced by the generation procedure. According to this evaluation, the selected features subset is updated. Without a suitable *stopping criterion* the feature selection process may run exhaustively or forever through the space of subsets. Stopping criteria can be:

1. Predefined number of features is selected.
2. Predefined number of iterations is reached.
3. Addition (or deletion) of any feature does not produce a better subset.
4. Optimal subset according to some evaluation function is obtained.

The validation procedure is not a part of the feature selection process itself, but a feature selection method must be validated.

3.3.6 Feature selection types:

In general, feature selection methods are divided into filters, wrapper, and hybrid methods [26, 27]. *Filters* select subsets of variables as a pre-processing step, independently of the chosen classifier. *Wrappers* utilize the learning machine of interest as a black box to score subsets of variable according to their classification power. *Hybrid* model attempts to take advantage of the two models by exploiting their different evaluation criteria in different search stages.

3.3.6.1 Filter Methods

Filter methods select a subset of features from the data based on some filter rule before the classification algorithm is trained see Figure 3-3. If available, the filter rule is derived from prior knowledge (e.g. from an expert). If no prior knowledge about the classification problem is at

hand, the rule is based on data statistics. Most filter methods require strong assumptions about the class distributions of the data to work efficiently. If the assumptions are true, filter methods are very quick methods, easy to implement and can sometimes even deliver best possible solutions. The risk of over-fitting the subset selection to the classification task is rather small compared to other approaches. For this reason, a filter method is often consulted for comparison with new methods. However, if the assumptions are wrong, filter methods might perform poorly. [26].

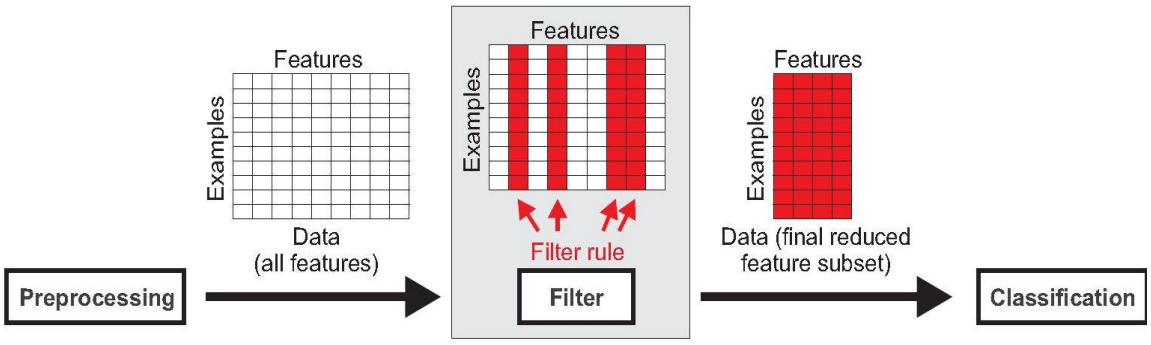

Figure 3-3 Filter method as feature selection

Filter methods usually rank the available features one-by-one according to a relevance criterion (score). It is then up to the user to define a final feature subset by e.g. choosing the *n* best ranked features or to determine a threshold of the score above which a feature is chosen.

In general, relevance criterion can be categorized as: Distance functions (e.g. Euclidean distance measure), Information (e.g. Entropy, Information Gain, etc.), Dependency (e.g. Correlation Coefficient), Consistency (e.g. min-features bias), and classifier error rate (e.g. RFLD, BLDA, etc). In the following, some common relevance criteria are introduced.

Correlation Coefficient

(Pearson's) correlation coefficient is a very well-known linear criterion. It determines for a single feature *j* how strong it is linearly correlated with the labels *Y*.

$$R(j) = \frac{COV(Xj, Y)}{\sqrt{var(Xj)var(Y)}}$$

The numerator is simply the covariance between feature *j* and the labels, while the denominator only serves for normalization so that *R* is invariant for scaling of the feature and the labels. Its values are in the range of [−1, 1]. |R|>0.5 stands for a strong linear dependency, values close to 0 for linear independence.

Determination Coefficient (r^2)

A slightly modified Correlation Coefficient is the Determination Coefficient (r^2) [28].

$$r^2(j) = \frac{COV(Xj, Y)}{var(Xj)var(Y)}$$

Fisher Criterion (FC)

The Fisher Criterion determines how strongly a feature can separate the given classes. It considers both, the within-class variances to be minimized and the between-classes distance to be maximized. The score *FC(j)* of feature *j* is given by:

$$FC(j) = \frac{(\mu 1 - \mu 2)^2}{var(X1) + var(X2)}$$

The ranking of all features is easily obtained by sorting the features according to their score.

Other methods

Other relevance criteria include Scattering Matrices, Mahalanobis Distance, and Bhattacharyya Distance.

3.3.6.2 Wrapper Methods

In a wrapper approach, the selected feature subset is dependent on the result of the classification error estimation (e.g. by cross-validation). Subsets leading to small classification errors estimates

are preferred over subsets resulting in higher error. Thus the feature selection and the classifier are not independent of each other anymore see Figure 2-4. The error rate of the classifier has to be estimated during all iterations the search, which can be very costly.

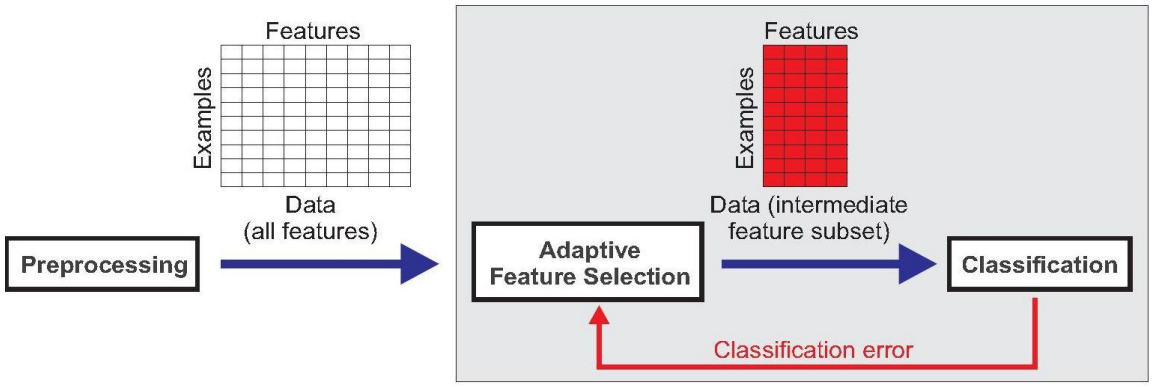

Figure 3-4 Wrapper method as feature selection

3.3.6.3 Hybrid Methods

To take advantage of the above two models, the hybrid model is recently proposed to handle large data sets. A typical hybrid algorithm makes use of both an independent measure and a mining algorithm to evaluate feature subsets: it uses the independent measure to decide the best subsets for a given cardinality and uses the mining algorithm to select the final best subset among the best subsets across different cardinalities [25].

3.3.7 Popular Search Strategies

3.3.7.1 Sequential forward selection (SFS)

This strategy iteratively adds one feature to the current subset until the classification error estimate does not improve any further or until other criteria are met. During iteration, each of the remaining features (one at a time) is added to the current feature subset and the error is estimated. Then the feature yielding the biggest improvement is permanently added to the

feature subset and the next iteration starts. The iteration starts with one feature only, which is chosen according to the lowest error. SFS can need a maximum of $d*(d-1)/2$ error estimations. Where d is the number of features.

3.3.7.2 Sequential backward elimination (SBE)

This strategy is a reversed SFS. Starting with all features, iteratively one feature of the current subset is removed until some stopping criterion is met. As the estimation of training errors is dependent on the number of features used, SBE is usually slower than SFS.

3.3.7.3 Sequential forward floating search (SFFS)

SFFS is a combination of SFS and SBE. Starting like the SFS strategy, a feature subset is built up until no further improvement is possible. Instead of stopping at this (possibly local) optimum, SFFS allows for some backward elimination steps before the search is continued by SFS in another subspace of the problem. As a worst case SFFS can lead to a full search testing $2^d - 1$ subsets.

3.3.7.4 Genetic Algorithm (GA)

Genetic algorithms are based on evolutionary principles, where feature subsets are coded in the form of simple sequences which are considered the genome of the individuals of a population. The population changes by reproduction of its individuals. For reproduction, operators like mutation and crossing over are applied. The fitness of individuals is represented by the classification performance of the corresponding feature subset and determines the chance of reproduction. Over several generations the fitness of the population and its individuals improves. When a stopping criterion is met, the feature subset represented by the fittest individual is selected. GA's are optimization strategies that do not assume a continuously differentiable search space. In a population usually feature subsets of varying numbers of features are present

that initially cover the search space randomly. GA's usually need more error estimations than e.g. the SFS strategy [29].

3.3.7.5 Particle Swarm Optimization (PSO)

The optimization method known as Particle Swarm Optimization (PSO) is originally due to Kennedy, Eberhart, and Shi [30, 31]. It works by having a swarm of candidate solutions called particles, each having a velocity that is updated recurrently and added to the particle's current position to move it to a new position.

3.3.7.6 Differential Evolution (DE)

The parallel multi-agent optimization method known as Differential Evolution (DE) is originally due to Storn and Price [32]. DE generates new agents by adding the weighted difference between two agents to a third agent. This operation is called ***mutation***. The mutated agents are then mixed with another predetermined agent, the ***target agent***, to yield the so-called ***trial agent***. Agents' mixing is often referred to as ***crossover***. If the trial agent yields a lower cost function value than the target agent, the trial agent replaces the target agent in the following generation. This last operation is called ***selection***. More specifically DE's basic strategy can be described as follows:
Let $\boldsymbol{x_i}$ denote the position of a target agent being updated and which has been picked at random from the entire population ***NP***.

Mutation

The mutant agent $\boldsymbol{y_i}$ (trial agent) is generated according to:

$$y_i = a_i + F(b_i - c_i)$$

Where vectors $\boldsymbol{a_i}$, $\boldsymbol{b_i}$, and $\boldsymbol{c_i}$ are distinct agents positions randomly picked from the population. They are also chosen to be different from the target agent, so that ***NP*** must be greater or equal to four to allow for this condition. The differential weight ***F*** is a real and constant factor which controls the amplification of the differential variation.

Crossover

Crossover is introduced as follows:

$$y_i = \begin{cases} a_i + F(b_i - c_i) & , \quad r_i < CR \; or \; i = i_{rand} \\ x_i & , \quad else \end{cases}$$

The index $r_i \sim U(0,1)$ is picked randomly for each evaluation ***i***, and ***CR*** is the crossover constant ∈ [0, 1].

Selection

To decide whether or not the trial agent $\boldsymbol{y_i}$ should become a member of the population, $\boldsymbol{y_i}$ is compared to the target agent $\boldsymbol{x_i}$ using the greedy criterion. If $\boldsymbol{y_i}$ yields a smaller cost function value than $\boldsymbol{x_i}$, then $\boldsymbol{x_i}$ is set to $\boldsymbol{y_i}$; otherwise, the old agent $\boldsymbol{x_i}$ is retained [32].

The user-defined parameters consist of the differential weight ***F***, the crossover probability ***CR***, and the population-size ***NP***.

In a comparison by [32] DE was more efficient than simulated annealing and genetic algorithms. In 2004 Ali and Torn [63] found that DE was both more accurate and more efficient than controlled random search and another genetic algorithm. In the same year, Lampinen and Storn [64] demonstrated that DE was more accurate than several other optimization methods including four genetic algorithms, simulated annealing and evolutionary programming

3.4 Feature selection for P300 based BCI (literature review)

P300 based BCI can be considered as machine learning application or classification. Hence, comprehensive surveys of feature selection for classification can be found in [25, 12, 27]. A brief survey of machine learning techniques - including feature selection methods - that can be applied to BCI data is given in [33].

Many feature selection algorithms have already been described in BCI literature. Filter is simplest and efficient method. Therefore, it has been commonly used in BCI [34, 35, 36, 37, and38]. In all of them, features are ranked individually according a specific evaluation function. Features containing highest values are selected. The evaluation

functions used were Correlation based feature selection CFS [34, 35], Fast Correlation based filter FCBS [36], r^2 [36, 39], Relief [35, 37], 1R ranking 1RR [35], Principal component analysis PCA [36], across group variance AGV [37], Information gain [35, 38, 40], and Consistency based feature selection [35]. Filter methods are limited to the fact that each feature is evaluated individually. Thus we can not avoid including redundant features, furthermore, features that are relevant only when combination with other features might not be included [27].

Wrapper methods were also used in BCI [41, 42, 43]. They are used mainly with SVM [41, 43]. Although wrapper methods usually produce higher accuracy results compared to filter methods, in high dimensional applications - such as BCI - the utilization of wrappers is not practical due to the high computational expense. Hybrid method which combines the advantages of filters and wrappers was also used in BCI [44].

Classical searching algorithms were used for feature selection in BCI literature. Such algorithms include breadth first searching algorithm [42], backward stepwise selection [45], greedy search [35], and sequential forward floating search [44]. Although exhaustive search is not practical for feature selection, it was also used in BCI [39, 46]. Genetic algorithms were the most used in BCI [36, 37, 41, 42, 43]. Some recent searching algorithms were adopted for BCI such as Particle Swarm Optimization PSO [36, 47] and Differential Evolution DE [48].

Most of the previous methods were applied in BCI in motor imagery paradigm [29, 34, 35, 37, 38, 39, 40, 44, 45, 46, 47, 48, 49]. Less intention were devoted for P300 [41, 42, 50, 51, 52].

However, to the best of my knowledge, the DE approach has not been previously used for feature selection in a P300 based BCI. On other hand no comparison between filter, wrapper, and hybrid was done for feature selection in a P300 based BCI.

Chapter 4 Methodology for Feature selection for P300 based BCI

4.1 Introduction

This chapter describes the implementation of the proposed model. As mentioned in the problem definition. Two datasets are used to study the impact of feature selection algorithms on BCI performance. One of them is from our BCI lab at KAUH, and the other from U. Hoffmann et al. [1]. Table 4-1 illustrates the differences between these datasets. This chapter describes the methodology and the materials used for feature selection for P300 based BCI. The offline analysis is identical to that presented in [1]. Except that I have inserted feature selection module to the system. Figure 4-1 shows the BCI lab at KAUH.

Table 4-1 Comparison between used datasets

	U. Hoffmann et al. dataset	KAUH dataset
Paradigm Description	P300, Screen on which six images were displayed, The images were flashed in random sequences, one image at a time, ISI was 400 ms	P300, Screen on which six characters were displayed, The letters were flashed in random sequences, one raw\column at a time, ISI was 300 ms
Sampling rate	2048Hz	256Hz
Sampling rate after down sampling	32Hz	32Hz
Subjects Number	4 normal + 4 abnormal	4 normal
Electrodes no.	32	8
Electrodes Locations	Fp1, AF3, F7, F3, FC1, FC5, T7, C3, CP1, CP5, P7, P3, Pz, PO3, O1, Oz, O2, PO4, P4, P8, CP6, CP2, C4, T8, FC6, FC2, F4, F8, AF4, Fp2, Fz, and Cz	F3, F4, C3, Cz, C4, P3, Pz and P4
Electrodes no. for offline analysis	8	8
Electrodes Locations for offline analysis [1]	Fz, Cz, P7, P3, Pz, P4, P8 and Oz	F3, F4, C3, Cz, C4, P3, Pz and P4
H/W Filtering	NA	0.1Hz-60.0Hz
S/W Filtering	1.0Hz-12.0Hz	1.0Hz-12.0Hz

[1] In U. Hoffmann et al. dataset 4 electrode configurations were used in offline analysis.4, 8, 16, and 32 electrodes. In this thesis we are considering 8 electrodes only.

Figure 4-1 BCI lab at KAUH

4.2 Experimental Setup

Users were facing a screen on which 6 Arabic characters were displayed. The characters were arranged in a 3 by 2 matrix, see Figure 4-2. The user's task was to focus attention (i.e., count silently) on one character at a time in a word that was prescribed by the investigator. Table 4-2 illustrates the prescribed words. All rows and columns of this matrix were flashed in random sequences, one row/column at a time. Two out of 5 flashes of rows or columns contained the desired character (i.e., one particular row and one particular column).

Figure 4-2 this figure illustrates the user display for our experiment

For each character, user display was as follows: the matrix was displayed for a 2.5s period, and during this time the matrix was blank. Subsequently, each row and column in the matrix was randomly intensified for 100ms (i.e., resulting in 5 different stimuli, 3 rows and 2 columns). After intensification of a row/column, the matrix was blank for 200ms. Row/column intensifications were block randomized in blocks of 5. The sets of 5 intensifications were repeated 20 times for each character (i.e., any specific row/column was intensified 20 times and thus there were 100 total intensifications for each character). Each character was followed by a 2.5s period, and during this time the matrix was blank. This period informed the user that this character was completed and to focus on the next character in the word that was displayed on the top of the screen (the current character was shown in parentheses).

Table 4-2 Words to spell in each session

Session	**1**	**2**	**3**	**4**
Word to spell	**أبتثجح**	**أبتثجح**	**أبتثجح**	**أبتثجح**

The experiments were designed and recorded with BCI2000. The offline analyses were implemented with MATLAB. The experiments were done at BCI lab in King Abdul-Aziz University Hospital (KAUH).

4.3 EEG Data Acquisition

The data for this experiment were collected from four normal males (26± 4.5 years).

The EEG was recorded at 256 Hz sampling rate, with band pass filter from 0.1-60 Hz, and the notch filter was set on at 60Hz. The EEG was recorded using eight electrodes placed at the standard positions of the 10-20 international system. The selected electrodes were F3, F4, C3, Cz, C4, P3, Pz and P4 with AFz as ground and right ear lobe as reference, see Figure 4-3.

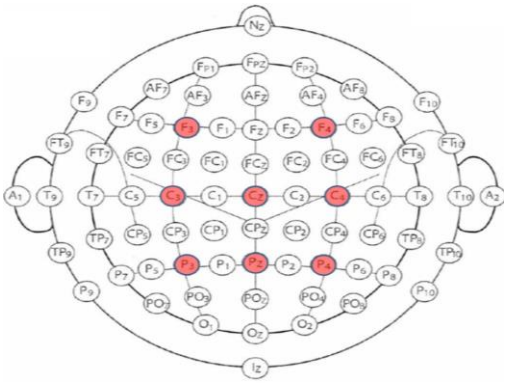

Figure 4-3 Selected electrodes for our experiment

The recording system consists of the following components: g.tec EEGcap, 8 Ag/AgCl electrodes, g.tec GAMMAbox, g.tec USBamp and BCI2000 [53, 53] see Figure 4-4.

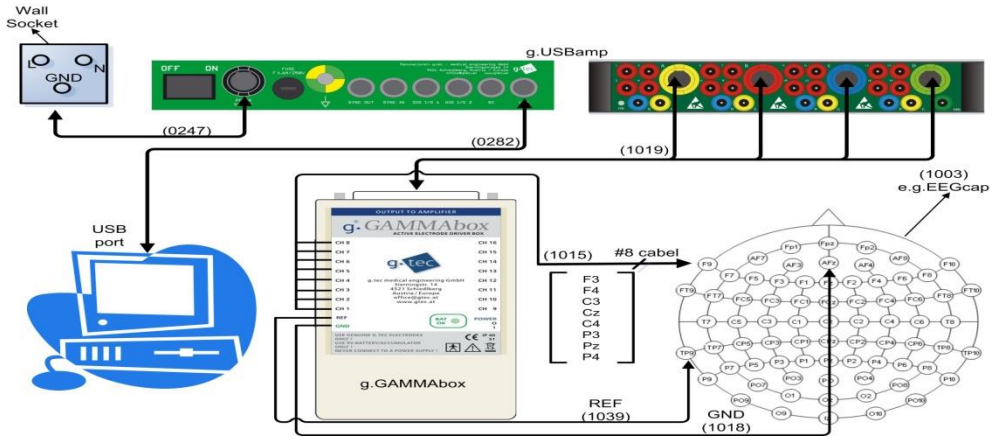

Figure 4-4 Simple 8 channels hardware diagram

4.4 Offline Analysis

The impact of feature selection algorithms on P300 based BCI performance was tested in an offline procedure. For each subject four-fold cross-validation was used to estimate average classification accuracy. More specifically, the data from three recording sessions were used to train a classifier and the data from the left-out session was used for validation. This procedure was repeated four times so each session served once for validation.

4.4.1 Pre-processing

Before learning a classification function and before validation, several pre-processing operations were applied to the data. The main objective of this phase is to enhance the signal to noise ratio SNR. The pre-processing operations were applied in the order stated below.

4.4.1.1 Referencing

During this phase twelve different re-reference techniques were applied. Their results were compared with each other. The results showed that Common Average Reference (CAR) is best suited to be the reference technique. The twelve different re-reference techniques are listed below:

1. Common Reference: No re-montaging is done
2. Common average reference: The mean of all the electrodes is removed for all the electrodes .
3. Laplacian (4 adjacent): The weighted mean (depends on the distance) of the 4 adjacent electrodes is removed from the central electrode.
4. Surface Laplacian (8 adjacent): The weighted mean (depends on the distance) of the 8 surrounding electrodes is removed from the central electrode.
5. Bipolar (front to back): The difference of an electrode with the one behind it .
6. Bipolar (front to back skip 1): The difference of 2 electrodes that lies in front and also behind that electrode.
7. Bipolar (Symmetrical): The difference of 2 electrodes that is symmetrical to one another.

8. Bipolar (left to right): The difference of an electrode with the one right to it.
9. Bipolar (right to left): The difference of an electrode with the one left to it.
10. Using T7,T8 channels : The mean of T7,T8 channels is removed for all the electrodes .
11. Common average reference without mastoid channels: The mean of all the electrodes without mastoid channels is removed for all the electrodes .
12. Reference estimation: Here we apply the work of *R. Ranta et al.* [55] by the estimation of the reference $\hat{r}$.

The findings of this phase were published as a journal article[2].

4.4.1.2 Filtering

A 6th order forward-backward Butterworth bandpass filter was used to filter the data. Cutoff frequencies were set to 1.0 Hz and 12.0 Hz. The MATLAB function butter was used to compute the filter coefficients and the function filtfilt was used for filtering.

4.4.1.3 Downsampling

The EEG was downsampled from 256 Hz to 32 Hz by selecting each 8th sample from the bandpass-filtered data.

4.4.1.4 Single Trial Extraction

Single trials of duration 1000 ms were extracted from the data. Single trials started at stimulus onset, i.e. at the beginning of the flash of raw\column, and ended 1000 ms after stimulus onset.

4.4.1.5 Winsorising

Winsorising or Winsorization is the transformation of statistics by limiting extreme values in the statistical data to reduce the effect of possibly spurious outliers. It is named after Charles P. Winsor (1895–1951). The effect is the same as clipping in signal processing.

[2] Mohammed J. Alhaddad, Mahmoud Kamel, Hussein Malibary, Khalid Thabit, Foud Dahlwi, and **Anas Hadi** "*P300 Speller Efficiency with Common Average Reference*", AIS 2012, LNCS 7326, pp. 234–241, 2012

Eye blinks, eye movement, muscle activity, or subject movement can cause large amplitude outliers in the EEG. To reduce the effects of such outliers, the data from each electrode were clipped. For the samples from each electrode the 10th percentile and the 90th percentile were computed. Amplitude values lying below the 10th percentile or above the 90th percentile were then replaced by the 10th percentile or the 90th percentile, respectively.

4.4.1.6 Normalization

The samples from all electrodes were scaled to the interval [−1, 1]. The normalization was done using z-score method [1].

4.4.1.7 Feature Vector Construction

The samples from the electrodes were concatenated into feature vectors. The dimensionality of the feature vectors was Ne × Ns × Nt, where Ne denotes the number of electrodes, Ns denotes the number of temporal samples in one trial, and Nt denotes the number of trials. Due to the trial duration of 1000 ms and the downsampling to 32 Hz, Ns always equaled 32. Depending on the electrode configuration, Ne equaled 8. Nt is varying according the number of trials in each session.

After that the feature vectors is converted form three dimensions to two dimensions by concatenating the electrodes to each other. The dimensionality of the new feature vectors was Nes × Nt, where Nes equaled (8 × 32 = 256) and Nt is as is with no change.

4.4.2 Feature Selection

Analysis of EEG signals is challenging due to the curse-of dimensionality, which states that an enormous number of samples are required to perform accurate predictions for present model with a high dimensionality. Dimensionality reduction, which extracts a small number of features by removing irrelevant, redundant, and noisy information, can be an effective solution. As stated previously, there are two ways to reduce the

dimensionality and avoid the curse of dimensionality, one of them is ***feature selection*** and the other is ***feature extraction*** [11]. Feature extraction as a dimensionality reduction method is out the scope of this thesis.

The main objective of feature selection phase is to improve the performance of the system. This is done by selecting the most appropriate feature among the other features. Before applying any feature selection algorithm, I have manually reduced the length of trials from 1000ms to 800ms and this is due the fact that P300 appears in this interval and the rest 200ms are not significant.

Here I will explain the three types of feature selection algorithms. These three types are filter, wrapper, and hybrid.

4.4.2.1 Filter

In filter the evaluation functions were Fisher score, Determination Coefficient (r^2), BLDA, and RFLD. Each feature was evaluated individually, after that all features were sorted according to their evaluation from higher to lower. Finally, the optimum number of features **α** was selected. Figure 4-5 illustrates the applied algorithm. The optimum **α** is discussed in Section 5.2.1.

```
Filter Algorithm
Input: D(F1, F2, ..., Fn) // a training data set with N features
Output: FList // the selected list of features

begin
  for i = 1  to N
    Fw(i)= eval(D(Fi),M); // evaluate Fi by an evaluation function M
  end
  Fw=sort(Fw);
  FList = Fw(1:α)
  return FList;
end;
```

Figure 4-5 Applied Filter Algorithm

4.4.2.2 Wrapper

In wrapper the classifiers themselves (BLDA, RFLD) were used as evaluation functions. And the differential evolution was used as a searching algorithm. In order to apply differential evolution algorithm, the function was modified to deal with boolean instead of real numbers. This modification has been introduced before in [48, 41, 64]. The modification was done by setting the upper and lower pounds to 1 and 0 respectively, where 1 means that the feature is selected while 0 means not selected. Another modification was to round the result of the differential function result to be boolean rather than real. The last modification was to keep the searching agents with no repetition. This was done during the generating procedure. The modified algorithm was tested first with toy examples to insure the performance of the algorithm. After that it was applied to our data. Figure 4-6 illustrates the applied wrapper algorithm. First, *S* is initialized randomly with 50 boolean agents one of them is the full set. The length of each agent is *N* where *N* is number of features 256. Each agent is evaluated by the selected classifier and we keep the best one. After that new agent y_i is generated according to:

$$y_i = \begin{cases} a_i + F(b_i - c_i) & , \quad r_i < CR \text{ or } i = i_{rand} \\ S_i & , \quad else \end{cases}$$

Where a_i, b_i, *and* c_i are three distinct agents picked randomly from S, *F* is the differential weight, and *CR* is the crossover probability. According to [32] *CR* was set to 0.9 and *F* was set to 0.5. The new generated agent y_i is rounded to be boolean and was checked to see if y_i is already in *S*. If so another agent is generated. After that y_i is evaluated using the selected classifier and compared with the best one. If it was better we keep it as the best one and update *S* to include y_i. Else, we move on to generate another agent y_i. The stopping criteria were set to be one of two, the first is to reach the fitness of 0.9% and according to our experiments and due to the evaluation of single trials this criterion is unreachable. The other stopping criterion is to reach a predefined number of iterations. Maximum number of iterations was set to 10,000 [32, 65]. That's mean that we are going to do 10,000 evaluations. Each evaluation function were done through cross validation with k-fold=5.

```
Wrapper Algorithm
Input: D(F1, F2, ..., Fn) // a training data set with N features
Output: FList // the selected list of features

Begin
  S= generate(50); // initialize by generating 50 random points one of
                   them is the full set
  for i = 1  to N
    Fw(i)= eval(D(Si),A); // evaluate Si by a classifier A
  end
  [Fbest , Wbest] =max(Fw);
  for j = 1  to 10,000
    Fnew= generate(S);
    Wnew=eval(D(Si),M);
    if (Wnew is better than Wbest)
      Fbest = Fnew;
      Wbes = Wnewt;
      S= update(S,Fnew);
    end
  End
  Flist = Fbest
  return FList;
end;
```

Figure 4-6 Applied wrapper algorithm

Formulation of Evaluation Function

In the literature, evaluation function or fitness function were formulated in different ways. The evaluation function was based on the accuracy only as in [48]. In [69] they calculate sensitivity of the classifier, specificity of the classifier and combine both measurements to get an overall estimate of fitness. In [41, 64] fitness function was based on both accuracy and features in a weighted formula. The fitness function in [41] was:

$$fitness = wa * accuracy + wf/ features$$

Where *wa* and *wf* refer to accuracy weight and features weight respectively. In [64] the fitness function in was:

$$fitness = accuracy - wf* (nr\_features/ nr\_all\_features)$$

We note that the fitness function used in [41] missed the normalization factor which applied in [64]. On the other hand the lower limit for the fitness function used in [64] is *wf*. In our work we add both normalization factor and the range of the fitness function is [1,0] as:

$$fitness = wa * accuracy + wf * (1 - nr\_features / nr\_all\_features)$$

The values of *wa* and *wf* were fix to 0.8 and 0.2 respectively as in [41].

4.4.2.3 Hybrid

To take advantage of the above two models the filter is applied first to reduce the features. And then we apply the wrapper. Figure 4-7 illustrates the applied hybrid algorithm.

```
Hybrid Algorithm
Input: D(F1, F2, ..., Fn) // a training data set with N features
Output: FList // the selected list of features

Begin
  for i = 1  to N
    Fw(i)= eval(D(Fi),M); // evaluate Fi by an evaluation function M
  end
  Fw=sort(Fw);
  F = Fw(1: α)
  S= generate(50); // initialize by generating 50 random points one of
                   them is the full set
  for i = 1  to N
    Fw(i)= eval(D(Si),A); // evaluate Si by a classifier A
  end
  [Fbest , Wbest] =max(Fw);
  for j = 1  to 50xN
    Fnew= generate(S);
    Wnew=eval(D(Si),M);
    if (Wnew is better than Wbest)
      Fbest = Fnew;
      Wbes = Wnewt;
      S= update(S,Fnew);
    end
  End
  Flist = Fbest
  return FList;
end;
```

Figure 4-7 Applied hybrid algorithm

The output of feature selection phase is a reduced feature list which is going to be used in farther processing. Table 4-3 summarized the used algorithms with each feature selection type.

Table 4-3 Used algorithms with each feature selection type

Feature Selection Method	
Filter	Fisher score
	Determination Coefficient(r^2)
	BLDA
	RFLD
Wrapper	BLDA
	RFLD
Hybrid	Fisher score + BLDA
	Fisher score + RFLD
	BLDA + BLDA
	RFLD + RFLD
	r^2 + BLDA
	r^2 + RFLD

4.4.3 Machine Learning and Classification

Used classifiers were trained on the data from three sessions and validated on the left-out session. BLDA and RFLD were used as classifiers. Both algorithms were fully automatic, i.e. no user intervention was required to adjust hyper parameters. After the classifiers had been trained, they were applied to validation data in the following way. For each character in the validation session, the single trials corresponding to the first twenty blocks of flashes were extracted using the preprocessing operations. Then the single trials were classified. This resulted in twenty blocks of classifier outputs. Each block consisted of twelve classifier outputs, one output for each raw\column on the display. To decide which character the user was concentrating on, the classifier outputs were summed over blocks for each raw\column and then the raw\column with the maximum summed classifier output was selected. And the intersection between the selected raw and column identify the desired character. This method of summing the scores of single trials is called *fixed number of blocks*. That means that the output of the classifier is given by the

end of 20 blocks. Another method is called *dynamic number of blocks*. In this method the system stops as soon as enough data has been acquired. Enough data here means that the system can take a reliable decision. More about the later method could be found in [62].

4.4.3.1 Performance Measures

In order to test the effect of adding feature selection module to P300 based BCI. The results are compared using four criteria: Accuracy rates, feature reduction rate, feature selection running time, and training/testing running time.

Accuracy rates: Accuracy rates were obtained using *Classification Accuracy Graphs (CAG)* and *Per Block Accuracy (PBA)* for both BLDA and RFLD. The results were averaged over all subjects and sessions. CAG and PBA were used and declared in [1] as the following: Classification accuracy graphs illustrate the dependence of classification accuracy on the amount of aggregated data. Hence the graph is supposed to be better towered the time. The PBA is computed from all blocks of EEG trials seen during cross-validation and hence no aggregating is used. More discussion about CAG and PBA can be found in [1].

Feature reduction rate: was the ratio between the resulted feature size and the origin feature size. The origin feature size is always 256.

Feature selection running time: was the time in seconds needed for the algorithm to be preformed.

Training/testing running time: was the summation of the time needed for the classifier to be trained – without feature selection time- and the time needed for the classifier to classify the given data. The improvement in running time was the ratio of the training/testing running time before and after applying feature selection algorithms.

Chapter 5 Results and Discussion

5.1 Introduction

This chapter addresses the impact of feature selection module in P300 based BCI. As mentioned before, two datasets are used. One of them is from our BCI lab at KAUH, and the other from U. Hoffmann et al. [1]. The results are compared using four criteria: Accuracy rates, feature reduction rate, feature selection running time, and training/testing running time.

All results were calculated using desktop computer with Intel core i7 processor at 2.93GHz and 4GB RAM, running Microsoft Windows 7 professional, 32-bit operating system.

5.2 Experimental Results with U. Hoffmann et al. Dataset

5.2.1 Results using filter

As previously discussed in section 4.4.2.1 in applying filter that we have to determine the optimum number of selected features (**α**)**.** The optimum number of selected features was adjusted for each type of filter algorithm. This was done by plotting the accuracy curve against the selected features and determined the threshold were the accuracy were not affected by the increasing of features. This method was described in [29]. In Figure 5-1 we can see the accuracy curves using fisher score plotted against the selected features. These curves were obtained using U. Hoffmann et al. [1] dataset for both BLDA (blue curve) and RFLD (red curve). And it was averaged over all subjects and sessions. The red dashed line determine the optimum number of selected features = 90. It's clear that the rest of features are redundant and can be removed without significant effect in the accuracy.

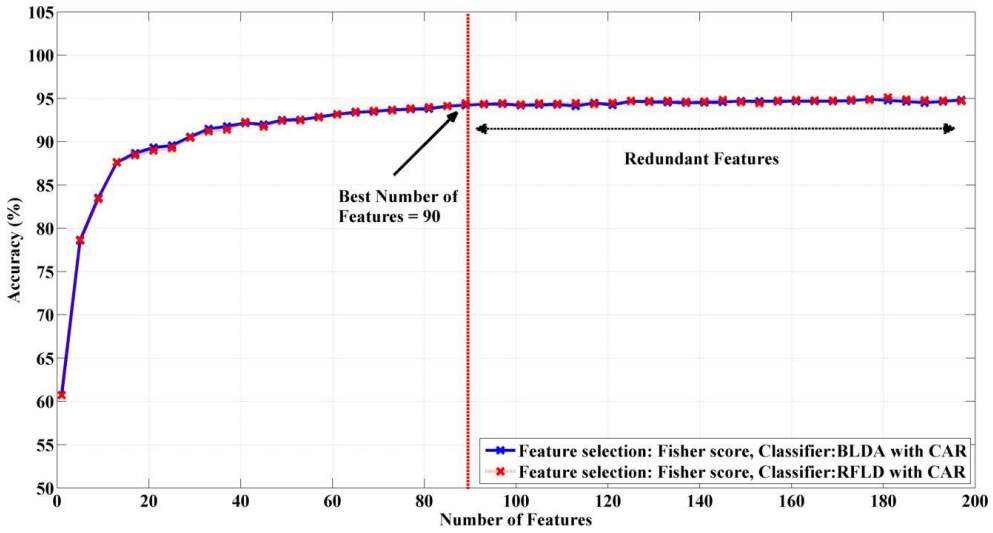

Figure 5-1 Selecting the optimum number of features using fisher score for U. Hoffmann et al. dataset averaged over all subjects and sessions

This method was repeated for each filter types: r^2as in Figure 5-2, BLDA as in figure 5-3, and RFLD as in figure 5-4. The optimum numbers of selected features were 90, 100, and 100 for r^2, BLDA, and RFLD respectively. Although higher accuracies can be obtained, the previous ones were selected due to the significant decrease in feature size without significant increase in the accuracy.

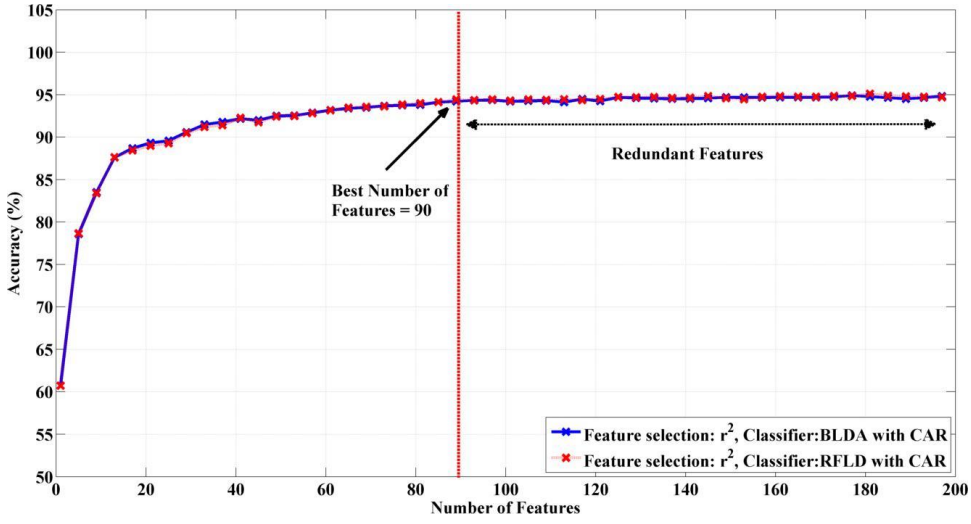

Figure 5-2 Selecting the optimum number of features using r^2 for U. Hoffmann et al. dataset averaged over all subjects and sessions

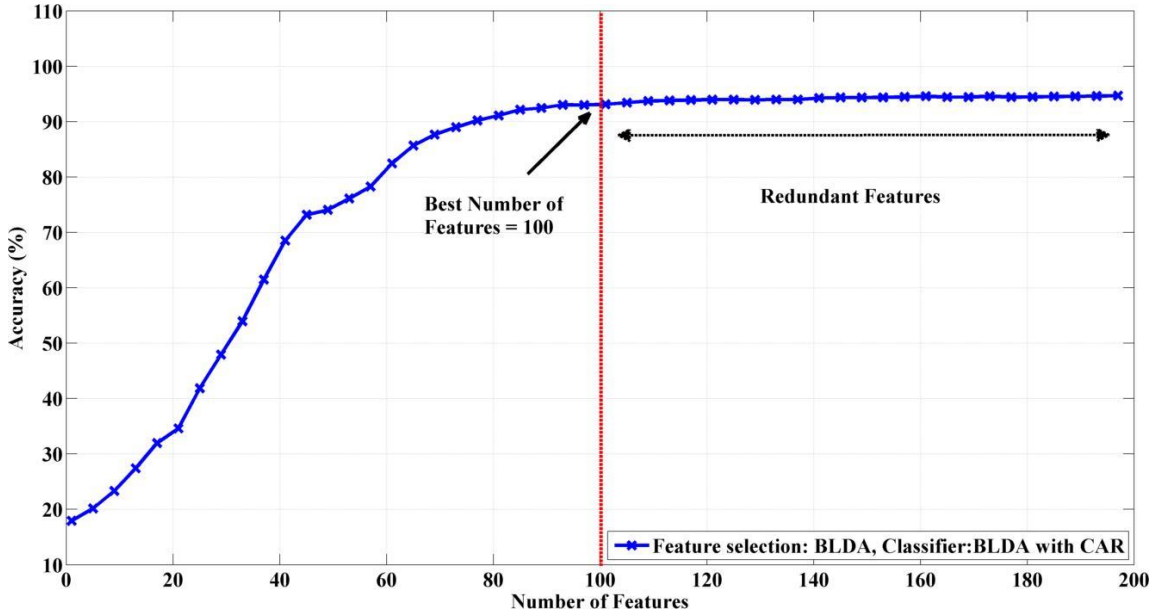

Figure 5-3 Selecting the optimum number of features using BLDA for U. Hoffmann et al. dataset averaged over all subjects and sessions

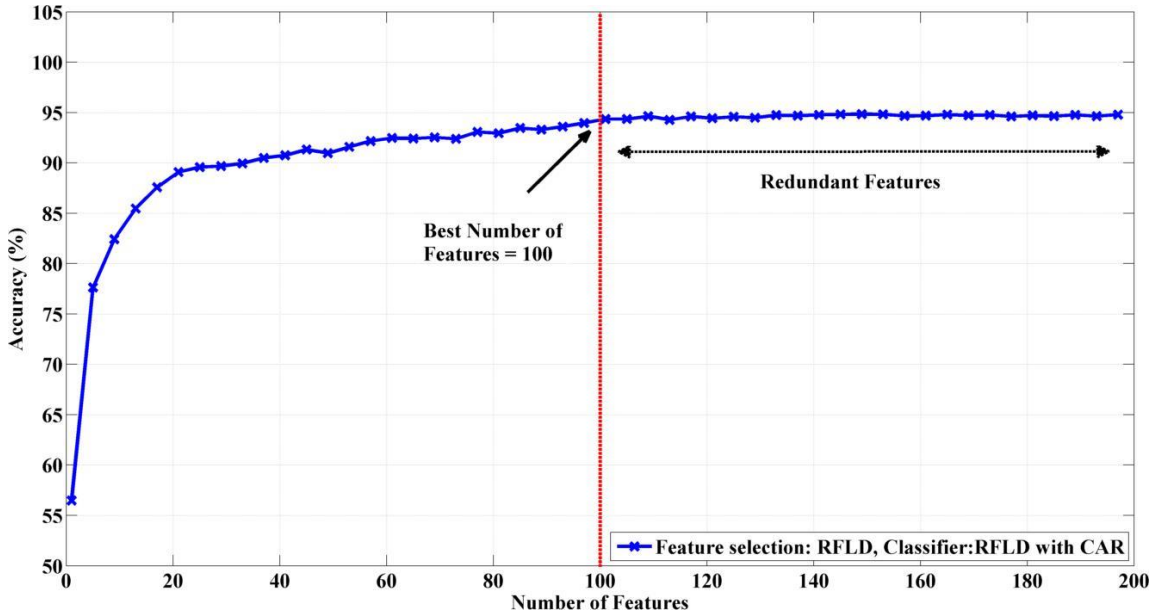

Figure 5-4 Selecting the optimum number of features using RFLD for U. Hoffmann et al dataset averaged over all subjects and sessions

In Figure 5-5 we can see an accuracy based comparison between filter algorithms when applied for both BLDA and RFLD. These curves were the average over all subjects and sessions, plotted against number of repetitions. The curve obtained by U. Hoffmann et al. [1] was also plotted for comparison purpose and was denoted by T7, T8.

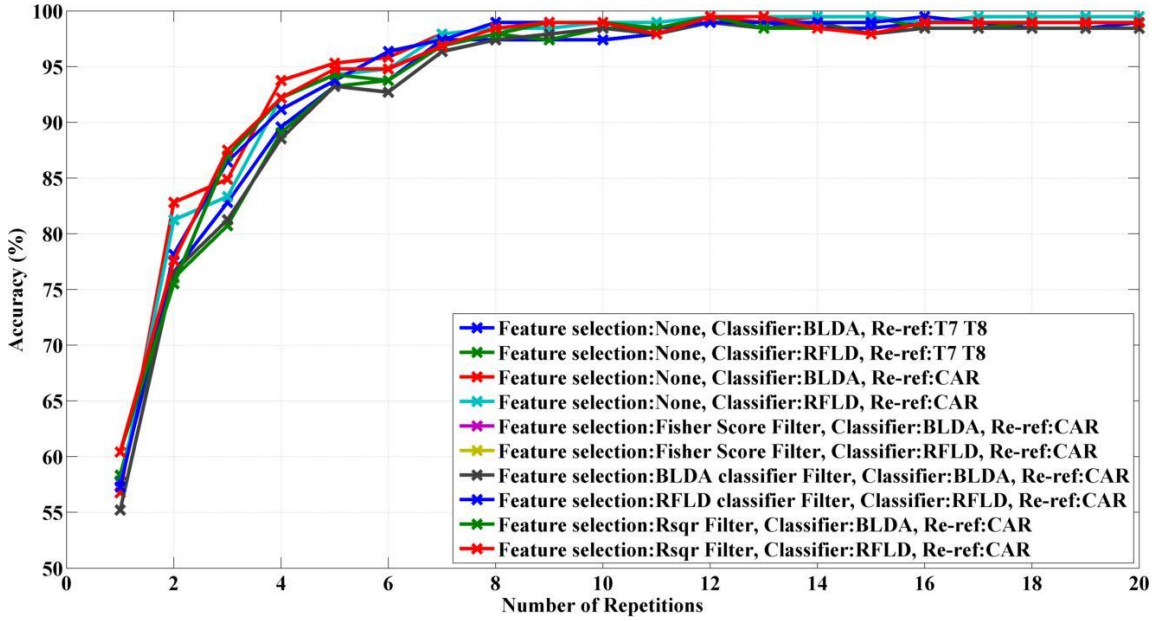

Figure 5-5 Filter results for U. Hoffmann et al dataset

5.2.1.1 Classification Accuracy Graphs CAG

Table 5-1 illustrates CAG comparison for filter types. The comparison is based on four criteria: classification correct rates, feature reduction rate, feature selection running time, and training/testing running time. The classification correct rates were averaged cross the number of repetitions. We can see that the average correct rate was not significantly affected. On the other hand, features sizes were reduced by 64.8% for both fisher score and r^2. Features sizes for BLDA and RFLD were 60.9%. This reduction in features improves the running time of the classifiers during training and testing. For BLDA the reduction was 77.4%, 82.7%, and 84.9% for *fisher score*, *BLDA*, and r^2 respectively. For RFLD the reduction was 79.7%, 85.3%, and 85.8% for *fisher score*, *RFLD*, and r^2 respectively. The algorithm running time was varying, where $r^2$was the fastest one with 7.8ms for BLDA and 7.59ms for RFLD. Fisher score was slower with 38.2ms for BLDA and 37.9ms for RFLD. BLDA and RFLD were the slowest with 1460ms for BLDA and 8350ms for RFLD. In general, RFLD is slower than BLDA, and this is due to

computations needed for the regularization. Finally we can see the improvement of using CAR as a re-reference technique instead of T7, T8 electrodes[3] in the first four rows.

Table 5-1 Filter results using CAG for U. Hoffmann et al dataset

Feature Selection Method	Classifier	Correct rate% (mean ± S.D)	Feature Reduction percentage %	Running Time (s)		
				Feature selection	Training + Testing	Improvement %
None	BLDA T7_T8	93.49 ± 10.3	0.0%	0	7.16E-02	-1.0%
None	RFLD T7_T8	93.54 ± 10.4	0.0%	0	1.19E+00	-0.8%
None	BLDA	94.71 ± 10.1	0.0%	0	7.09E-02	0.0%
None	RFLD	94.53 ± 10.2	0.0%	0	1.18E+00	0.0%
Fisher score	BLDA	94.17 ± 9.8	64.8%	3.82E-02	1.60E-02	77.4%
Fisher score	RFLD	94.43 ± 9.6	64.8%	3.79E-02	2.40E-01	79.7%
BLDA	BLDA	93.1 ± 10.85	60.9%	1.46E+00	1.23E-02	82.7%
RFLD	RFLD	94.35 ± 10.3	60.9%	8.35E+00	1.73E-01	85.3%
Determination Coefficient(r^2)	BLDA	94.17 ± 9.8	64.8%	7.85E-03	1.07E-02	84.9%
Determination Coefficient(r^2)	RFLD	94.43 ± 9.6	64.8%	7.59E-03	1.67E-01	85.8%

5.2.1.2 Per Block Accuracy PBA

Tables 5-2 illustrate PBA comparison for filter types. We can see that the average correct rate was not significantly affected –max 3%-. On the other hand, features sizes were reduced by 64.8% for both fisher score and r^2. Features sizes for BLDA and RFLD were 60.9%. This reduction in features improves the running time of the classifiers during training and testing. For BLDA the reduction was 76.1%, 75.0%, and 72.9% for *fisher score*, *BLDA*, and r^2 respectively. For RFLD the reduction was 85.7%, 87.2%, and 85.7% for *fisher score*, *RFLD*, and r^2 respectively. The algorithm running time was varying, where $r^2$was the fastest one with 7.74ms for BLDA and 8.07ms for RFLD. Fisher score was slower with 30.0ms for BLDA and 29.6ms for RFLD. BLDA and RFLD were the slowest with 1370ms for BLDA and 8070ms for RFLD. Finally we can see the improvement of using CAR as a re-reference technique instead of T7, T8 electrodes in the first four rows.

[3] Mohammed J. Alhaddad, Mahmoud Kamel, Hussein Malibary, Khalid Thabit, Foud Dahlwi, and Anas Hadi "*P300 Speller Efficiency with Common Average Reference*", AIS 2012, LNCS 7326, pp. 234–241, 2012.

Table 5-2 Filter results using PBA for U. Hoffmann et al dataset

Feature Selection Method	Classifier	Correct rate% (mean ± S.D)	Feature Reduction percentage %	Running Time (s)		
				Feature selection	Training + Testing	Improvement %
None	BLDA T7_T8	60.36 ± 10.96	0.0%	4.67E-04	8.90E-02	0.0%
None	RFLD T7_T8	60.03 ± 11.5	0.0%	1.59E-04	1.22E+00	0.0%
None	BLDA	61.07 ± 11.27	0.0%	1.52E-04	8.91E-02	-0.2%
None	RFLD	61.02 ± 11.53	0.0%	1.58E-04	1.18E+00	3.6%
Fisher score	BLDA	58.23 ± 11.63	64.8%	3.00E-02	2.13E-02	76.1%
Fisher score	RFLD	58.39 ± 11.31	64.8%	2.96E-02	1.75E-01	85.7%
BLDA	BLDA	57.42 ± 11.49	60.9%	1.37E+00	2.23E-02	75.0%
RFLD	RFLD	58.65 ± 11.1	60.9%	8.10E+00	1.56E-01	87.2%
Determination Coefficient(r^2)	BLDA	58.23 ± 11.63	64.8%	8.07E-03	2.41E-02	72.9%
Determination Coefficient(r^2)	RFLD	58.39 ± 11.31	64.8%	7.74E-03	1.75E-01	85.7%

5.2.2 Results with wrapper

In Figure 5-6 we can see an accuracy based comparison between Wrapper algorithms when applied for both BLDA and RFLD. These curves were the average over all subjects and sessions, plotted against number of repetitions.

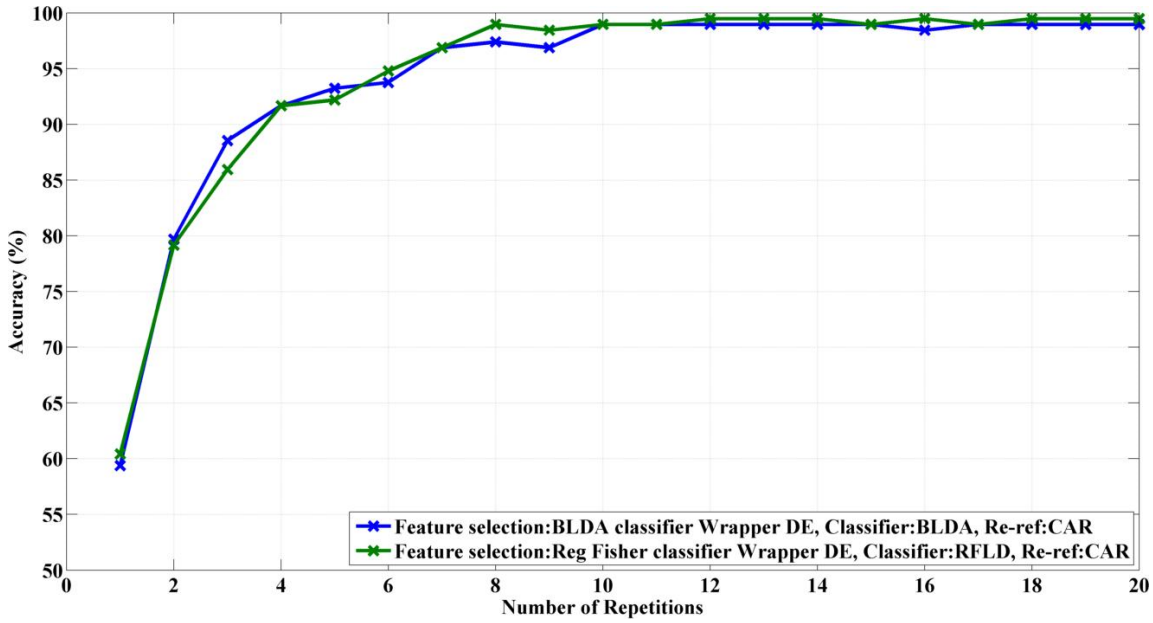

Figure 5-6 Wrapper results for U. Hoffmann dataset

5.2.2.1 Classification Accuracy Graphs CAG

Tables 5-3 illustrate CAG comparison for wrapper types. For BLDA, we can see that features sizes were reduced by 66.4%, 63.9% for BLDA and RFLD respectively. The reduction in features improves the running time of the classifiers during training and testing. For BLDA the reduction was 60.5% while for RFLD the reduction was 78.5%. The algorithm running time was the highest when compare with filter or hybrid methods. BLDA feature selection running time was 1,890s. On the other hand, RFLD feature selection running time was 13,000s which is about 3.6 hours. No significant change in accuracy was happened.

Table 5-3 Wrapper results using CAG for U. Hoffmann dataset

Feature Selection Method	Classifier	Correct rate% (mean ± S.D)	Feature Reduction percentage %	Running Time (s)		
				Feature Selection	Training + Testing	Improvement %
BLDA	BLDA	93.96 ± 10.48	66.4%	1.89E+03	4.08E-02	60.5%
RFLD	RFLD	94.06 ± 9.81	63.9%	1.30E+04	2.49E-01	78.5%

5.2.2.2 Per Block Accuracy PBA

Tables 5-4 illustrate PBA comparison for wrapper types. For BLDA, we can see that features sizes were reduced by 66.0%, 64.0% for BLDA and RFLD respectively. The reduction in features improves the running time of the classifiers during training and testing. For BLDA the reduction was 75.8% while for RFLD the reduction was 87.9%. BLDA feature selection running time was 846s. On the other hand, RFLD feature selection running time was 7,630s which is about 2.12 hours. No significant change in accuracy was happened –max 3%-.

Table 5-4 Wrapper results using PBA for U. Hoffmann dataset

Feature Selection Method	Classifier	Correct rate% (mean ± S.D)	Feature Reduction percentage %	Running Time (s)		
				Feature Selection	Training + Testing	Improvement %
BLDA	BLDA	57.6 ± 11.74	66.0%	8.46E+02	2.15E-02	75.8%
RFLD	RFLD	58.1 ± 10.79	64.0%	7.63E+03	1.48E-01	87.9%

5.2.3 Results with hybrid

In Figure 5-7 we can see an accuracy based comparison between Hybrid algorithms when applied for both BLDA and RFLD. These curves were the average over all subjects and sessions, plotted against number of repetitions.

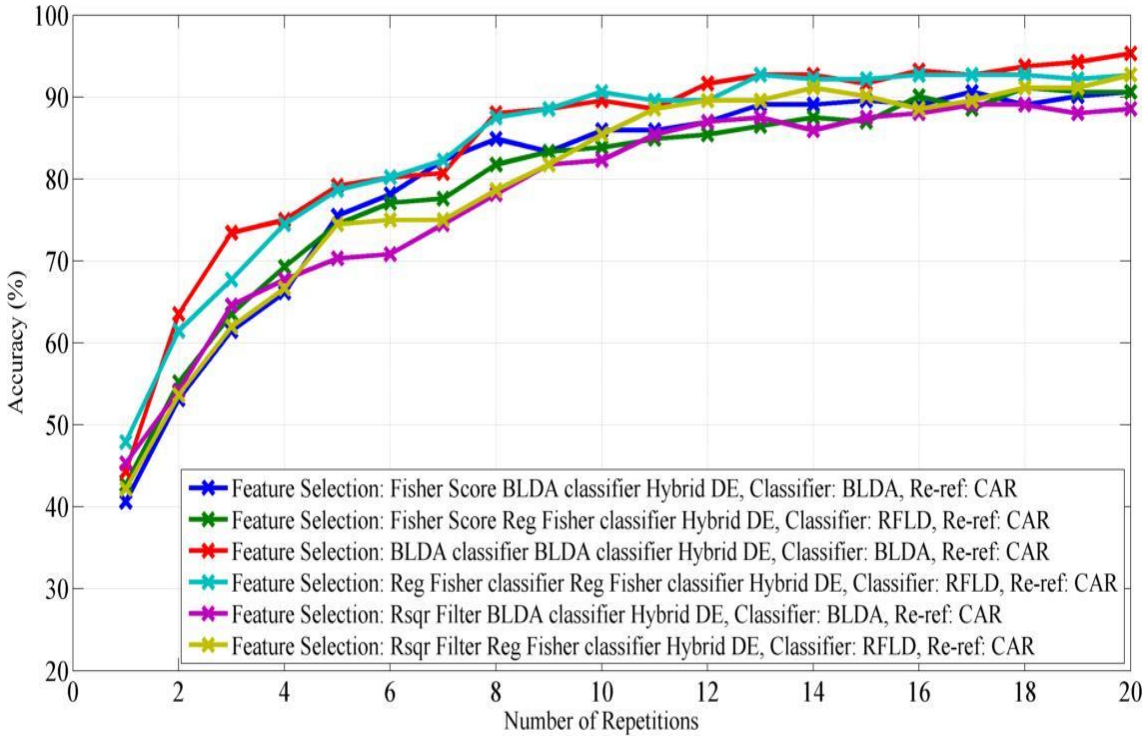

Figure 5-7 Hybrid results for U. Hoffmann et al dataset

5.2.3.1 Classification Accuracy Graphs CAG

Tables 5-5 illustrate CAG comparison for hybrid types For BLDA; we can see that features sizes were reduced by 88.5%, 88.8%, and 86.7% for fisher score, $r^2$and BLDA respectively. For RFLD, features sizes were reduced by 87.7%, 87.9% and 85.7% for fisher score, $r^2$and RFLD respectively. The reduction in features improves the running time of the classifiers during training and testing. For BLDA the reduction was 86.6%, 86.8%, and 87.0% for *fisher score*, r^2, and *BLDA* respectively. For RFLD the reduction was 96.6%, 95.7%, and 96.2% for *fisher score*, r^2, and *RFLD* respectively.

Table 5-5 Hybrid results using CAG for U. Hoffmann et al dataset

Feature Selection Method	Classifier	Correct rate% (mean ± S.D)	Feature Reduction percentage %	Running Time (s)		
				Feature Selection	Training + Testing	Improvement %
Fisher score + BLDA	BLDA	80.08 ± 14.03	88.5%	2.53E+02	1.39E-02	86.6%
Fisher score + RFLD	RFLD	79.56 ± 12.95	87.7%	1.82E+03	3.93E-02	96.6%
BLDA + BLDA	BLDA	84.45 ± 12.73	86.7%	2.69E+02	1.34E-02	87.0%
RFLD + RFLD	RFLD	83.93 ± 12.42	85.7%	2.11E+03	4.37E-02	96.2%
r^2 + BLDA	BLDA	78.28 ± 12.59	88.8%	2.33E+02	1.36E-02	86.8%
r^2 + RFLD	RFLD	79.84 ± 14.14	87.9%	2.16E+03	4.93E-02	95.7%

5.2.3.2 Per Block Accuracy PBA

Tables 5-6 illustrate PBA comparison for hybrid types For BLDA; we can see that features sizes were reduced by 88.5%, 86.7%, and 88.8% for fisher score, $r^2$and BLDA respectively. For RFLD, features sizes were reduced by 87.7%, 87.5% and 85.7% for fisher score, $r^2$and RFLD respectively. The reduction in features improves the running time of the classifiers during training and testing. For BLDA the reduction was 84.8%, 84.7%, and 84.5% for *fisher score*, r^2, and *BLDA* respectively. For RFLD the reduction was 96.8%, 96.3% and 96.6% for *fisher score*, r^2, and *RFLD* respectively.

Table 5-6 Hybrid results using PBA for U. Hoffmann et al dataset

Feature Selection Method	Classifier	Correct rate% (mean ± S.D)	Feature Reduction percentage %	Running Time (s)		
				Feature Selection	Training + Testing	Improvement %
Fisher score + BLDA	BLDA	41.74 ± 8.13	88.5%	2.60E+02	1.36E-02	84.8%
Fisher score + RFLD	RFLD	41.93 ± 8.75	87.7%	1.77E+03	3.97E-02	96.8%
BLDA + BLDA	BLDA	46.04 ± 10.06	86.7%	2.88E+02	1.38E-02	84.5%
RFLD + RFLD	RFLD	44.53 ± 9.65	85.7%	2.14E+03	4.50E-02	96.3%
r^2 + BLDA	BLDA	40.55 ± 9.37	88.8%	2.51E+02	1.37E-02	84.7%
r^2 + RFLD	RFLD	40.91 ± 9.98	87.5%	1.71E+03	4.21E-02	96.6%

The algorithm running time was high in general. For BLDA the feature selection running time was 233s with r^2 using CAG and 251s using PBA. Fisher score was slower with 253s for CAG and 260 for PBA. BLDA was the slowest with 269s for CAG and 288s for PBA. Put this considered to be fast when comparing with RFLD associated algorithms. The fastest one for RFLD was r^2 with 2160s and 1710 for CAG and PBA respectively. Fisher score was 1820s for CAG and 1770 for PBA. And finally, RFLD was the lowest with 2110s for CAG and 2140s for PBA which is about 35.67 minutes. We recall that this time is the average of sessions. Beyond this, the correct rate was significantly affected negatively. The accuracies were reduced by 15%. This can be explained by the concept of *over-fitting*. More discussion about this point will be in section 5.4.3.

5.3 Experimental Results with KAUH Dataset

5.3.1 Results with Filter

There were some differences between our dataset (KAUH) and U. Hoffmann et al. [1] dataset. See Table 5-1. According to this we have to check again for the optimum number of selected features (**α**) during applying filter method. The same methodology was used. In Figures 5-6, 5-7, 5-8, and 5-9 we can see that there was no change in the optimum numbers of the selected features. They were the same for both datasets. 90, 90, 100, and 100 for fisher score, r^2, BLDA, and RFLD respectively.

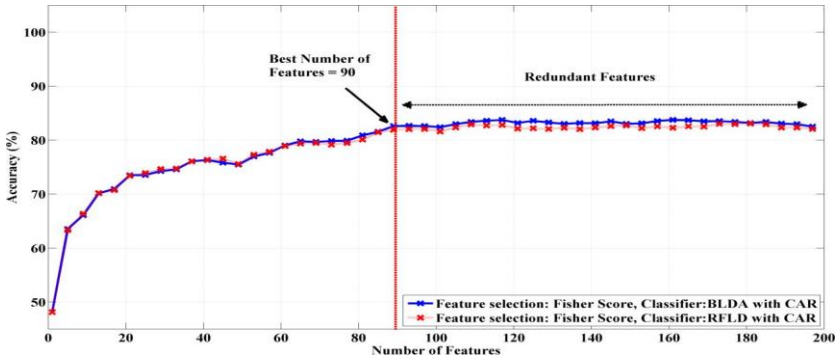

Figure 5-8 Selecting the optimum number of features using fisher score for KAUH dataset averaged over all subjects and sessions

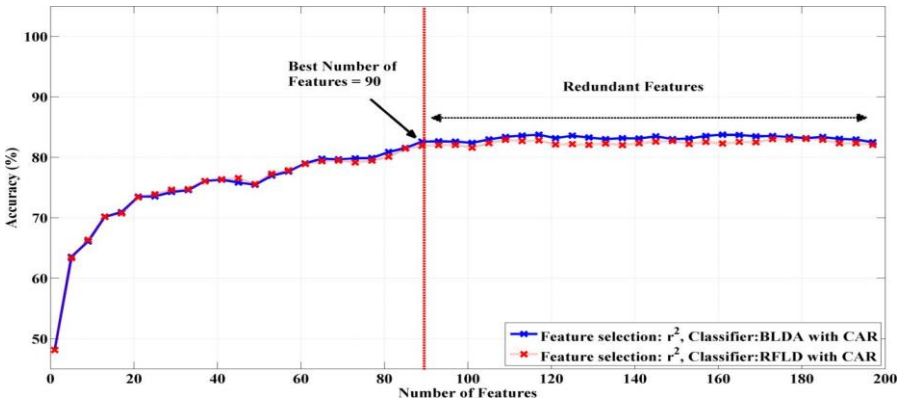

Figure 5-9 Selecting the optimum number of features using r2 for KAUH dataset averaged over all subjects and sessions

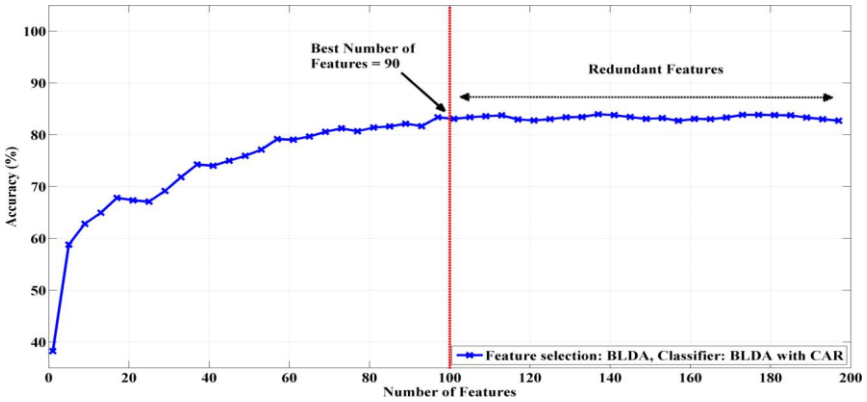

Figure 5-10 Selecting the optimum number of features using BLDA for KAUH dataset averaged over all subjects and sessions

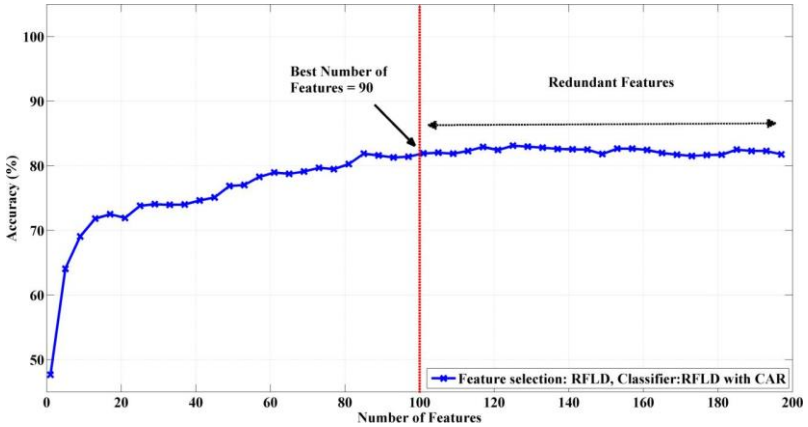

Figure 5-11 Selecting the optimum number of features using RFLD for KAUH dataset averaged over all subjects and sessions

In Figure 5-12 we can see an accuracy based comparison between filter algorithms when applied for both BLDA and RFLD. These curves were the average over all subjects and sessions, plotted against number of repetitions.

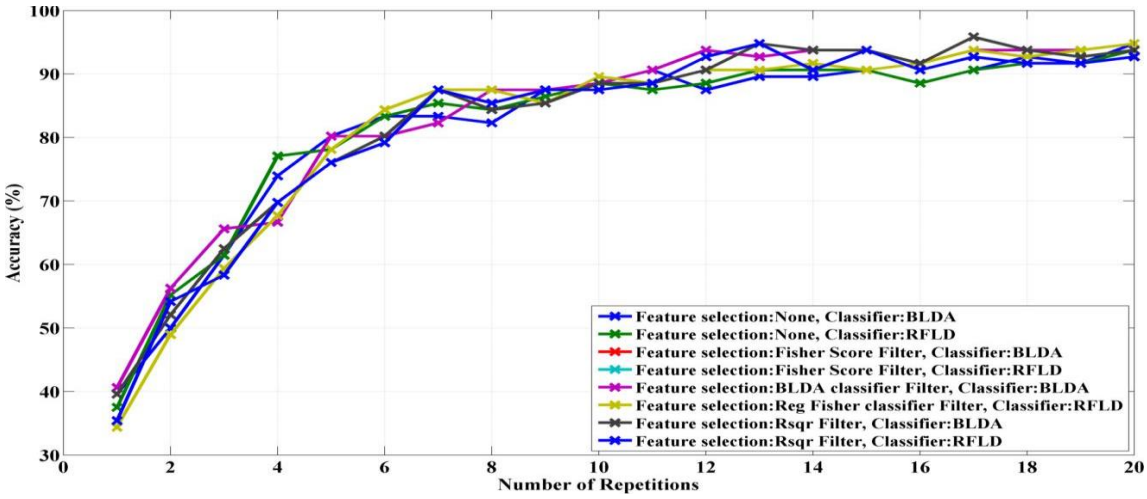

Figure 5-12 Filter results for KAUH dataset

5.3.1.1 Classification Accuracy Graphs CAG

Table 5-7 illustrates CAG comparison using four criteria: classification correct rates, feature reduction rate, feature selection running time, and training/testing running time. The classification correct rates were averaged cross the number of repetitions. We can see that the average correct rate was not significantly affected. On the other hand, features sizes were reduced by 64.8% for both fisher score and r^2. Features sizes for BLDA and RFLD were 60.9%. This reduction in features improves the running time of the classifiers during training and testing. For BLDA the reduction was 85.1%, 83.0%, and 84.7% for *fisher score*, *BLDA*, and r^2 respectively. For RFLD the reduction was 85.7%, 87.5%, and 84.7% for *fisher score*, *RFLD*, and r^2 respectively. The algorithm running time was varying, where $r^2$was the fastest one with 5.93ms for BLDA and 5.83ms for RFLD. Fisher score was slower with 26.4ms for BLDA and 25.9ms for RFLD. BLDA and RFLD were the slowest with 1210ms for BLDA and 6180ms for RFLD. The overall accuracy for KAUH dataset is lower than U. Hoffmann et al. [1] dataset. More discussion about this point will be in section 5.4.2.

Table 5-7 Filter results using CAG for KAUH dataset

Feature Selection Method	Classifier	Correct rate% (mean ± S.D)	Feature Reduction percentage %	Running Time (s)		
				Feature selection	Training + Testing	Improvement %
None	BLDA	81.82 ± 14.82	0.0%	4.77E-05	6.83E-02	0.0%
None	RFLD	82.08 ± 14.47	0.0%	4.20E-05	8.51E-01	0.0%
Fisher score	BLDA	82.76 ± 15.51	64.8%	2.64E-02	1.02E-02	85.1%
Fisher score	RFLD	82.03 ± 15.93	64.8%	2.59E-02	1.22E-01	85.7%
BLDA	BLDA	83.39 ± 14.91	60.9%	1.21E+00	1.16E-02	83.0%
RFLD	RFLD	82.08 ± 16.6	60.9%	6.18E+00	1.07E-01	87.5%
r^2	BLDA	82.76 ± 15.51	64.8%	5.93E-03	1.05E-02	84.7%
r^2	RFLD	82.03 ± 15.93	64.8%	5.83E-03	1.19E-01	86.0%

5.3.1.2 Per Block Accuracy PBA

Table 5-8 illustrates PBA comparison. We can see that the average correct rate was not significantly affected. On the other hand, features sizes were reduced by 64.8% for both fisher score and r^2. Features sizes for BLDA and RFLD were 60.9%. This reduction in features improves the running time of the classifiers during training and testing. For BLDA the reduction was 89.6%, 90.0%, and 89.3% for *fisher score*, *BLDA*, and r^2 respectively. For RFLD the reduction was 87.8%, 88.4%, and 88.7% for *fisher score*, *RFLD*, and r^2 respectively. The algorithm running time was varying, where $r^2$was the fastest one with 9.37ms for BLDA and 9.86ms for RFLD. Fisher score was slower with 38.8ms for BLDA and 35.7ms for RFLD. BLDA and RFLD were the slowest with 1630ms for BLDA and 8230ms for RFLD.

Table 5-8 Filter results using PBA for KAUH dataset

Feature Selection Method	Classifier	Correct rate% (mean ± S.D)	Feature Reduction percentage %	Running Time (s)		
				Feature selection	Training + Testing	Improvement %
None	BLDA	41.67 ± 7.1	0.0%	2.12E-04	1.96E-01	0.0%
None	RFLD	43.54 ± 5.03	0.0%	2.17E-04	1.40E+00	0.0%
Fisher score	BLDA	40 ± 3.94	64.8%	3.88E-02	2.04E-02	89.6%
Fisher score	RFLD	39.22 ± 4.14	64.8%	3.57E-02	1.71E-01	87.8%
BLDA	BLDA	40.21 ± 5.03	60.9%	1.63E+00	1.97E-02	90.0%
RFLD	RFLD	40.83 ± 3.47	60.9%	8.23E+00	1.62E-01	88.4%
r^2	BLDA	40 ± 3.94	64.8%	9.37E-03	2.11E-02	89.3%
r^2	RFLD	39.22 ± 4.14	64.8%	9.86E-03	1.58E-01	88.7%

5.3.2 Results with Wrapper

In Figure 5-13 we can see an accuracy based comparison between Wrapper algorithms when applied for both BLDA and RFLD. These curves were the average over all subjects and sessions, plotted against number of repetitions.

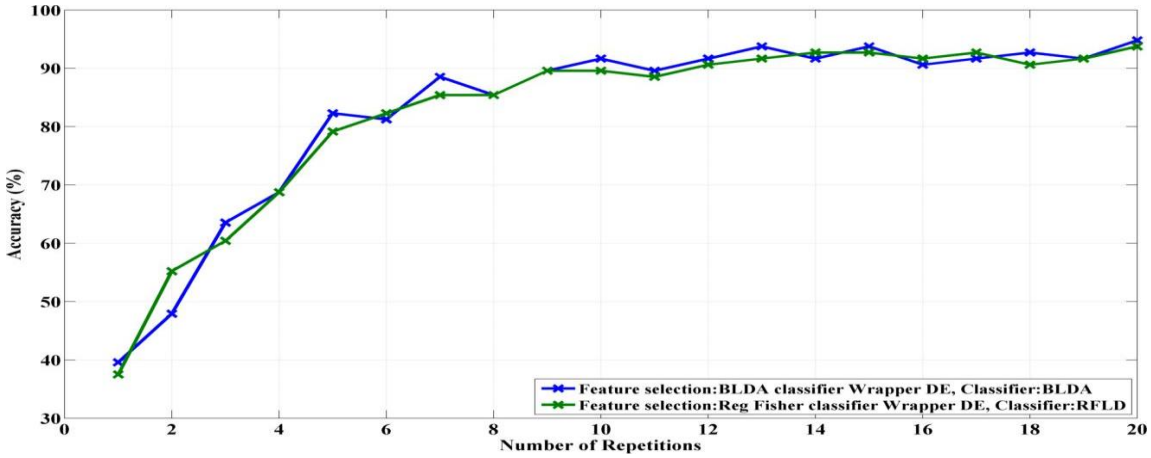

Figure 5-13 Wrapper results for KAUH dataset

5.3.2.1 Classification Accuracy Graphs CAG

Table 5-9 illustrates CAG comparison. We can see that features sizes were reduced by 63.9%, 62.5% for BLDA and RFLD respectively. The reduction in features improves the running time of the classifiers during training and testing. For BLDA the reduction was 83.4% while for RFLD the reduction was 86.9%. The algorithm running time was the highest when compare with filter or hybrid methods. BLDA feature selection running time was 778s. On the other hand, RFLD feature selection running time was 6030s which is about 60 minutes. The change in accuracy was not significant.

Table 5-9 Wrapper results using CAG for KAUH dataset

Feature Selection Method	Classifier	Correct rate% (mean ± S.D)	Feature Reduction percentage %	Running Time (s)		
				Feature Selection	Training + Testing	Improvement %
BLDA	BLDA	82.14 ± 15.55	63.9%	7.78E+02	1.13E-02	83.4%
RFLD	RFLD	82.03 ± 16.82	62.5%	6.03E+03	1.12E-01	86.9%

5.3.2.2 Per Block Accuracy PBA

Table 5-10 illustrates PBA comparison. We can see that features sizes were reduced by 64.1%, 62.3% for BLDA and RFLD respectively. The reduction in features improves the running time of the classifiers during training and testing. For BLDA the reduction was 90.1% while for RFLD the reduction was 88.7%. The algorithm running time was the highest when compare with filter or hybrid methods. BLDA feature selection running time was 1120s. On the other hand, RFLD feature selection running time was 8270s which is about 60 minutes. The change in accuracy was not significant.

Table 5-10 Wrapper results using PBA for KAUH dataset

Feature Selection Method	Classifier	Correct rate% (mean ± S.D)	Feature Reduction percentage %	Running Time (s)		
				Feature Selection	Training + Testing	Improvement %
BLDA	BLDA	40.36 ± 3.88	64.1%	1.12E+03	1.94E-02	90.1%
RFLD	RFLD	40.52 ± 5.69	62.3%	8.27E+03	1.59E-01	88.7%

5.3.3 Results with Hybrid

In Figure 5-14 we can see an accuracy based comparison between Hybrid algorithms when applied for both BLDA and RFLD. These curves were the average over all subjects and sessions, plotted against number of repetitions.

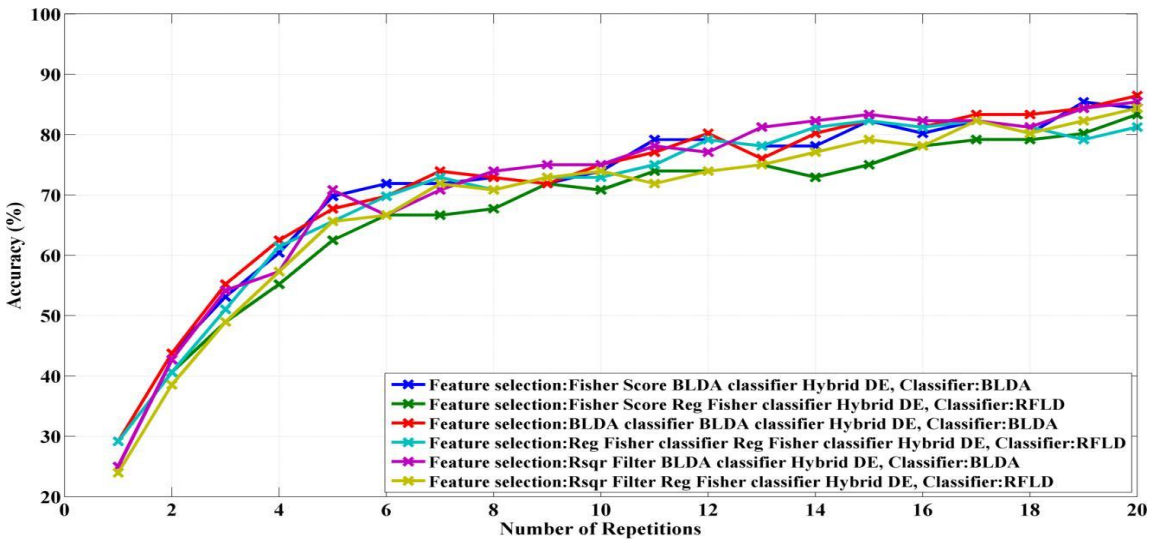

Figure 5-14 Hybrid results for KAUH dataset

5.3.3.1 Classification Accuracy Graphs CAG

Table 5-11 illustrates CAG comparison. For BLDA, we can see that features sizes were reduced by 88.0%, 88.2% and 86.0% for fisher score, $r^2$and BLDA respectively. For RFLD, features sizes were reduced by 88.3%, 87.4% and 85.5% for fisher score, $r^2$and RFLD respectively. The reduction in features improves the running time of the classifiers during training and testing. For BLDA the reduction was 95.3%, 95.7, and 95.3 for *fisher score*, r^2, and *BLDA* respectively. For RFLD the reduction was 97.6%, 97.3%, and 96.9% for *fisher score*, r^2, *RFLD* and respectively.

Table 5-11 Hybrid results using CAG for KAUH dataset

Feature Selection Method	Classifier	Correct rate% (mean ± S.D)	Feature Reduction percentage %	Running Time (s)		
				Feature Selection	Training + Testing	Improvement %
Fisher score + BLDA	BLDA	61.3 ± 15.84	88.0%	1.98E+02	3.23E-03	95.3%
Fisher score + RFLD	RFLD	62.08 ± 14.75	88.3%	1.25E+03	2.01E-02	97.6%
BLDA + BLDA	BLDA	65.68 ± 14.57	86.0%	2.31E+02	3.23E-03	95.3%
RFLD + RFLD	RFLD	61.77 ± 13.14	85.5%	1.63E+03	2.63E-02	96.9%
r^2 + BLDA	BLDA	66.82 ± 14.87	88.2%	1.90E+02	2.95E-03	95.7%
r^2 + RFLD	RFLD	65.36 ± 13.37	87.4%	1.30E+03	2.27E-02	97.3%

5.3.3.2 Per Block Accuracy PBA

Table 5-12 illustrates the comparison. For BLDA, we can see that features sizes were reduced by 88.1%, 88.4% and 85.7% for fisher score, $r^2$and BLDA respectively. For RFLD, features sizes were reduced by 85.6%, 87.8% and 84.8% for fisher score, $r^2$and RFLD respectively. The reduction in features improves the running time of the classifiers during training and testing. For BLDA the reduction was 96.4%, 96.6%, and 95.8% for *fisher score*, r^2, and *BLDA* respectively. For RFLD the reduction was 97.7%, 98.1%, and 97.5% for *fisher score*, r^2, *RFLD* and respectively.

Table 5-12 Hybrid results using PBA for KAUH dataset

Feature Selection Method	Classifier	Correct rate% (mean ± S.D)	Feature Reduction percentage %	Running Time (s)		
				Feature Selection	Training + Testing	Improvement %
Fisher score + BLDA	BLDA	28.49 ± 0.95	88.1%	3.08E+02	7.01E-03	96.4%
Fisher score + RFLD	RFLD	28.96 ± 2.21	85.6%	1.95E+03	3.26E-02	97.7%
BLDA + BLDA	BLDA	27.71 ± 1.46	85.7%	3.74E+02	8.24E-03	95.8%
RFLD + RFLD	RFLD	28.13 ± 2.77	84.8%	2.06E+03	3.47E-02	97.5%
r^2 + BLDA	BLDA	30.83 ± 2.4	88.4%	3.09E+02	6.62E-03	96.6%
r^2 + RFLD	RFLD	31.25 ± 3.63	87.8%	1.53E+03	2.64E-02	98.1%

The algorithm running time was high in general. RFLD associated algorithms tend to take much time than BLDA associated algorithms. For BLDA the feature selection running time was 190s with r^2. Fisher score was slower with 198s. BLDA was the slowest with 231s. Put this considered to be fast when comparing with RFLD associated algorithms. The fastest one for RFLD was Fisher score with 1250s. r^2 was 1300s. And finally, RFLD was the lowest with 1630s which is about 27.17 minutes. Beyond this, again the correct rate was significantly affected negatively. The accuracies were reduced in average by 28%. More discussion about this point will be in section 5.4.3.

5.4 General Discussion

5.4.1 Comparing the three feature selection types

Table 5-13 summarizes the result for used feature selection algorithms averaged from both datasets using CAG.

Table 5-13 Results for both datasets

	Feature Selection Method	Classifier	Correct rate% (mean ± S.D)	Feature Reduction Percentage %	Running Time (s)		
					Feature selection	Training + Testing	Improvement %
Filter	**None**	BLDA	88.27%	0.0%	0.00	7.64E-02	0.00%
	None	RFLD	88.31%	0.0%	0.00	1.02E+00	0.00%
	Fisher score	BLDA	88.47%	64.8%	2.89E-02	1.63E-02	81.69%
	Fisher score	RFLD	88.23%	64.8%	2.84E-02	1.52E-01	84.93%
	BLDA	BLDA	88.25%	60.9%	1.32E+00	1.76E-02	80.05%
	RFLD	RFLD	88.22%	60.9%	7.18E+00	1.34E-01	86.75%
	Determination Coefficient(r^2)	BLDA	88.47%	64.8%	6.80E-03	1.59E-02	82.00%
	Determination Coefficient(r^2)	RFLD	88.23%	64.8%	6.70E-03	1.49E-01	85.24%
	Average		**88.3%**	**63.5%**	**1.43E+00**	**8.10E-02**	**83.44%**
Wrapper	**BLDA**	BLDA	88.05%	65.1%	1.34E+03	2.60E-02	71.97%
	RFLD	RFLD	88.05%	63.2%	9.51E+03	1.80E-01	82.68%
	Average		**88.05%**	**64.18%**	**5.42E+03**	**1.03E-01**	**77.33%**
Hybrid	**Fisher score + BLDA**	BLDA	70.69%	88.2%	2.22E+02	8.55E-03	90.92%
	Fisher score + RFLD	RFLD	70.82%	88.0%	1.53E+03	2.97E-02	97.12%
	BLDA + BLDA	BLDA	75.07%	86.4%	2.50E+02	8.33E-03	91.13%
	RFLD + RFLD	RFLD	72.85%	85.6%	1.87E+03	3.50E-02	96.56%
	r^2 + BLDA	BLDA	72.55%	88.5%	2.15E+02	8.29E-03	91.24%
	r^2 + RFLD	RFLD	72.60%	87.7%	1.73E+03	3.60E-02	96.54%
	Average		**72.43%**	**87.39%**	**9.69E+02**	**2.10E-02**	**93.92%**

Correct rate: According to the result we can see that there was no significant difference in accuracy for both Filter and Wrapper, on the other hand Hybrid accuracy was decreased which is going to be illustrated in Section 5.4.3.

Feature reduction: Hybrid was the best in feature reduction percentage with average 87.39% and the highest feature reduction percentage was 88.5% for r^2 with BLDA as hybrid feature selection. The average of feature reduction for filter and wrapper were 63.5% and 64.18 respectively.

Feature selection running time: In general filter running time was very fast during feature selection. While hybrid was slow and wrapper was the very slow. According to this filter types are preferred if the training time is critical.

Classifier training and testing running time: The running time during training and testing the classifier was improved by average 83.44% for filter types. For hybrid the improvement was 93.92%. While in wrapper the improvement was 77.33%. It's clear that RFLD associated algorithms tend to take much time than BLDA associated algorithms, and this is due the extra computation during the regularization. FLD without regularization is much faster. Although hybrid improvement in time was the highest, the correct rate was decreased.

As a conclusion we can see that filter methods reduce the features by average 63.5%. The reduction for both fisher score and r^2 was 64.8%. While the reduction for both BLDA and RFLD was 60.9%. Figure 5-15 shows a comparison between filter methods. In this figure we can see that all of them tend to be the same after 100 features put BLDA filter feature selection was not good as the other methods in less number of features.

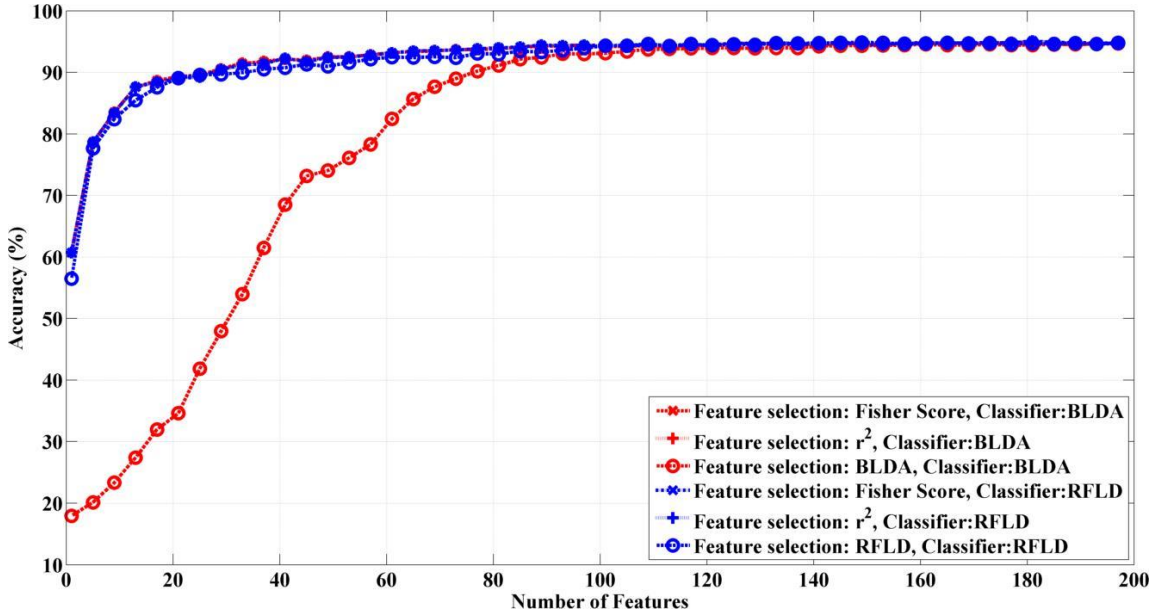

Figure 5-15 Comparing filter methods

The reduction improves the training and testing running time by average 83.44%. The highest improvement among filter methods was when using *RFLD* with 86.75%.The accuracy was not significantly affected for all of them. Feature selection running time for filter methods was fast. We can see that the fastest one were r^2 with 6.75ms. Fisher score running time was 28.6ms. BLDA and RFLDA were slower. BLDA running time was 1,320ms, while RFLD was 7,180ms. According to the previous comparison we can conclude that r^2 is the best one to be selected as filter feature selection method.

Wrapper methods save the accuracy. The reduction in features was as good as filter methods. On the other hand the computation time was the highest. For BLDA the needed time was 1,340s which is about 22.33minutes. For RFLD the needed time was 9,510s which is about 2.64hours.

Hybrid methods were the best in feature reduction 87.39% and so it was the best in the improvement of training and testing running time 93.92%. But this was at the expense of the correct rate. On the other hand, feature selection running time was high.

5.4.2 Why U. Hoffmann et al dataset accuracy was overcome KAUH dataset?

All through the offline analysis U. Hoffmann et al dataset accuracy was overcome KAUH dataset. This could be explained by the difference in the paradigm and in particular the stimulation method. In U. Hoffmann et al. [1] dataset the stimulation was done for each image, one stimulus per image. Thus the images are recognized directly. On the other hand, the stimulation in KAUH dataset were done for each raw\column, one stimuli per raw\column at a time. This means that the character is identified by two stimuli's, one for the raw and one for the column. The desired character is the intersection of them. In other words, the error in KAUH dataset is doubled. Once the raw or the column is wrong, the character is wrong. Figure 5-11 illustrates the curves obtained considering raw\column error. Table 5-14 illustrates the comparison using the other criteria.

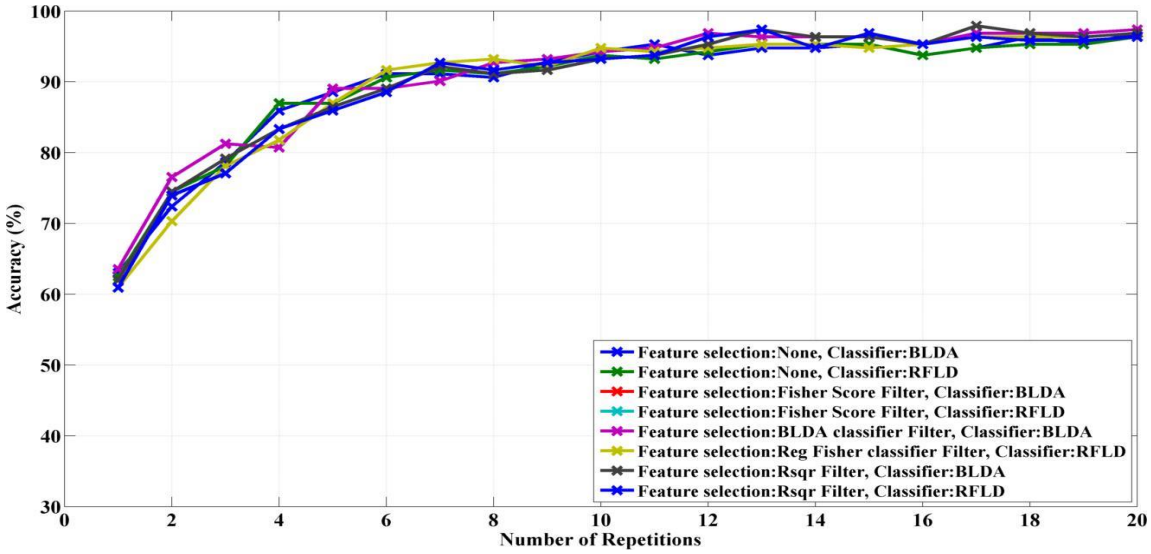

Figure 5-16 Filter results for KAUH dataset considering raw\column error

Table 5-14 Filter results for KAUH dataset considering raw\column error

Feature Selection Method	Classifier	Correct rate% (mean ± S.D)	Feature Reduction percentage %	Running Time (s)	
				Feature selection	Training + Testing
None	BLDA	89.97 ± 8.84	0.0%	5.84E-05	1.09E-01
None	RFLD	89.87 ± 8.67	0.0%	4.49E-05	1.32E+00
Fisher score	BLDA	90.26 ± 9.24	64.8%	3.56E-02	1.78E-02
Fisher score	RFLD	89.95 ± 9.5	64.8%	3.36E-02	1.96E-01
BLDA	BLDA	90.73 ± 8.84	60.9%	1.88E+00	1.88E-02
RFLD	RFLD	89.92 ± 9.78	60.9%	7.96E+00	1.85E-01
Determination Coefficient(r^2)	BLDA	90.26 ± 9.24	64.8%	7.64E-03	1.47E-02
Determination Coefficient(r^2)	RFLD	89.95 ± 9.5	64.8%	7.32E-03	1.93E-01

5.4.3 Hybrid and Over-fitting

Although, Hybrid methods reduce the feature size and therefore increase the experiment speed, the accuracy was decreased. For U. Hoffmann et al dataset the reduction in accuracy was 15% while in KAUH data set the reduction was 28%. This reduction in accuracy can be explained with over-fitting.

A classifier *over-fits* the dataset if it models the training data too well and its predictions are poor [56]. Jensen and Cohen [57] argue that over-searching the training data for optimal feature subsets is a problem. In this case the feature selection process can come up with a subset of noisy features that is not truly relevant for the classification of mental tasks but is only correlated to the labels by coincidence [29]. Table 5-15 illustrates this problem using BLDA classifier, and r^2 with BLDA as feature selection. We can see that without involving the optimization average feature size was 90 features, average training accuracy was 64.81%, and average testing accuracy was 62.86%. After optimization feature size was improved with average 35.44 features, accuracies during training were optimized with average 66.03%; put accuracies during testing were decreased. The average was 56.93%. We should recall, as mentioned in section 4.4.2.2, that the optimization were done through cross validation using k-fold=5. The performance was tested again using k-fold=10 put no improvement was acquired.

Table 5-15 Hybrid and over-fitting illustration

Subjects	Sessions	Without Optimization			With Optimization		
		Feature size	Accuracy		Feature size	Accuracy	
			Training	Testing		Training	Testing
Subject # 1	Session#1	90	67.11%	65.83%	35	68.06%	56.83%
	Session#2	90	67.44%	66.00%	36	67.67%	64.67%
	Session#3	90	69.39%	59.33%	38	72.22%	51.33%
	Session#4	90	67.50%	67.33%	32	67.50%	59.83%
Subject # 2	Session#1	90	60.06%	58.67%	36	62.17%	57.83%
	Session#2	90	61.44%	59.50%	32	61.72%	54.33%
	Session#3	90	62.89%	56.00%	37	63.44%	55.00%
	Session#4	90	59.28%	60.17%	34	61.78%	57.83%
Subject # 3	Session#1	90	65.94%	66.33%	35	66.83%	58.67%
	Session#2	90	65.83%	66.17%	32	66.56%	59.83%
	Session#3	90	66.11%	59.83%	31	66.72%	52.67%
	Session#4	90	65.83%	67.83%	37	67.11%	60.83%
Subject # 4	Session#1	90	65.11%	63.33%	41	66.89%	54.67%
	Session#2	90	63.50%	65.33%	39	66.22%	59.00%
	Session#3	90	65.06%	65.17%	38	66.28%	55.00%
	Session#4	90	64.50%	59.00%	34	65.39%	52.50%
Average		**90**	**64.81%**	**62.86%**	**35.44**	**66.03%**	**56.93%**

5.4.4 Comparing our results with previous studies

As previously mentioned, most of feature selection methods were applied in BCI in motor imagery paradigm. Less intention was devoted for P300. In 2008 Hoffmann et al. [52] apply SBDA, which is an algorithm that uses ARD for feature selection. SBLDA was tested for subjects 6, 7, 8, and 9 for 32 electrodes. To compare the results obtained using r^2 filter, the same data set with the same electrodes number was used. The comparison was based on PBA measure.

The first row of the table is the PBA as appeared in [1]. The second row is the PBA as obtained using PBA code. These values – if rounded - are the same of in row first row except for S7 the difference was 1%. The third row is the PBA as obtained using SBLDA as appeared in [52]. The fourth row is the PBA as obtained using r^2 filter. Although r^2 filter did not outperformed SBLDA the results are comparable. The deference in PBA was less than 1.69% while the deference in reduction was 2.6%.

Table 5-16 Comparing r^2 results with SBLDA

n	Feature selection	classifier	S6	S7	S8	S9	(Mean ± STD)		Features reduction %
							PBA	Features	
1	None	BLDA	70	72	87	58	72±12	1024±0	0
2	None	BLDA	70.21	70.63	86.88	58.13	71.46±10.22	1024±0	0
3	SBLDA	SBLDA	68	72	87	59	72±12	229±20	77.6
4	r^2 Filter	BLDA	70.42	71.67	86.04	53.13	70.31±11.67	256±0	75

Chapter 6 Conclusion and Future Work

6.1 Conclusion

This thesis discusses the impact of adding feature selection module to P300 based Brain Computer Interface BCI. Three types of feature selection algorithms were applied. These types are Filter, Wrapper, and Hybrid. Deferential evolution was used as searching algorithm. As a conclusion, Filter types were the preferred to be selected as feature selection method, in particular r^2. This is due to the good reduction in dimension and low computational cost. The time required for training and testing the classifier was improved by 83.62%. When comparing with filter, high computational cost was required for both wrapper and hybrid with no significant improvement in performance.

6.2 Contribution of the Thesis

The main contribution in this thesis was:

- The utilization of DE as feature selection algorithm for P300 based BCI.
- The given comparison between the three types of feature selection is another contribution.
- The improvement of U. Hoffmann et al work by CAR and by adding feature selection module.
- Transfer BCI technology, and improving the infrastructure at KAUH for further research.

This thesis work resulted in 1 conference paper, 2 journal papers and 1 other paper under publishing.

6.3 Future work

As a future work we can study the remedy of over-fitting problem with hybrid methods. Other possible extension is feature selection for unsupervised BCI. Alternatives of searching algorithms may be valuable for this context. Fixed number of blocks only was used in this thesis. One may improve the system by modifying it to the dynamic number of blocks. Another improvement may include the study of multi-objective optimization. Last but not least, the impact of dimensionality reduction using feature extraction rather than feature selection may be a good extension.

References

[1] Hoffmann, Ulrich , Vesin, Jean-Marc, Ebrahimi, Touradj, and Diserens, Karin (2008) An efficient P300-based brain–computer interface for disabled subjects, Journal of Neuroscience Methods 167, pp. 115–125

[2] Smith, Raymond C. (2004) Electroencephalograph based Brain Computer Interfaces, Master Thesis, University College Dublin, Dublin, Ireland.

[3] Luck, Steven J. (2005) An Introduction to the Event Related Potential Technique Cognitive Neuroscience, Cambridge, Mass. MIT Press.

[4] Mason S. G., Bashashati A., Fatourechi M., Navarro K. F., And Birch G. E. (2007) A Comprehensive Survey of Brain Interface Technology Designs, Annals of Biomedical Engineering , Vol. 35, No. 2, ,pp. 137–169.

[5] Citi, Luca, Poli, Riccardo and Cinel, Caterina. Antalya (2009) Exploiting P300 Amplitude Variations Can Improve Classification Accuracy in Donchin's BCI Speller, IEEE. Proceedings of the 4th International IEEE EMBS Conference on Neural Engineering. pp. 478-481.

[6] Arbabi E. and Shamsollahi M.B. (2005) Comparison Between Different Types Of Features Used For Classification In BCI, 12th International Conference on Biomedical Engineering (ICBME2005), Singapore.

[7] Lotte F., Congedo M., Lécuyer A., Lamarche, F., and Arnaldi B. (2007) A Review of Classification Algorithms for EEG-based Brain-Computer Interfaces, Journal of Neural Engineering.

[8] Müller Klaus-Robert, Anderson Charles W., and Birch Gary E. (2003) Linear and Nonlinear Methods for Brain–Computer Interfaces, IEEE Transactions on Neural Systems And Rehabilitation Engineering, VOL. 11, NO. 2, pp 165 – 169

[9] Croux, C., Filzmoser P., and Joossens K., (2008) Classification efficiencies for robust linear discriminant analysis, Statistica Sinica, 18(2): p. 581-599.

[10] Blankertz, B., Tangermann, M. , Vidaurre, C. , Dickhaus, T. , Sannelli, C., Popescu, F., Fazli, S., Danóczy,M., Curio, G., and Müller, Klaus-Robert (2010) Detecting Mental States by Machine Learning Techniques: The Berlin Brain–Computer Interface, B. Graimann et al. (eds.), Brain–Computer Interfaces, The Frontiers Collection,Springer-Verlag Berlin Heidelberg

[11] Martin Sewell (2007) Feature Selection, Access date, June 5, 2012, from: http://www.cs.ucl.ac.uk/staff/M.Sewell/publications.html

[12] Jain, Anil K., Duin, Robert P.W., and Mao Jianchang (2000) Statistical Pattern Recognition: A Review, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 1 ,pp.4-37.

[13] Dash, M., Liu, H. (1997) Feature Selection for Classification, Intelligent Data Analysis 1 131–156.

[14] Sellers, E., Donchin, E. (2006) A P300-based brain-computer interface: Initial tests by ALS patients, Clinical Neurophysiology 117, pp538–548.

[15] Donchin, E., Spencer, K., Wijesinghe, R. (2000) The mental prosthesis: Assessing the speed of a P300-based brain-computer interface, IEEE Transactions on Rehabilitation Engineering, pp174–179.

[16] Bostanov, V. (2004) Feature extraction from event-related brain potentials with the continuous wavelet transform and the t-value scalogram, IEEE Transactions on Biomedical Engineering, pp1057–1061.

[17] Lalor, E. C., Kelly, S. P., Finucane, C., Burke, R., Smith, R., Reilly,R. B., and McDarby, G. (2005) Steady-state VEP-based brain-computer interface control in an immersive 3D gaming environment, EURASIP Journal on Applied Signal Processing, pp 3156–3164.

[18] Schlögl, A., Lee, F., Bischof, H., Pfurtscheller, G. (2005) Characterization of four-class motor imagery EEG data for the BCI-competition 2005, Journal of Neural Engineering , pp L14–L22.

[19] Lemm, S., Schäfer, C., Curio, G. (2004) Probabilistic modeling of sensorimotor mu-rhythms for classification of imaginary hand movements, IEEE Transactions on Biomedical Engineering, pp.1077–1080.

[20] McFarland, Dennis J., McCane, Lynn M., David, Stephen V., and Wolpaw, Jonathan R. (1997) Spatial filter selection for EEG-based communication, EEG Clin. Neurophysiol., Volume 103, Issue 3, pp 386-394

[21] Ng, S.C., Raveendran, P. (2007) "Comparison of different Montages on to EEG classification", Biomed 06, IFMBE Proceedings 15, pp. 365-368

[22] Lal, Thomas N., Schroder, Michael, Hinterberger, Thilo, Weston, Jason, Bogdan, Martin, Birbaumer, Niels, and Scholkopf, Bernhard (2004) Support vector channel selection in BCI. IEEE Transactions on Biomedical Engineering, pp1003–1010.

[23] Rakotomamonjy, Alain, and Guigue, Vincent (2008) BCI Competition III: Dataset II-Ensemble of SVMs for BCI P300 Speller, IEEE Transactions on Biomedical Engineering, vol. 55, no. 3

[24] Salimi-Khorshidi, Gholamreza, Nasrabadi, Ali Motie, and Golpayegani, Mohammadreza Hashemi (2008) Fusion of classic P300 detection methods'inferences in a framework of fuzzy labels, Artificial Intelligence in Medicine , 44, pp 247—259

[25] Liu, H. and Yu, L. (2005) Toward integrating feature selection algorithms for classification and clustering, IEEE Trans. on Knowledge and Data Engineering, 17(3):1–12.

[26] Cabrera, Alvaro R. (2009) Feature Extraction and Classification for Brain-Computer Interfaces Ph.D. Thesis, Brain-Computer Interface Laboratory Center for Sensory-Motor Interaction (SMI) ,Department of Health Science and Technology Aalborg University, Denmark.

[27] Guyon ,I. and Elisseeff, A. (2003) An introduction to variable and feature selection, J. Mach. Learn. Res., 3:1157–1182.

[28] Wonnacott, T. H. and Wonnacott, R. (1977) Introductory Statistics, John Wiley and Sons, New York, 3ed edition.

[29] Tangermann, MichaelW. (2007) Feature Selection for Brain-Computer Interfaces, Ph.D. Thesis, Eberhard Karls University , Tuebingen

[30] Kennedy, J. and Eberhart, R. Perth (1995) Particle Swarm Optimization, IEEE International Conference on Neural Networks

[31] Shi, Y. and Eberhart, R. Anchorage (1998) A Modified Particle Swarm Optimizer, IEEE International Conference on Evolutionary Computation

[32] Storn, R. and Price, K. (1997) Differential evolution - a simple and efficient heuristic for global optimization over continuous space, Journal of Global Optimization, Vol. 11, pp. 341-359

[33] Müller, K.-R., Krauledar, M., Dornhege, G., Curio, G., Blankertz, B. (2004) Machine Learning Techniques for Brain-Computer Interfaces, Biomedical Technology, 11–22

[34] AlZoubi, Omar, Koprinska, Irena and Calvo, Rafael A. (2008) Classification of Brain-Computer Interface Data, the Seventh Australasian Data Mining Conference.

[35] Koprinska, Irena (2010) Feature Selection for Brain-Computer Interfaces, T. Theeramunkong et al. (Eds.): PAKDD Workshops 2009, LNAI 5669, pp. 100–111, Springer-Verlag Berlin Heidelberg

[36] Spüler, M., Rosenstiel, W., Bogdan, M. (2011) A fast feature selection method for high-dimensional MEG BCI data, Proceedings of the 5th International Brain-Computer Interface Conference, pp. 24-27

[37] Dias, N.S., Jacinto, L.R., Mendes, P.M., and Correia, J.H. (2009) Feature Down-Selection in Brain-Computer Interfaces Dimensionality Reduction and Discrimination Power, Proceedings of the 4th International IEEE EMBS Conference on Neural Engineering, Antalya, Turkey, April 29 - May 2, 2009

[38] Dat, Tran Huy, and Guan, Cuntai (2007) Feature Selection Based on Fisher Ratio and Mutual Information Analyses for Robust Brain Computer Interface, Acoustics, Speech and Signal Processing, ICASSP 2007.

[39] Cabrera, Alvaro Fuentes, Farina, Dario, and Dremstrup, Kim (2010) Comparison of Feature Selection and Classification Methods for a Brain-Computer Interface Driven by Non-Motor Imagery, Medical and Biological Engineering and Computing, Volume 48, Number 2 (2010), 123-132

[40] Zhang, Haihong, Ang, Kai Keng, Guan, Cuntai, and Wang, Chuanchu (2009) Spatio-spectral feature selection based on robust mutual information estimate for brain computer interfaces, 31st Annual International Conference of the IEEE EMBS Minneapolis, Minnesota, USA, September 2-6, 2009

[41] Atum, Yanina, Gareis, Iván, Gentiletti, Gerardo, Acevedo, Rubén, and Rufiner, Leonardo (2010) Genetic Feature Selection to Optimally Detect P300 in Brain Computer Interfaces, 32nd Annual International Conference of the IEEE EMBS, Buenos Aires, Argentina, August 31 - September 4, 2010

[42] Citi, Luca, Poli, Riccardo, Cinel, Caterina, and Sepulveda, Francisco (2004) Feature Selection and Classification in Brain Computer Interfaces by a Genetic Algorithm, Genetic and Evolutionary Computation Conference, (GECCO) 2004, June 26 - 30, 2004, Seattle, Washington, USA

[43] Schroder, M., Bogdan, M., Hinterberger, T., and Birbaumer, N. (2003) Automated EEG Feature selection for Brain Computer Interfaces, First International IEEE EMBS Conference on Neural Engineering, Conference Proceedings.

[44] Hasan, Bashar Awwad Shiekh (2010) Adaptive Methods Exploiting the Time Structure in EEG for Self-paced Brain-Computer Interfaces", Phd thesis, University of Essex, December, 2010

[45] Tanaka, Kenji, Kurita, Takio, Meyer, Friedrich, Berthouze, Luc, And Kawabe, Tohru (2006) Stepwise Feature Selection by Cross Validation for EEG-based Brain Computer Interface 2006 International Joint Conference on Neural Networks Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada

[46] Anderson , Charles W., and Kirby, Michael J. (2003) EEG Subspace Representations and Feature Selection for Brain-Computer Interfaces, 2003 conference on computer vision and pattern recognition workshop- volume 5 - June 12-22, 2003, Madison, Wisconsin

[47] Zhiping, Hu, Guangming, Chen, Cheng, Chen, He, Xu, and Jiacai, Zhang (2010) A new EEG feature selection method for self-paced brain-computer interface, 2010 10th International Conference on Intelligent Systems Design and Applications

[48] Khushaba, Rami N., Al-Ani, Ahmed, and Al-Jumaily, Adel (2008) Differential Evolution based Feature Subset Selection, 19th International Conference on Pattern Recognition. ICPR 2008.

[49] Haibin, Zhao, Xu, Wang, and Hong, Wang (2008) Feature Selection using Relative Wavelet Energy for Brain-Computer Interface Design, Control and Decision Conference. CCDC 2008. Chinese

[50] Qi, Hongzhi, Xu, Minpeng, Li, Wen, Yuan, Ding, Zhu, Weixi, An, Xingwei, Ming, Dong, Wan, Baikun, and Wang, Weijie (2010) Feature Selection Study of P300 Speller Using Support Vector Machine, Proceedings of the 2010 IEEE ,International Conference on Robotics and Biomimetics, December 14-18, 2010, Tianjin, China

[51] Nikolay Chumerin, Nikolay V. Manyakov, Adrien Combaz, Johan A. K. Suykens, Marc M. Van Hulle, (2009) An application of feature selection to on-line P300 detection in brain-computer interface Machine Learning for Signal Processing, 2009. MLSP 2009. IEEE International Workshop on 1-4 Sept.

[52] Hoffmann, Ulrich, Yazdani, Ashkan, Vesin, Jean-Marc, Ebrahimi,Touradj (2008) Bayesian Feature Selection Applied in a P300 Brain-Computer Interface, 16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29

[53] Schalk, G., and Mellinger, J. (2010) A Practical Guide to Brain-Computer Interfacing with BCI2000. Springer

[54] g.tec. Medical Engineering. (2012) Access date, June 3, 2012, from: http://www.gtec.at/.

[55] Ranta, R., Salido-Ruiz, R., and Louis-Dorr, V. (2010) Reference estimation in EEG recordings, Conf Proc IEEE Eng Med Biol Soc.;2010:5371-4

[56] Kohavi, R., and John, G. (1997) Wrappers for feature selection. Artificial Intelligence, 97(1-2):273–324.

[57] Jensen, D. D., and Cohen, P. R. (2000) Multiple comparisons in induction algorithms, Machine Learning 38, no. 3, 309–338

[58] Reunanen, J. (2003) Overfitting in making comparisons between variable selection methods, JMLR, 3: 1371–1382.

[59] Schäfer, J. And Strimmer, K. (2005) A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. Stat Appl Genet Mol Biol, 4, Article32.

[60] Duda, R.O., Hart, P.E., and Stork,D.G. (2001) Pattern classification. Vol. 2. wiley New York.

[61] Goshvarpour, Ateke, and Goshvarpour, Atefeh (2012) Classification of Heart Rate Signals during Meditation using Lyapunov Exponents and Entropy", I.J. Intelligent Systems and Applications, 2, 35-41

[62] Serby, H., Yom-Tov, E., and Inbar, G. (2005) An improved P300-based brain-computer interface, IEEE Transactions on Neural Systems and Rehabilitation Engineering 13(1):89–98.

[63] Bishop, C. M., (2006) Pattern recognition and machine learning, Springer.

[64] Flotzinger, D., Pregenzer, M. , Pfurtscheller, G (1994) Feature selection with distinction sensitive learning vector quantisation and genetic algorithms , Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on 27 Jun- 2 Jul

[65] Mishra, S. K., (2006) Global Optimization by Differential Evolution and Particle Swarm Methods: Evaluation on Some Benchmark Functions Available at SSRN: http://dx.doi.org/10.2139/ssrn.933827

[66] H. Liu and H. Motoda, (1998) Less Is More, In Feature Extraction, Construction and Selection: A Data Mining Perspective, Kluwer Academic Publishers. Pages: 3 - 12. Editors: H. Liu and H. Motoda. July.

[67] Anahita Zarei, El-Sharkawi, M.A. (2005) Pareto Multi Objective Optimization, Intelligent Systems Application to Power Systems, 2005. Proceedings of the 13th International Conference.

[68] Dragan Savic (2002), Single-objective vs. Multiobjective Optimisation for Integrated Decision Support, Proceedings of the First Biennial Meeting of the International Environmental Modelling and Software Society.

[69] Berkman Sahinery, Heang-Ping Chan, Nicholas Petrick, Mark A Helvie and Mitchell M Goodsitt, Design of a high-sensitivity classifier based on a genetic algorithm: application to computer-aided diagnosis,Phys. Med. Biol. 43 (1998) 2853–2871

[70] George H. John, Ron Kohavi, Karl Pfleger (1994), Irrelevant Features and the Subset Selection Problem, In International Conference on Machine Learning (1994), pp. 121-129

[71] Auffarth, B., Lopez, M., Cerquides, J. (2010) Comparison of redundancy and relevance measures for feature selection in tissue classification of CT images Advances in Data Mining. Applications and Theoretical Aspects. p. 248-262. Springer