EEG Signal Processing in Brain-Computer interface

By

Mohammad-Mahdi Moazzami

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Computer Science

2012

Abstract

EEG Signal Processing in Brain-Computer interface

By

Mohammad-Mahdi Moazzami

It has been known for a long time that as neurons within the brain fire, some measurable electrical activities is produced. Electroencephalography (EEG) is one way of measuring and recording of the electrical signals produced by these electrical activities via some sensor arrays across the scalp. Although there is plentiful research in using EEG technology in the fields of neuroscience and cognitive science and the possibility of controlling computer-based devices by thought signals, utilizing of EEG measurements as inputs to the control devices has still been an emerging technology over the past 20 years, current BCIs suffer from many problems including inaccuracies, delays between thought, false positive detections, inter people variances, high costs, and constraints on invasive technologies, that needs further research in this area.

The purpose of this research is to examine this area from Machine Learning and Signal Processing perspective and exploit the Emotiv System as a cost-effective, noninvasive and also a portable EEG measurement device, and utilize it to build a thought-based BCI to control the cursor movement. This research also assists the analysis of EEG data by introducing the algorithms and techniques useful in processing of EEG signals and inferring the desired actions from the thoughts. It also offers a brief future potential capabilities of research based on the platform provided.

.

To

who have offered me unconditional love and care throughout the course of my life,

My lovely parents

Acknowledgements

I would like to thank my advisor, Prof. Matt Mutka, for his constant support and guidance rendered during my studies and this thesis work. I am grateful to Dr. Esfahanian and Dr. Xiao for serving on my committee. I would also like to thank the Computer Science Department at Michigan State University for providing the facilities for my studies, without which my research would have been impossible. Furthermore, I extend my gratitude towards the Emotiv Team who have developed a great cost effective product for anyone looking to get into EEG research and who have answered many of my questions on their closely monitored forums. Lastly I would like to thank my friends and family for their support.

.

If you can't explain it simply, you don't understand it well enough.

Albert Einstein

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Who don't love to control a device, a computer or a system with their mind? Interfaces between the brain and computers have been an essential element of science fiction where they are used in a variety of applications from controlling powered exoskeletons, robots, and artificial limbs to creating art envisioned by the user to allowing for machine-assisted telepathy.

This fantasy is not truly real yet, however simple BCIs do currently exist and research and public interest in them continues to grow.

This research explores the process in creating a novel BCI that utilizes the Emotiv EPOC System to measure EEG waves and controls the mouse cursor to do an action like dialling a phone.

1.1 ElectroEncephaloGraphy(EEG)

EEG waves are created by the firing of neurons in the brain and were first measured by Vladimir Pravdich-Neminsky who measured the electrical activity in the brains of dogs in 1912, although the term he used was electrocerebrogram [35].

Ten years later Hans Berger became the first to measure EEG waves in humans and, in addition to giving them their modern name, began what would become

intense research in utilizing these electrical measurements in the fields of neuroscience and psychology [23].

EEG waves are measured using electrodes attached to the scalp which are sensitive to changes in postsynaptic potentials of neurons in the cerebral cortex. Postsynaptic potentials are created by the combination of inhibitory and excitatory potentials located in the dendrites. These potentials are created in areas of local depolarization or polarization following the change in membrane conductance as neurotransmitters are released. Each electrode has a standard sensitivity of 7 μV/mm and averages the potentials measured in the area near the sensor. These averages are amplified and combined to show rhythmic activity that is classified by frequency (Table 1.1) [28].

Electrodes are usually placed along the scalp following the "10-20 International System of Electrode Placement" developed by Dr. Herbert Jasper in the 1950's that allows for standard measurements of various parts of the brain figure 1.2 see [26].

The primary research that utilizes EEG technology is based on the fact that this rhythmic activity is dependent upon mental state and can be influenced by level of alertness or various mental diseases. One of the historical downsides of EEG measurement has been the corruption of EEG data by artifacts, which are electrical signals that are picked up by the sensors that do not originate from cortical neurons. One of the most common cause of artifacts is eye movement and blinking, however other causes can include the use of scalp, neck, or other muscles or even poor contact between the scalp and the electrodes [30]. Many EEG systems attempt to reduce artifacts and general noise by utilizing reference electrodes placed in locations where there is little cortical activity and attempting to filter out correlated patterns [21].

The human brain is the site of consciousness, allowing humans to think, learn and create. The brain is broadly divided into the cerebrum, cerebellum, limbic system and the brain stem. The cerebrum is covered with a symmetric convoluted cortex

Band	Frequency(Hz)
Delta	1 to 4
Theta	4 to 7
Alpha	7 to 13
Beta	13 to 30
Gamma	30+

Table 1.1: EEG Bands and Frequencies - EEG Bands and Frequencies.

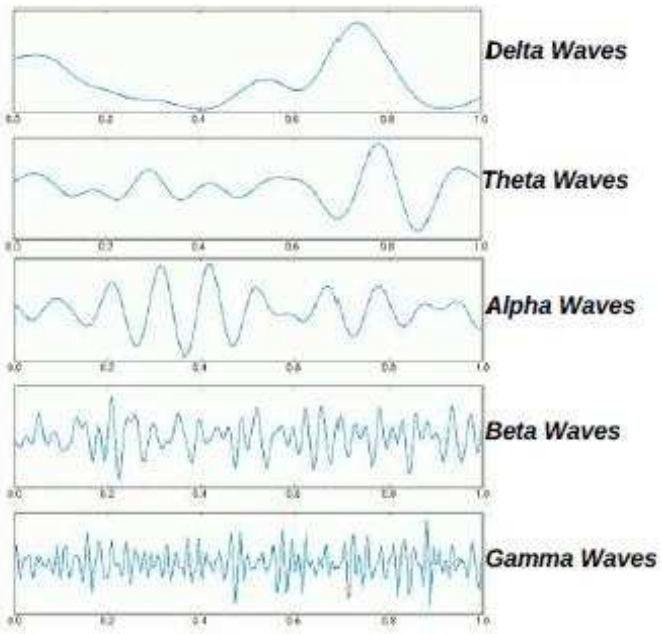

Figure 1.1: EEG Frequencies - The EEG Frequencies and different wave forms - Image reprinted from [42]. *"For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this thesis."*

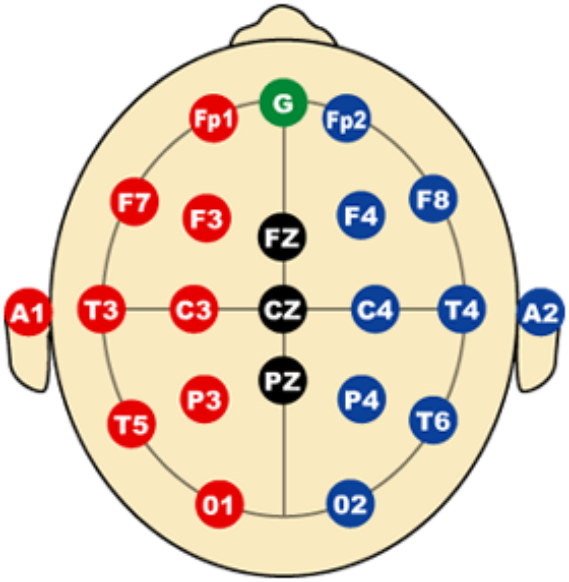

Figure 1.2: International 10-20 Electrode Placement - Odd numbers on the left, even on the right. Letters correspond to lobes F(rontal), T(emporal), P(arietal), and O(ccipital). C stands for Central (there is no central lobe)- [26].

with a left and right hemispheres. The brain stem is located below the cerebrum and the cerebellum is located beneath the cerebrum and behind the brain stem.

The limbic system, which lies at the core of the brain, contains the thalamus and hypothalamus among other parts. Anatomists divide the cerebrum into Frontal, Parietal, Temporal and Occipital lobes. These lobes inherit their names from the bones of the skull that overlie them.

It is generally agreed that the Frontal lobe is associated with planning, problem solving, reasoning, parts of speech, bodily movement and coordination. The Parietal lobe is associated with bodily movement, orientation and recognition (such as touch, taste, pressure, pain, heat, cold, etc). The Temporal lobe is associated with perception, auditory stimuli, memory and speech. The Occipital lobe is associated with visual stimuli.

Figure 1.3 is an image of human brain with the lobes and their associated functions. The brain is made up of approximately 100 billion neurons. Neurons are nerve cells with dendrite and axon projections that take information to and from

the nerve cells, respectively.

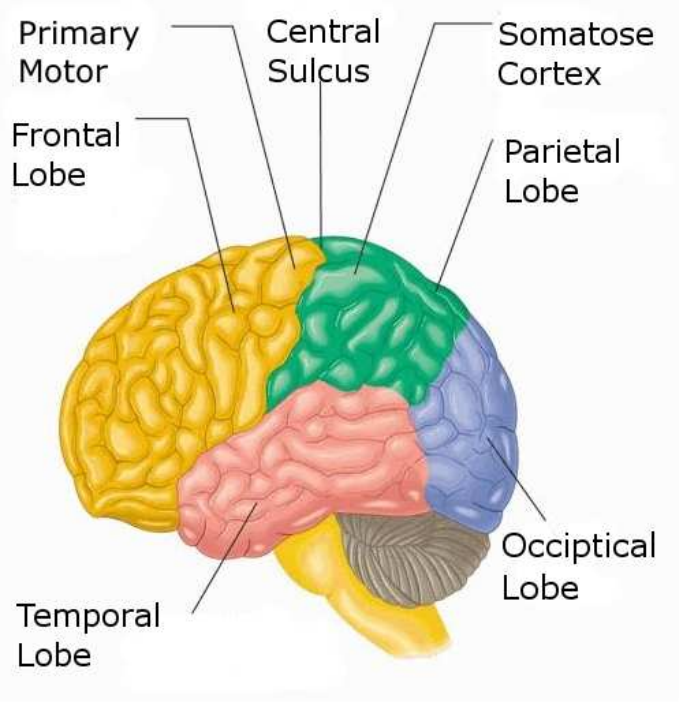

Figure 1.3: The human brain - Image reprinted from [41]

These neurons have a resting potential, typically between -70 to - 65 microvolts, which is the difference between the interior potential of the cell and the extra cellular space. A stimulus triggers an action potential in the neuron which sends out information (also know as firing property, impulse, spike) along the axons. Neurons transmit messages electrochemically. A neuron fires only when its resting potential drops below a threshold of -55 microvolts.

A spiking neuron triggers another neuron to spike and in turn another. This causes the resting potential to fluctuate as a result of impulses arriving from other neurons at contact points (synapses). These impulses result in post synaptic potentials which cause electric current to flow along the membrane of the cell body and dendrites.

This is the source of brain electric current. Brain electrical current consists mostly of Na+, K+, Ca++, and Cl- ions. Only large populations of active neurons

can generate electrical activity strong enough to be detected at the scalp as neurons tend to line up to fire.

The Electroencephalogram is defined as electrical activity of an alternating type recorded from the scalp surface after being picked up by metal electrodes and conductive media [38]. Electroencephalography (EEG) is the process of picking up the electrical activity from the cortex. Hans Berger suggested that periodic fluctuations of the EEG might be related in humans to cognitive processes [11]. Electrical activity recorded of the scalp with surface electrodes constitute a non-invasive approach to gathering EEG data, while semi-invasive or invasive approaches implant electrodes under the skull or on the brain, respectively [17]. The trade off in these approaches lies in the EEG source localization, quality of EEG data, surgical process involved and/or the effect of the electrodes interacting with the tissues.

EEG recorded over a continuous period of time are characterized as spontaneous EEG. These signals are not time locked and are usually triggered by an auditory or visual cue. Another closely related application of EEG is the event related potential. Electrical activity triggered by presenting a stimulus is time locked and is an evoked response.

These evoked responses are called the event related potentials (ERP). The latency and peaks in the evoked responses are in predictable ranges given the stimulus. Typically ERPs are recorded from a single electrode over the region of activation along with a ground electrode. EEG has applications among clinical diagnosis, neuroscience and the entertainment (gaming) industry.

EEG activity is broadly divided into five frequency bands. The boundaries are flexible but do not vary much from 0.5-4 Hz (delta), 5-8 Hz (theta), 9-12 Hz (alpha), 13-30 Hz (beta) and above 30 Hz (gamma) frequencies. Refer to Figure 2.2 for EEG frequency bands. The EEG frequencies and their associated activities are the delta activity is associated with deep sleep. Theta activity is associated with

hypnagogic imagery, rapid eye movement (REM), sleep, problem solving, attention and hypnosis [11]. Alpha activity is associated with relaxation and non-cognitive processes [3]. Beta activity with active thinking or active attention [3]. Gamma frequencies are associated with attention and recognition [15].

Over the years various BCI models have been developed that categorically fall into the BCI model spectrum. The primary difference lies in the interaction between the user and the BCI. At one end of the BCI model spectrum is an architecture where all emphasis is placed on the subject to generate quality EEG data with little effort on the BCI to recognize the task. In a way it is training the subject to control their EEG activity.

On the other end of the spectrum the burden lies on the BCI to recognize the task with the user putting in little effort to generate quality data. Somewhere in between this spectrum is an architecture where both the subject and the BCI mutually learn and evolve together. This is achieved when the BCI gives feedback to the user regarding the quality of EEG data generated [17].

Some BCI architectures also use the error related negativity (ERN) signals, a type of ERP, which are used along with the EEG to aid in identification of the subject's true intentions [10].

The purpose of this thesis is to understand the patterns of EEG signals and design methods and algorithms to take advantages of intentions associated with each type of signal in making a Brain-Computer Interface, and to examine the Emotiv EPOC System as a cost-effective gateway to non-invasive portable EEG measurements and utilize it to build a EEG-based BCI. This research assists the analysis of the current pros and cons of EEG technology as it relates to BCIs and offers a brief perspective of the future potential capabilities of BCI systems. The rest of this thesis is organized as following: In chapter 2, a survey of accomplished researches in the area of EEG-based BCI is presented. Different types of BCIs are discussed and some of the

pattern recognition and Machine Learning algorithms and techniques used in BCIs to mine the EEG signals are introduced. Chapter 3, discusses about Emotiv EPOC system as an EEG measurement tool, and gives a comprehensive explanation of the APIs associated with its SDK and describes how to exploit it to making a EEG-based BCI. And chapter 4 describes the experiment and quantitative measurement of the BCI system designed based on the learning algorithms and the capabilities provided by the Emotiv EPOC. Chapter 5 summarizes the thesis and mentions the potential future work.

Chapter 2

Background Survey

The term Brain-Computer Interface first appeared in scientific literature in the 1970's, though the idea of hooking up the mind to computers was nothing new [38]. The ultimate goal of BCI research is to create a system that not only an open-loop system that responds to users thoughts but a closed loop system that also gives feedback to the user.

Researchers initially focused on the motor-cortex of the brain, the area which controls muscle movements, and testing on animals quickly showed that the natural learning behaviors of the brain could easily adapt to new stimuli as well as control the firing of specific areas of the brain [11]. This research dealt primarily with invasive techniques but slowly algorithms emerged which were able to decode the motor neuron responses in monkeys in real-time and translate them into robotic activity [17], [40].

Recently, a system developed by researchers and Carnegie Mellon University and the University of Pittsburgh allowed a monkey to feed itself via a prosthetic arm using only its thoughts [3]. This research is extremely promising for the disabled, and indeed by 2006 a system was developed for a tetraplegiac that enabled him to use prosthetic devices, a mouse cursor, and a television via a 96-micro-electrode array implanted into his primary motor cortex [15].

Despite these achievements, research is beginning to veer away from invasive BCIs due to the costly and dangerous nature of the surgeries required for such systems. Non-invasive alternatives for BCIs include EEG technology, Magnetoencephalography (MEG), and Magnetic Resonance Imaging (MRI), as well as the partially invasive Electrocorticography where sensors are placed within the skull but outside the gray matter of the brain. Non-invasive methods are limited in that they are often susceptible to noise, have worse signal resolution due to distance from the brain, and have difficulty recording the inner workings of the brain. However more sophisticated systems are constantly emerging to combat these difficulties and non-invasive techniques have the advantage of lower cost, greater portability, and the fact that they do not require any special surgery [10].

2.1 Brain-Computer Interface

A brain-computer interface (BCI), sometimes called a direct neural interface or a brain-machine interface (BMI), is a direct communication pathway between the brain and an external device. BCIs are often aimed at assisting, augmenting or repairing human cognitive or sensorymotor functions.

BCI technology can be divided into two categories: invasive and non-invasive. The invasive BCI employs insertion of electrodes below the skull, often onto the surface of the brain [10], but occasionally deep within the cortex for purposes of monitoring one particular region [21].

Non-invasive methods generally involve measurements of electro-magnetic potentials from outside the head. This thesis is concerned with electroencephalography (EEG), which is a non-intrusive method of recording voltage measurements from the scalp surface.

Electroencephalography (EEG) is the recording of electrical activity along the scalp. EEG measures voltage fluctuations resulting from ionic current flows within

the neurons of the brain. The brain's electrical charge is maintained by billions of neurons. Neurons are electrically charged by membrane transport proteins that pump ions across their membranes.

EEG is the most studied potential non-invasive interface, mostly because of its fine temporal resolution, portability, ease of use and low set-up cost, but as well as the technology's susceptibility to noise. Another substantial barrier to use EEG as a brain-computer interface is the extensive training required before users can work the technology.

BCIs equipped with EEG for data collection are easy to setup, can be deployed in numerous environments, are preferred for their lack of risk and are inexpensive.

Other techniques used to map brain activation are functional magnetic resonance imaging (fMRI), positron emission tomography (PET) and magnetoencephalography (MEG). fMRI measures the changes in blood flow level (blood oxygen level) in the brain. In PET a tracer isotope is injected into the subject that emits gamma rays when it annihilates, these gamma rays are further traced by placing the subject under a scanner. These approaches have relatively poor time resolution but excellent spacial resolution when compared to EEG. MEG is an imaging technique that measures the magnetic field produced by electrical activity; EEG can be simultaneously recorded along with MEG.

Recognizing brainwave patterns for Brain-Computer interface presents an interesting pattern recognition problem. It can be determined that the BCI is a classification problem, and like any classification task it operates in two phases: training and testing.

In the training phase the subjects perform different tasks (imagine moving hands, feet, doing simple math, etc) and the BCI is trained to recognize one task from another. In the testing phase (classifying phase) the already trained BCI is applied on new data sample to determine the intended class label (intended task).

An accurate recognition system must be able to accommodate variations introduced by the different brain patterns of individuals as well as variations in the brain patterns caused by the use of different mind state like being tired or having less concentration. This coupled with numerous challenges like the quality of Electroencephalography (EEG) signals, its non-stationary nature, common EEG artifacts and the feature dimensionality presents a significant challenge to EEG-based BCI systems.

Though the idea of using EEG waves as input to BCIs has existed since the initial conception of BCIs, actual working BCIs based on EEG input have only recently appeared [38]. Most EEG-BCI systems follow a similar paradigm of reading in and analyzing EEG data, translating that data into device output, and giving some sort of feedback to the user. Figure 2.1 illustrates the high level block diagram a genral BCI system. However implementing this model can be extremely challenging. The primary difficulty in creating an EEG-based BCI is the feature extraction and classification of EEG data that must be done in real-time if it is to have any use.

Feature extraction deals with separating useful EEG data from noise and simplifying that data so that classification, the problem of trying to decide what the extracted data represents, can occur. There is no best way of extracting features from EEG data and modern BCIs often use several types of feature extraction including wavelet transforms, Fourier transforms, and various other types of filters.

The major features that EEG-BCI systems rely on are event-related potentials (ERPs) and event-related changes in specific frequency bands. The P300 wave is one of the most often used ERPs in BCIs and is utilized because it is easily detectable and is only created in response to specific stimuli [26], [32].

BCI systems are further complicated by the fact that there is no standard way of classifying the extracted data. Various means including neural networks, threshold parameters, and various other types of pattern recognizers are employed to try to

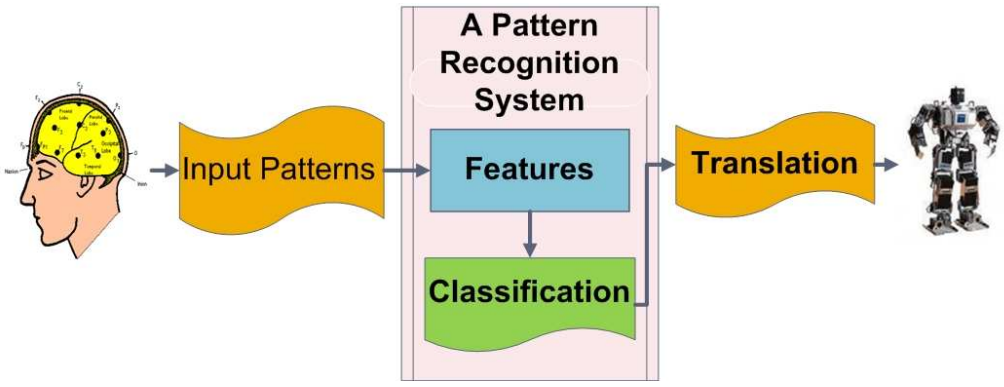

Figure 2.1: BCI-Components - A Brain-Computer Interface Basic Components

match the input data to known categories of EEG archetypes [18].

Furthermore, researchers have also relied on unsupervised learning algorithms to find natural clusters of EEG segments that are indicative of certain kinds of mental activities with varying degrees of success [20], [26].

Feedback is essential in BCI systems as it allows users to understand what brainwaves they just produced and to learn behavior that can be effectively classified and controlled. Feedback can be in the form of visual or auditory cues and even haptic sensations, and ongoing research is still attempting to figure out the optimal form feedback should take [16].

EEG-BCIs can also be classified as either synchronous or asynchronous. The computer drives synchronous systems by giving the user a cue to perform a certain mental action and then recording the user's EEG patterns in a fixed time-window. Asynchronous systems, on the other hand, are driven by the user and operate by passively and continuously monitoring the user's EEG data and attempting to classify it on the fly. Synchronous protocols are far easier to construct and have historically been the primary way of operating BCI systems [26].

EEG-BCI systems have made incredible progress in recent years. By 2000, researchers had created a thought-translation device for completely paralyzed patients that allowed patients to select characters based on their thoughts, although the char-

acter selection process was time consuming and not perfectly accurate [1].

By 2008, researchers collaborating from Switzerland, Belgium, and Spain created a feasible asynchronous BCI that controlled a motorized wheelchair with a high degree of accuracy though again the system was not perfect [12].

Today, the 2010 DARPA budget has allocated $4 million to develop an EEG-based program called Silent Talk which will allow user-to-user communication on the battlefield without the use of vocalized speech through analysis of neural signals [6]
.

State-of-the-art EEG-based BCIs are a cutting-edge emerging technology and researchers are constantly developing newer and more accurate algorithms to aid making BCIs that are simpler and more effective than ever before.

2.2 Pattern Recognition and Machine Learning in BCI systems

Recognition of intended commands from Brain signals presents all the classic challenges associated with any pattern recognition problem. These challenges include noise in the input data, variations and ambiguities in the input data, feature extraction, as well as overall recognizer performance. A Brain-Computer Interface (BCI) system designed to meet specific customer performance requirements must deal with all of these challenges.

Figure 2.1 illustrates the basic components found in a typical BCI system. Figure 2.2 shows these components in more details.

Two major problems opposing BCI developers are the non-stationarity and natural variability of EEG brain signals. Data from the same experimental pattern but recorded on different days (or even in different sessions on the same day) are likely to exhibit significant differences. Besides the variability in neuronal activity, the user's

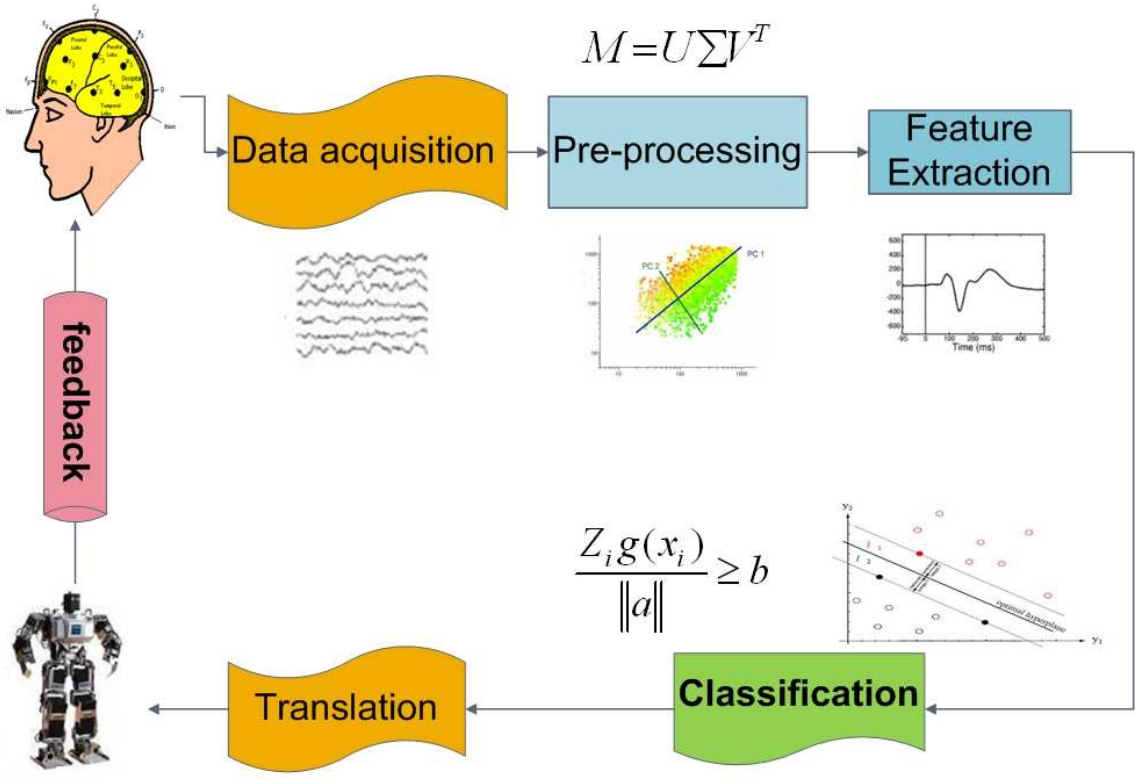

Figure 2.2: BCI Components Details - A Brain-Computer Interface Basic Components

current mental state may affect the measured signals as well: Stress, excessive workload, boredom, or frustration may cause temporary distortions of EEG activity. User adaptation to the BCI, as well as the changing impedance of EEG sensors during recording, contribute to making the recorded signal statistically non-stationary.

Pattern recognition and machine learning algorithms play a crucial role in recognizing patterns in noisy EEG signals and translating them into control signals. Pattern recognition methods are sometimes broadly categorized as performing classification or regression. Classification involves assigning one of N labels to new brain signals, given a labelled training set consisting of previously seen signals and their corresponding labels. Regression typically involves mapping brain signals directly to a continuous output signal, such as position or velocity of a prosthetic device.

The standard training of a classification or regression algorithm consists of collecting data from a user prior to BCI use and analyzing it offline to find user-specific parameters. A major drawback to this procedure is that the brain signal patterns of a user typically change over time as a result of feedback during BCI use, making parameters learned offline suboptimal. Therefore there are some research activities being done to develop online adaptation methods [34], [39] as well as research on extracting features that remain stable over users and time [33].

The translation of brain activities into control commands for external devices involves a series of signal processing and pattern recognition stages such as: signal acquisition, preprocessing, feature extraction and classification. This is for generating a control signal which is followed by a feedback to the user by the application being controlled. This section introduces each of these stages.

2.2.1 Signal acquisition and quality issues

The non-invasive electroencephalogram (EEG) have emerged as important source of brain signals for BCI in humans. EEG signals are potential differences recorded

from electrodes placed on the scalp. The measured signals are thought to reflect the temporal and spatial summation of post-synaptic potentials from large groups of cortical neurons located beneath the electrode.

EEG has time resolution in the range of millisecond and a spatial resolution in the range of cm^2. EEG signal's amplitude is in the range of a few microvolts ($\pm$ 100 μV). Some other methods types of signals like ECoG, which is sometimes called as the invasive electroencephalogram, has better spatial resolution (in the range of mm^2) with amplitude of much greater than what EEG has (up to 100 times greater) [27].

Therefore EEG has a poor spatial resolution and signal strength. It's poor spatial resolution is mainly caused by different layers of tissue placed between source of signal (neural activities in the cortex) and the place measuremnt happens (sensors placed on the scalp). These layers of tissues are meninges, cerebrospinal fluid, skull, scalp and act as a resistor and effectively a low pass filter and weaken the source signal.

Hence, because of small amplitude, EEG signals are very vulnerable to noise and artifacts. The most frequent artifacts are muscle activities (electromyogram (EMG)) and eye movements (electrooculogram (EOG)) generated by the user and signal contamination due to nearby electrical devices (e.g., 60-Hz power line interference). Additional noise sources include changing electrode impedance and the user's varying psychological states due to boredom, distraction, stress, or frustration (e.g., caused by BCI mistranslation). Some of these noises and artifacts are minimized in invasive techniques like ECoG, because signals are measured directly from the brain surface. However their invasive nature has a medical risk and is not possible at any time.

One of the major types of EEG signals have been used in BCIs are Event-Related Potentials (ERPs). ERPs are electrical potential shifts which are time-locked to perceptual, cognitive, and motor events; thus they represent the temporal signature

of macroscopic brain electrical activity. Typical ERPs include the P300, so named because it is characterized by a positive potential shift occurring about 300 ms after stimulus presentation 2.3.

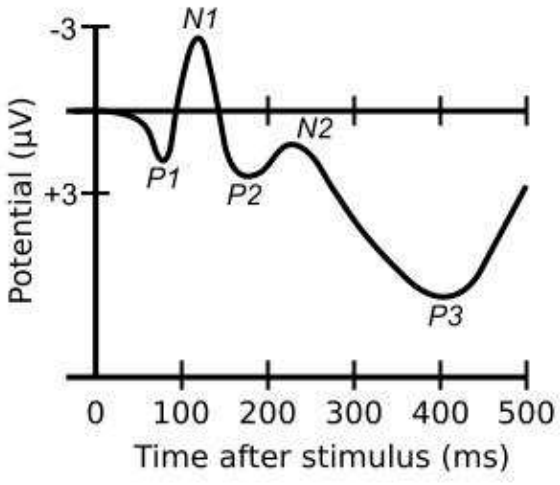

Figure 2.3: ERP Components - A waveform showing several ERP components including the P300 component - Image from [43]

ERP signals are aggregated waveform derived from different EEG measurements over different trials on the same stimulus. Therefore time-locking and Signal to Noise ratio reduction will derives ERP from EEG. Figure 2.4 shows the ERP wave derived from multiple EEG waves. The picture on the left, picture (a), illustrates a single trial EEG signal measured for a visual stimulus. And the picture on the right, picture (b), shows the ERP signal obtained by averaging over 200 EEG trials on the same stimulus.

2.2.2 Pre-processing

Preprocessing methods are typically applied to EEG signals to detect, reduce, or remove noise and artifacts. The goal is to increase the signal-to- noise ratio (SNR) and isolate the signals of interest. One very common but also very subjective approach is to visually screen and score the recordings. In this setting, experts manually mark

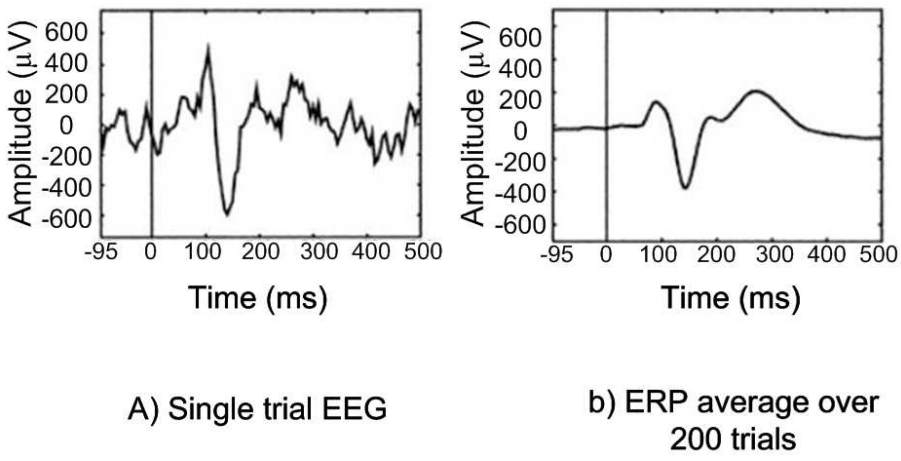

Figure 2.4: ERP vs. EEG - ERP wave form derived from EEG by averaging over 200 trials - Image reprinted from [37].

artifacts and exclude this data from further analysis. While this may work well for offline analysis, visual inspection is not practical during online BCI use.

A number of online preprocessing methods have been utilized in BCIs to date. These include notch filtering (e.g., to filter out 60-Hz power line interference), regression analysis for the reduction of eye movement (EOG) artifacts, and spatial filtering (e.g., bipolar, Laplacian, or common average referencing(CAR)). Spatial filtering aims at enhancing local EEG activity over single channels while reducing common activity visible in several channels.

2.2.3 Feature Extraction

The goal of feature extraction is to transform the acquired and preprocessed EEG signal into a set of p features $x = [x_1, ..., x_p]^T \in X^p$ that are suitable for subsequent classification or regression. In some cases, the preprocessed signal may already be in the form of an appropriate set of features (e.g., the outputs of CSP filters). More typically, some form of time or frequency domain transform is used.

The most commonly used features for EEG are based on frequency domain. Overt and imagined movements typically activate premotor and primary sensori-

motor areas, resulting in amplitude/power changes in the mu ($7 \sim 13$Hz), central beta ($13 \sim 30$Hz) and gamma (30+Hz) rhythms in EEG. These changes can be characterized using classical power spectral density estimation methods such as the digital bandpass filter, the short-term Fourier transform, and wavelet transform. The resulting feature vectors are usually high dimensional because the features are extracted from several EEG channels and from several time segments prior to, during, and after the movement.

The computation of a large number of spectral components can be computationally demanding. Another common and often faster way to estimate the power spectrum is to use adaptive auto-regressive (AAR) [31] models, although the model parameters have to be selected a priori.

Features that are extracted in the time domain include AR parameters [31], (smoothed) signal amplitude parameters [14], the fractal dimension (FD), and common spatial patterns (CSP).

Finally, features estimated in the phase domain have not received much attention in EEG-based BCIs, although coherence and phase-locking values [25] have been explored as potential features.

One of the simplest way of creating the feature space of the data for further analysis is as illustrated in figure 2.5. In this method the signals measured by each electrode is vectorized and concatenated by measurements of the other electrodes during the specific time.

The feature space created by this method will be considerably high dimensional and will face with a problem called *curse of dimensionality* [36]. Curse of dimensionality is the phenomena in the high dimensional data analysis saying, in practice, increasing dimensionality beyond a certain point in the presence of finite number of training samples results in worse, rather than better performance. One of the main reasons for this paradox is that since training sample size is always finite, so the

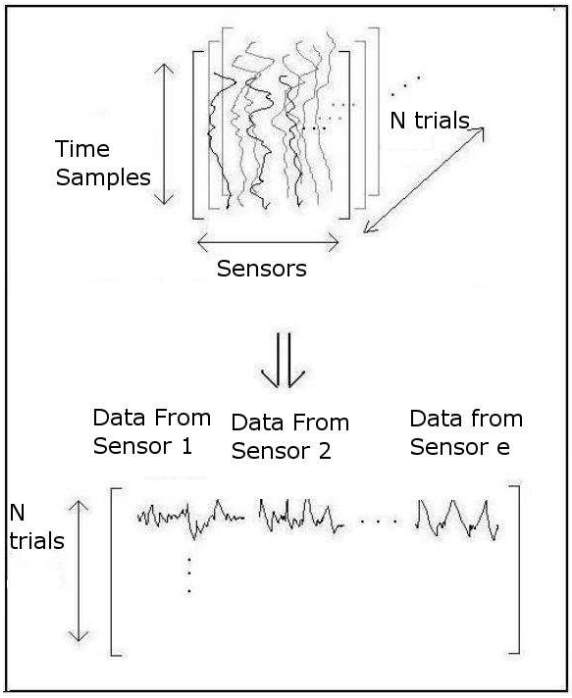

Figure 2.5: Multi-Sensor Feature Space: - The sensor time series are vectorized to generate the feature space [24]

estimation of the class conditional density is incorrect. Analysis of this problem is challenging and subtle. However there many techniques developed to decrease the dimensionality. Some of these techniques are reviewed in the following subsections.

Principal Component Analysis (PCA)

Principal component analysis (PCA) can be developed from many different points of view, but it will be most useful in the context of dimensionality reduction view PCA as an optimization problem.

In general in component analysis methods the features are combined in order to reduce the dimension of the feature space. And basically they project the high dimensional data (i.e EEG recordings) onto a lower dimentional space. Linear combinations are simple to compute and tractable. PCA is one of the classical and most popular approaches for finding optimal transformation. PCA projects the data onto a feature space that best **represents** the data.

In the other hands, PCA tends a linear transformation of a data set that maximizes the variance of the transformed variables subject to orthogonality constraints on the transformation and transformed variables. The transformation is found by solving the following eigenvalue decomposition problem:

$$Sw = \lambda w \tag{2.1}$$

In which S is the scatter matrix and μ is the sample mean of the data as follows:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2.2}$$

$$S = \sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T \tag{2.3}$$

Once the eigenvalues obtained, the transform matrix columns will be equal to those eigenvectors corresponding to biggest eigenvalues. And the principal components are equivalent to the normalized eigenvectors. The figure 2.6 illustrates the linear transformation to the new feature space generated by principal components:

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) searches for those vectors in the underlying space that best **discriminate** or **separate** among classes (rather than those that best describe(represent) the data which is done in PCA).

More formally, given a number of independent features relative to which the data is described, LDA creates a linear combination of these which yields the largest mean differences between the desired classes.

Mathematically speaking, for all the samples of all classes, we define two measures:

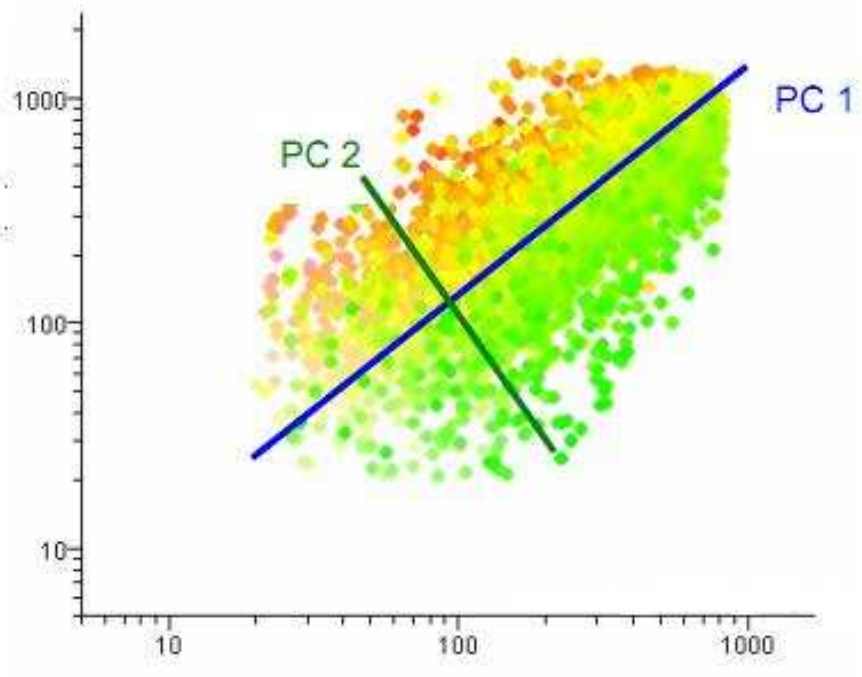

Figure 2.6: PCA - Principal Components Analysis - Image reprinted from [2]

1 one is called within-class scatter matrix, as given by:

$$S_w = \sum_{j=1}^{c} \sum_{i=1}^{N_j} (x_i - \mu_j)(x_i - \mu_j)^T \tag{2.4}$$

where x_i is the ith sample of class j with sample mean μ_j and number of samples as N_j, and c is the number of classes.

2 second measure is called between-class scatter matrix as :

$$S_b = \sum_{j=1}^{c} (\mu_j - \mu)(\mu_i - \mu)^T \tag{2.5}$$

LDA tends to maximize the between class variance while minimizing the within-class variance, therefore the transformation is found by solving the eigenvalue decomposition problem as following:

$$S_w^{-1} S_b w = \lambda w \tag{2.6}$$

A nice comparison of LDA and PCA is explained here [22].

Independent Component Analysis (ICA)

Whereas PCA, extract uncorrelated signals that optimize some criterion, independent component analysis (ICA) is defined as extracting independent signals from a data set.

It is realistic to assume in most of the applications the measurements and observations, the observed signal is a mixture of different source(original) signals, then it is assumed the source signals are independent to each other such that knowing about one doesn't tell us any information about the others. Therefore the observation, x_j can be formulated as the linear combination of n independent signals (components):

$$x_j = \sum_{i=1}^{n} a_{ji} s_i, \tag{2.7}$$

Lets denote the matrix A the matrix with elements a_{ij}, therfore the above mixing model can be written as $x = As$.

The starting point for ICA is the very simple assumption that the components s_i are **statistically independent**. The task is to transform the observed data x, using a linear static transformation W as:

$$s = Wx \tag{2.8}$$

into maximally independent components s. The independence between source signals may be defined in many ways, but it is broadly defined either as minimization of mutual information or maximization of non-Gaussianity.

The goal is now to estimate the mixing matrix A or the corresponding trans-

form matrix W, which is done by the optimization problem of maximizing the non-Gaussianity of the source components s_is or minimizing the mutual information.

2.2.4 Classification

Classification is the problem of assigning one of N labels to a new input signal, given labelled training data of inputs and their corresponding output labels. Regression is the problem of mapping input signals to a continuous output signal. Given the limited spatial resolution of EEG, most BCIs based on EEG rely on classification to generate discrete control outputs (e.g., move a cursor up or down by a small amount by detecting left- versus right- hand motor imagery). BCIs based on neuronal recordings, on the other hand, have utilized regression to generate continuous output signals, such as position or velocity signals for a prosthetic device [27]. Given our emphasis on EEG-based BCIs in this project, we focus primarily on classification methods used in BCIs.

A useful categorization of classifiers into various types (not necessarily mutually exclusive) is suggested in [19]:

- **Generative classifiers** learn a statistical model for each class of inputs. For classification of an input, the likelihood of the input within each class is computed and the most likely class is selected (e.g., Hidden Markov Models (HMMs), Bayesian classifiers).

- **Discriminative classifiers** attempt to learn a boundary that best discriminates between input classes (e.g., linear discriminant analysis (LDA)).

- **Linear classifiers** use a linear function to approximate the boundary between classes. This is the most commonly used type of classifier in BCI applications (e.g., LDA, linear support vector machine (SVM)).

- **Nonlinear classifiers** attempt to learn a nonlinear decision boundary between classes (e.g., k-nearest neighbours (k-NN), kernel SVMs, HMMs).

- **Dynamic classifiers** capture the temporal dynamics of the input classes and use this information for classifying a new input time series of data (e.g., HMMs, recurrent neural networks).

- **Stable classifiers** are those that remain robust to small variations in the training data (e.g., LDA, SVM).

- **Unstable classifiers** are those whose performance can be significantly affected by small variations in the training data (e.g., neural networks).

- **Regularized classifiers** incorporate methods to prevent over-fitting to the training data, allowing better generalization and greater robustness to outliers (e.g., regularized linear discriminant analysis (RDA)).

Support Vector Machines (SVM)

SVM is one the most important supervised learning algorithms used for classification and also regression. SVM is a binary classifier and and therefore assigns class labels -1 or 1 to the new data points. Given a set of training samples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new samples into one category or the other. An SVM model is a representation of the samples as points in space, mapped so that the samples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

More specifically, SVM constructs a hyperplane a high-dimensional space, which can be used for classification or regression. The hyperplane can be define as a linear discriminant function such as:

$$g(x) = a^t x \tag{2.9}$$

In which a is the weight vector and x is the pattern vector. Thus the discriminant hyperplane should satisfy the criterion below for all data patterns x_i:

$$z_i g(x_i) \geq 1 \tag{2.10}$$

In this formula z_i represents the label which can take only value 1 or -1.

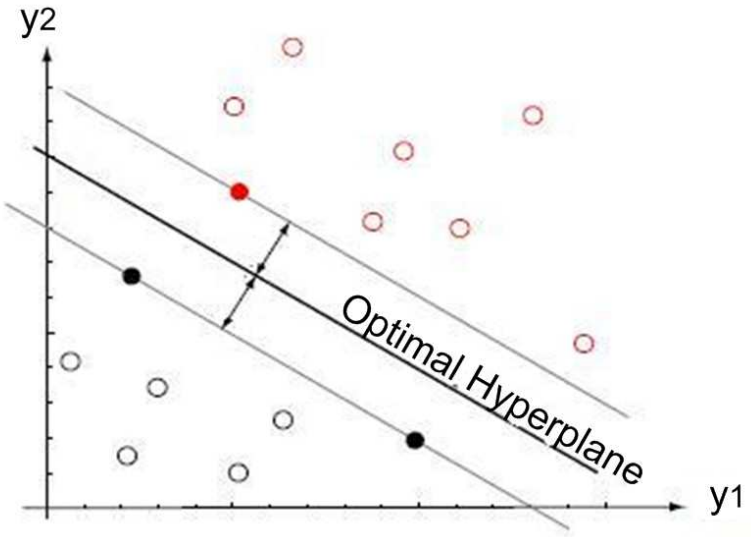

Figure 2.7: SVM - The illustration of Support Vector Machine training - Image reprinted from [2]

As it is illustrated in figure 2.7, the goal in training a Support Vector Machine is to find the separating hyperplane with the largest margin; we expect that the larger the margin, the better generalization of the classifier. Therefore the SVM training can be formulated as an optimization problem as follows:

$$\frac{z_i g(x_i)}{\|a\|} \geq b \tag{2.11}$$

As it is shown in the figure 2.7, for each data point x_i the distance from hyperplane is $z_i g(x_i)$, and assuming that a positive margin b exists, thus the goal is to

find the weight vector a such that it maximizes the margin b and minimizes the size $\|a\|^2$ [7].

Artificial Neural Network (ANN)

An artificial neural network (ANN) is an interconnected group of nodes, similar to the vast network of neurons in the human brain. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. Modern neural networks are non-linear statistical data modelling tools. They are usually used to model complex relationships between inputs and outputs or to find patterns in data.

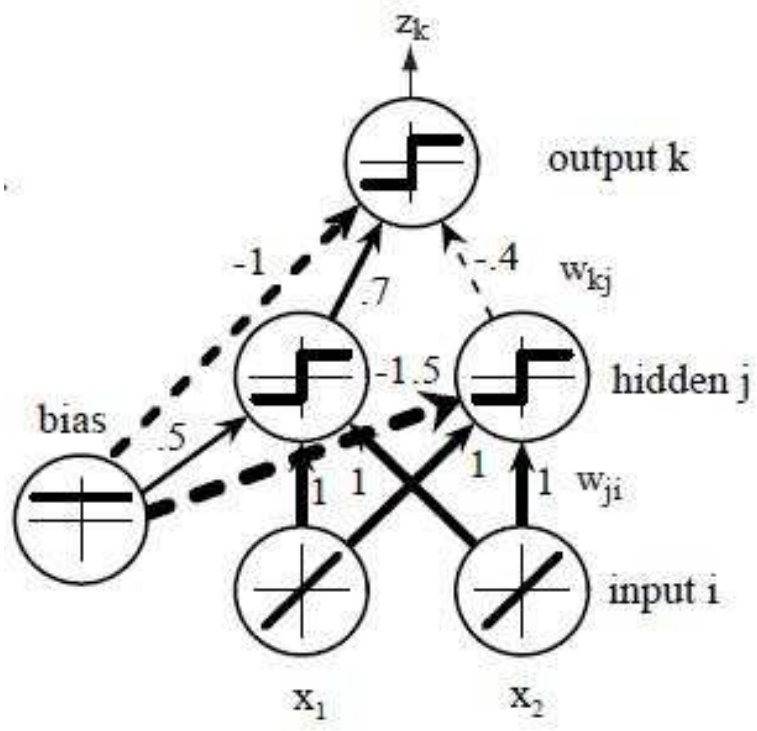

Figure 2.8: ANN - Artificial Neural Networks consist of different layers of node learning the (non-linear) relationship between input (training patterns and output class labels - Image reprinted from [2]

Mathematically, the output of a neuron , node j is a function of the weighted sum of the inputs plus a bias , b_j

$$g_j(x) = f(\sum_{i=1}^{n} w_{ij}x_i + b_{ij}) \tag{2.12}$$

Where the subscript i indexes units on the input layer, j for the hidden; w_{ij} denotes the input-to-hidden layer weights at the hidden unit j.

The function of the entire neural network is simply the computation of the outputs of all the neurons:

$$z_k = f\left(\sum_{j=1}^{c} w_{jk}g_j(x)\right) = f\left(\sum_{j=1}^{c} w_{jk}f(\sum_{i=1}^{n} w_{ij}x_i + b_{ij})\right) \tag{2.13}$$

Here the outputs, z_k, are predicted class labels by the neural network classifier.

The function, f, is called the *activation function*. Majority of ANNs use sigmoid functions, like logistic function(figure 2.9), as their activation function, which is smooth, continuous, and monotonically increasing:

$$f(x) = \frac{1}{1 + e^x} \tag{2.14}$$

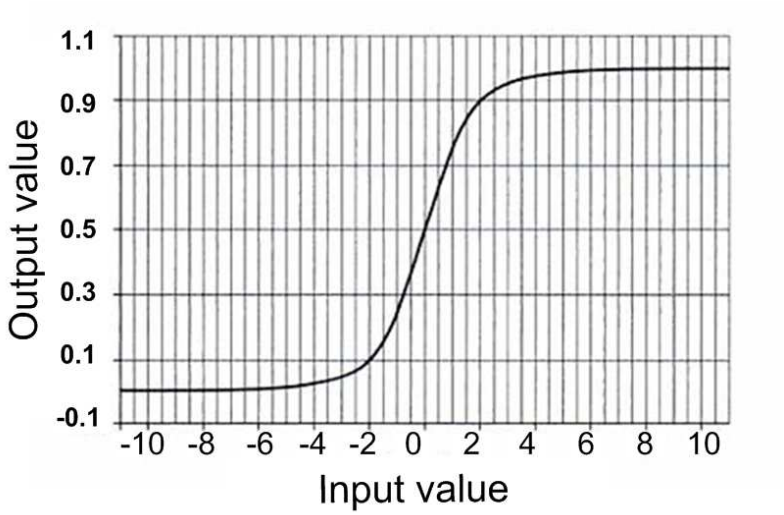

Figure 2.9: Logistic function - The illustration of Logistic function as a common sigmoid function - Image reprinted from [29]

Other sigmoid functions like *hyperbolic tangent* and *arctangent* are also used.

However the exact nature of the function has little effect on the abilities of the neural network.

Training is the act of presenting the network with some sample data and modifying the weights to better approximate the desired function. In order to do so, the neural network is supplied with inputs and the desired outputs and response of the network to the inputs is measured. Then the weights are modified to reduce the difference between the actual and desired outputs.

The process of providing the network with an input and updating the network's weights generally happens in different iterations. Typically many iterations are required to train the neural network.

Usually a criterion is denied to be used during the training process as a metric of updating the weights. The basic approach in learning is to start with an untrained network, present an input training pattern and determine the output. Mostly the training error is defined as the criterion function which is some scalar function of the weights that is minimized when the network outputs match the desired outputs.

The training error is defined as follows:

$$E(w) = \frac{1}{2} \sum_{k=1}^{c} (y_k - z_k)^2 \tag{2.15}$$

Where w represents all weights in the network, and the y_k is the desired output and the z_k is the network output. During the learning process the weights are adjusted to reduce this measure of error.

Hidden Markov Model (HMM)

In problems that have an inherent temporarility – that is, consist of a process that unfolds in time – we may have states at time t that are influenced directly by a state at $t-1$. Hidden Markov models (HMMs) have found greatest use in such problems, for instance speech recognition , gesture recognition or EEG signal processing .

While the notation and description is unavoidably more complicated than the simpler models considered up to this point, we stress that the same underlying ideas are exploited. Hidden Markov models have a number of parameters, whose values are set so as to best explain training patterns for the known category. Later, a test pattern is classified by the model that has the highest posterior probability, i.e., that best explains the test pattern.

More formally, it is assumed that there exists a set of states such as: $\{S_1, S_2, ..., S_N\}$, in which the process moves from one state to another generating a sequence of states as: $S_{i1}, S_{i2}, ..., S_{ik}, ...$

To define a Markov model two probabilities have to be specified: the state transition probability $a_{ij} = P(S_i|S_j)$, and the initial or prior probability of being in a state $\pi_i = P(S_i)$. Hence, the transition and prior matrices are defined as $A = (a_{ij})$ and $\pi = (\pi_{ij})$ respectively.

The fundamental assumption in the Markov Models is that the probability of each subsequent state depends only on what was the previous state, therefore we have:

$$P(S_{ik}|S_{i1}, S_{i2}, ..., S_{ik-1}) = P(S_{ik}|S_{ik-1}) = a_{ik,ik-1} \tag{2.16}$$

In the Hidden Markov Model, it is assumed that states are not visible, but each state randomly generates one of M observations (or visible states): $\{Z_1, Z_2, ...Z_n, ..., Z_N\}$.

To define Hidden Markov model, a matrix B of the observation probabilities should also be defined as :

$$B = (b_i(Z_n)) \tag{2.17}$$

Where

$$b_i(Z_n) = P(Z_n|S_i) \tag{2.18}$$

Thus, a Hidden Markov model is defined as $M = (A, B, \pi)$.

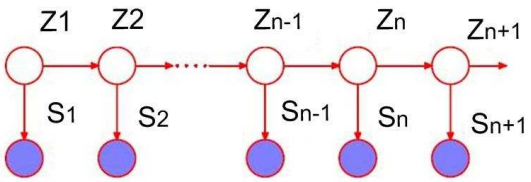

Figure 2.10: HMM - Hidden Markov Model of latent variables - Image reprinted from [2]

Therefore like other learning algorithms, through learning process, given some training observation sequences $Z = \{Z_1, Z_2, ...Z_n, ..., Z_N\}$, figure 2.10, and general structure of HMM (numbers of hidden and visible states), determine HMM parameters, $M = (A, B, \pi)$, that best fit training data. Once the parameters of the model determined, a test pattern is classified by the model that has the highest posterior probability.

Chapter 3

The Emotiv System

The Emotiv System, whose tag-line is "you think, therefore you can", bills itself as a "revolutionary new personal interface for human computer interaction". It is based on the EPOC headset for recording EEG measurements and a software suit which processes and analyzes the data.

Emotiv offers both a consumer headset for $299, which only works with approved applications and a developer headset which supports product development and includes a software bundle for $500. Both headsets are wireless and utilize a proprietary USB dongle to communicate using the 2.4GHz band.

This research used the developer EPOC headset that contains a rechargeable 12 hour lithium battery, 14 EEG sensors (plus CMS/DRL references), and a gyroscope, and has an effective bandwidth of 0.16- 43Hz (Figures 3.1, 3.2).

Emotiv offers 4 different software development kits which grant various control over the EPOC headset API and detection libraries and come with up to 6 different licences: Individual, Developer, Research, Enterprise, Educational Institutes.

This research originally used the Development SDK and later upgraded to the Research Edition. The Research Edition includes the Emotiv Control Panel, Emo-Composer (an emulator for simulating EEG signals), EmoKey (a tool for mapping various events detected by the headset into keystrokes), the TestBench, and an

upgraded API that enables the capture of raw EEG data from each individual sensor [8], [9].

Figure 3.1: Headset - Emotiv EPOC headset- Image reprinted from [9]

3.1 Programming with the Emotiv API

The Emotiv API is exposed as an ANSI C interface implemented in two Windows DLLs: *edk.dll* and $edk_utils.dll$.

The core of the Emotiv SDK is the "EmoEngine", which is a logical abstraction that communicates with the Emotiv headset, receives preprocessed EEG and gyroscope data, manages user-specific or application-specific settings, performs post-processing, and translates the Emotiv detection results into an easy-to-use structure called an EmoState.

Every EmoState represents the current input from the headset including "facial, emotional, and cognitive state" and, with the upgrade to the research edition, contains electrode measurements for each contact. As it is illustrated in the figure 3.3, utilizing the Emotiv API consists of connecting to the EmoEngine, detecting and decoding new EmoStates, and calling code relevant to the new EmoState [9].

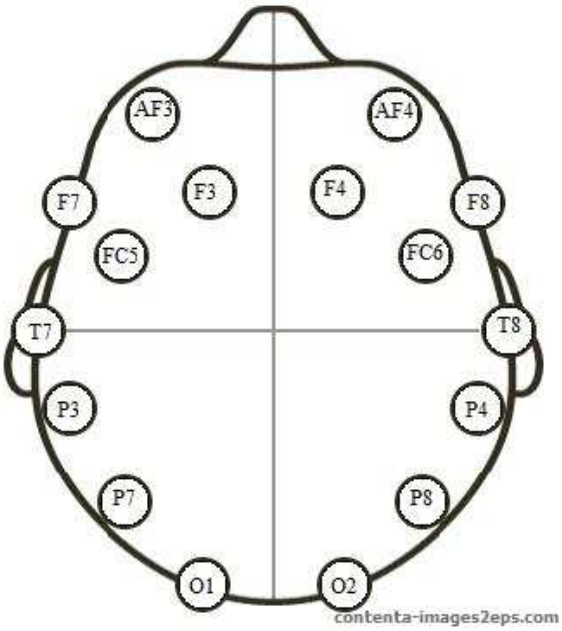

Figure 3.2: Sensors - The 14 EPOC Headset 14 contacts.In addition there is a Common Mode Sense (CMS)electrode in the P3 location and a Driven Right Leg (DRL) electrode in the P4 location which form a feedback loop for referencing the other measurements - Image reprinted from [9]

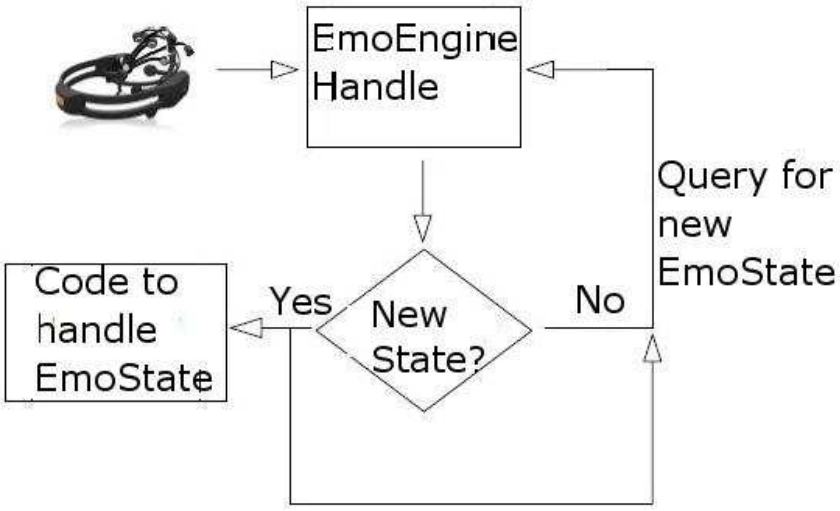

Figure 3.3: Emotiv EPOC API - High-level View of the Utilization of the Emotiv API - Image reprinted from [9]

3.1.1 Detecting Expressive Events

This section demonstrates how an application can use the Expressiv detection suite to control an animated head model called BlueAvatar.

The model emulates the facial expressions made by the user wearing an Emotiv headset. ExpressivDemo connects to Emotiv EmoEngine and retrieves EmoStates for all attached users.

The EmoState is examined to determine which facial expression best matches the users face. ExpressivDemo communicates the detected expressions to the separate BlueAvatar application by sending a UDP packet which follows a simple, predefined protocol.

The Expressiv state from the EmoEngine can be separated into three groups of mutually exclusive facial expressions:

- Upper face actions: Raised eyebrows, furrowed eyebrows
- Eye related actions: Blink, Wink left, Wink right, Look left, Look right
- Lower face actions: Smile, Smirk left, Smirk right, Clench, Laugh

The protocol that ExpressivDemo uses to control the BlueAvatar motion is very simple. Each facial expression result will be translated to plain ASCII text, with the letter prefix describing the type of expression, optionally followed by the amplitude value if it is an upper or lower face action.

Multiple expressions can be sent to the head model at the same time in a comma separated form. However, only one expression per Expressiv grouping is permitted (the effects of sending smile and clench together or blinking while winking are undefined by the BlueAvatar).

Table 3.1, below, excerpts the syntax of some of expressions supported by the protocol. The prepared ASCII text is subsequently sent to the BlueAvatar via UDP socket.

Expressive action type	Corresponding ASCII Text (case sensitive)	Amplitude value
Blink	B	n/a
Wink left	l	n/a
Wink right	r	n/a
Look left	L	n/a
Look right	R	n/a
Eyebrow	b	0 to 100 integer
Smile	S	0 to 100 integer
Clench	G	0 to 100 integer

Table 3.1: Expressive Control Syntax

For example:

- Blink and smile with amplitude 0.5: B,S50
- Eyebrow with amplitude 0.6 and clench with amplitude 0.3: b60, G30
- Wink left and smile with amplitude 1.0: l, S100

ExpressivDemo supports sending expression strings for multiple users. BlueAvatar should start listening to port 30000 for the first user. Whenever a subsequent Emotiv USB receiver is plugged-in, ExpressivDemo will increment the target port number of the associated BlueAvatar application by one. Tip: when an Emotiv USB receiver is removed and then reinserted, ExpressivDemo will consider this as a new Emotiv EPOC and still increases the sending UDP port by one.

In addition to translating Expressiv results into commands to the BlueAvatar, the ExpressivDemo also implements a very simple command-line interpreter that can be used to demonstrate the use of personalized, trained signatures with the Expressiv suite.

Expressiv supports two types of "signatures" that are used to classify input from the Emotiv headset as indicating a particular facial expression. The default signature is known as the universal signature, and it is designed to work well for a large population of users for the supported facial expressions. If the application

or user requires more accuracy or customization, then you may decide to use a trained signature. In this mode, Expressiv requires the user to train the system by performing the desired action before it can be detected. As the user supplies more training data, the accuracy of the Expressiv detection typically improves. If you elect to use a trained signature, the system will only detect actions for which the user has supplied training data. The user must provide training data for a neutral expression and at least one other supported expression before the trained signature can be activated.

Important note: not all Expressiv expressions can be trained. In particular, eye and eyelid-related expressions (i.e. "blink", "wink", "look left", and "look right") can not be trained. The API functions that configure the Expressiv detections are prefixed with *EE_Expressiv*. The *trained_sig* command corresponds to the function called EE_ExpressivGetTrainedSignatureAvailable(), and the *training_exp* command corresponds to another function called EE_ExpressivSetTrainingAction().

It will be useful to first get familiarized with the training procedure on the Expressiv tab in Emotiv Control Panel before attempting to use the Expressiv training API functions.

3.1.2 Detecting Cognitive Events

This section demonstrates how the users conscious mental intention can be recognized by the Cognitive detection and used to control the movement of a 3D virtual object. It also shows the steps required to train the Cognitive suite to recognize distinct mental actions for an individual user.

The design of the Cognitive Demo application is quite similar to the ExpressivDemo covered in previous section. In the last section, ExpressivDemo retrieves EmoStates from Emotiv EmoEngine and uses the EmoState data describing the users facial expressions to control an external avatar. In this section, it is explained

how information about the cognitive mental activity of the users is extracted. The output of the Cognitive detection indicates whether users are mentally engaged in one of the trained Cognitive actions (pushing, lifting, rotating, etc.) at any given time. Based on the Cognitive results, corresponding commands are sent to a separate application called EmoCube to control the movement of a 3D cube.

Commands are communicated to EmoCube via a UDP network connection. The protocol is very simple: an action is communicated as two comma-separated, ASCII-formatted values. The first is the action type returned by *ES_CognitivGetCurrentAction*(), and the other is the action power returned by ES_CognitivGetCurrentActionPower(), as shown in listing A.2 in the appendix A.

Training Cognitive Actions

The Cognitiv detection suite requires a training process in order to recognize when a user is consciously imagining or visualizing one of the supported Cognitiv actions. Unlike the Expressiv suite, there is no universal signature that will work well across multiple individuals. An application creates a trained Cognitiv signature for an individual user by calling the appropriate Cognitiv API functions and correctly handling appropriate EmoEngine events. The training protocol is very similar to that described in the last section to create a trained signature for Expressiv.

To better understand the API calling sequence, an explanation of the Cognitiv detection is required. As with Expressiv, it will be useful to first familiarize yourself with the operation of the Cognitiv tab in Emotiv Control Panel before attempting to use the Cognitiv API functions.

Cognitiv can be configured to recognize and distinguish between up to 4 distinct actions at a given time. New users typically require practice in order to reliably evoke and switch between the mental states used for training each Cognitiv action. As such, it is imperative that a user first masters a single action before enabling two

concurrent actions, two actions before three, and so forth.

During the training update process, it is important to maintain the quality of the EEG signal and the consistency of the mental imagery associated with the action being trained. Users should refrain from moving and should relax their face and neck in order to limit other potential sources of interference with their EEG signal.

Unlike Expressiv, the Cognitiv algorithm does not include a delay after receiving the COG_START training command before it starts recording new training data.

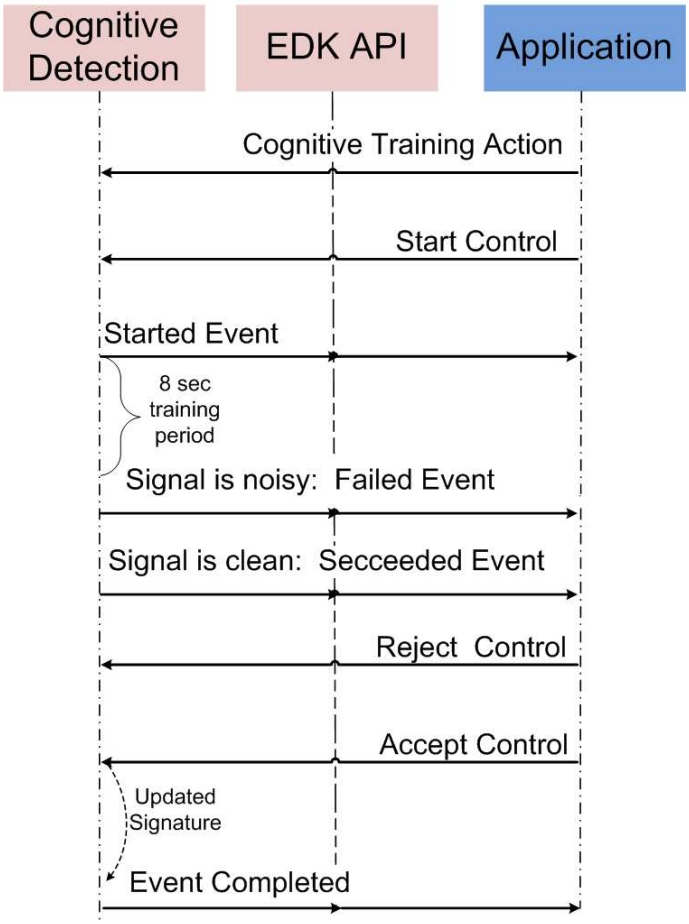

Figure 3.4: EmoEngine Sequence Diagram - Sequence Diagram for communicating with EmoEngine to get the cognitive states - Image reprinted from [9]

The above sequence diagram describes the process of carrying out Cognitiv training on a particular action. The cognitive-specific events are declared as enumerated type EE_CognitivEvent_t in $EDK.h$. Note that this type differs from the EE_Event_t

type used by top-level EmoEngine Events. The code snippet in listing A.3 illustrates the procedure for extracting Cognitive-specific event information from the EmoEngine event.

Before the start of a training session, the action type must be first set with the API function EE_CognitivSetTrainingAction(). In EmoStateDLL.h, the enumerated type EE_CognitivAction_t defines all the Cognitiv actions that are currently supported (COG_PUSH, COG_LIFT, etc.). If an action is not set before the start of training, COG_NEUTRAL will be used as the default.

EE_CognitivSetTrainingControl() can then be called with argument COG_START to start the training on the target action.

In EDK.h, enumerated type EE_CognitivTrainingControl_t defines the control command constants for Cognitiv training. If the training can be started, an event of type EE_CognitivTrainingStarted will be sent almost immediately. The user should be prompted to visualize or imagine the appropriate action prior to sending the COG_START command.

The training update begin after the EmoEngine sends the EE_CognitivTrainingStarted event. This delay will help to avoid training with undesirable EEG artifacts resulting from transitioning from a "neutral" mental state to the desired mental action state.

After approximately 8 seconds, two possible events will be sent from the EmoEngine: EE_CognitivTrainingSucceeded: If the quality of the EEG signal during the training session was sufficiently good to update the algorithms trained signature, EmoEngine will enter a waiting state to confirm the training update, which will be explained below. EE_CognitivTrainingFailed: If the quality of the EEG signal during the training session was not good enough to update the trained signature then the Cognitiv training process will be reset automatically, and user should be asked to start the training again. If the training session succeeded (EE_CognitivTrainingSucceeded was received) then the user should be asked whether to accept or reject the session.

The user may wish to reject the training session if he feels that he was un-

able to evoke or maintain a consistent mental state for the entire duration of the training period. The users response is then submitted to the EmoEngine through the API call EE_CognitivSetTrainingControl() with argument COG_ACCEPT or COG_REJECT.

If the training is rejected, then the application should wait until it receives the EE_CognitivTrainingRejected event before restarting the training process.

If the training is accepted, EmoEngine will rebuild the users trained Cognitiv signature, and an EE_CognitivTrainingCompleted event will be sent out once the calibration is done. Note that this signature building process may take up several seconds depending on system resources, the number of actions being trained, and the number of training sessions recorded for each action.

To test Cognitive detection capability, you need to launch the Emotiv Control Panel and the EmoComposer. In the Emotiv Control Panel select Connect→To EmoComposer and accept the default values and then enter a new profile name. Navigate to the /example4/EmoCube folder and launch the EmoCube, enter 20000 as the UDP port and select **Start Server**. Start a new instance of CognitivDemo, and observe that when you use the Cognitiv control in the EmoComposer the EmoCube responds accordingly.

Next, experiment with the training commands available in CognitivDemo to better understand the Cognitive training procedure described above. Listing A.4 shows a sample CognitivDemo session that demonstrates how to train.

3.1.3 Working with EEG raw Data

This section demonstrates how to extract live EEG data using the EmoEngine. Data is read from the headset and sent to an output file for later analysis. This examples only works with the SDK versions that allow raw EEG access (Research, Education and Enterprise Plus).

The process starts in the same manner as the earlier one. A connection is made to the EmoEngine through a call to EE_EngineConnect(), or to EmoComposer through a call to EE_EngineRemoteConnect(). The EmoEngine event handlers and EmoState Buffers are also created as before.

To initiate retrieval of the latest EEG buffered data, a call is made to DataUpdateHandle(). When this function is processed, EmoEngine will ready the latest buffered data for access via the $hDatahandle$. All data captured since the last call to DataUpdateHandle will be retrieved. Place a call to DataGetNumberOfSample() to establish how much buffered data is currently available. The number of samples can be used to set up a buffer for retrieval into your application as shown.

Finally, to transfer the data into a buffer in our application, we call the EE_DataGet function. To retrieve the buffer we need to choose from one of the available data channels:

ED_COUNTER,ED_AF3, ED_F7, ED_F3, ED_FC5, ED_T7,
ED_P7, ED_O1, ED_O2, ED_P8, ED_T8, ED_FC6, ED_F4,
ED_F8, ED_AF4, ED_GYROX, ED_GYROY, ED_TIMESTAMP,
ED_FUNC_ID, ED_FUNC_VALUE, ED_MARKER, ED_SYNC_SIGNAL

For example, to retrieve the first sample of data held in the sensor $AF3$, place a call to EE_DataGet as follows:

$$EE\_DataGet(hData, ED\_AF3, databuffer, 1);$$

You may retrieve all the samples held in the buffer using the bufferSizeInSample parameter. Finally, we need to ensure correct clean up by disconnecting from the EmoEngine and free all associated memory. $EE\_EngineDisconnect()$;
$EE\_EmoStateFree(eState)$;

$EE\_EmoEngineEventFree(eEvent)$;

There is also an Emotiv useful tool which allows user to monitor the EEG signals captured by each channel in real time. This tool called TestBench as shown in figures 3.5 and 3.6, can be used as the ground truth of raw data captured by SDK. Also it is really useful to see the patterns (waveforms) associated with events in real-time, figures 3.5 and 3.6. The testBench can also be used for recording the data in the EDF format which is compatible with some of the tool for offline analysis of EEG data such EEGLab [5].

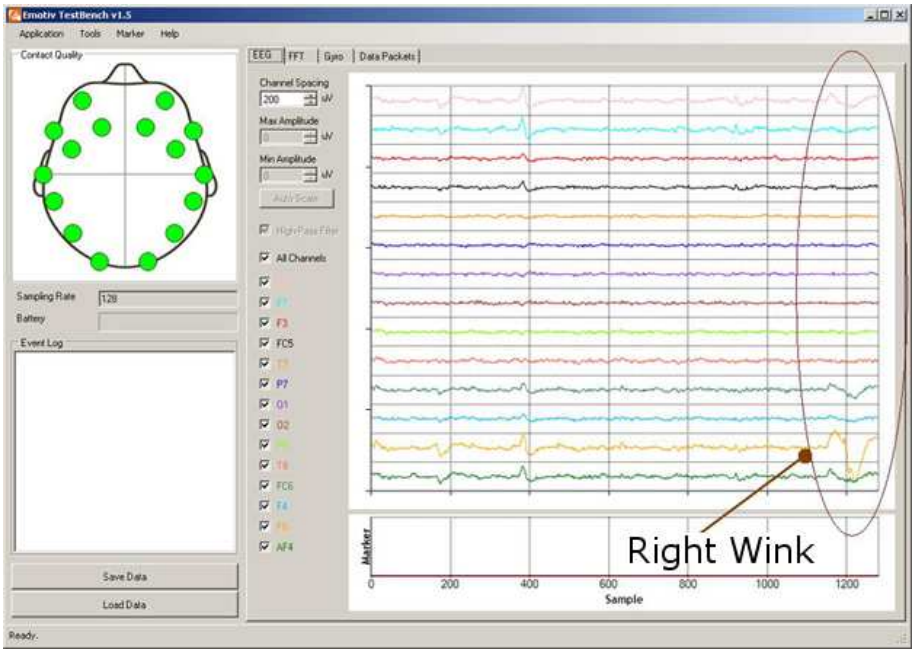

Figure 3.5: Right Wink - Right Wink wave form can be detected easily- Image Captured from Emotiv TestBench Application

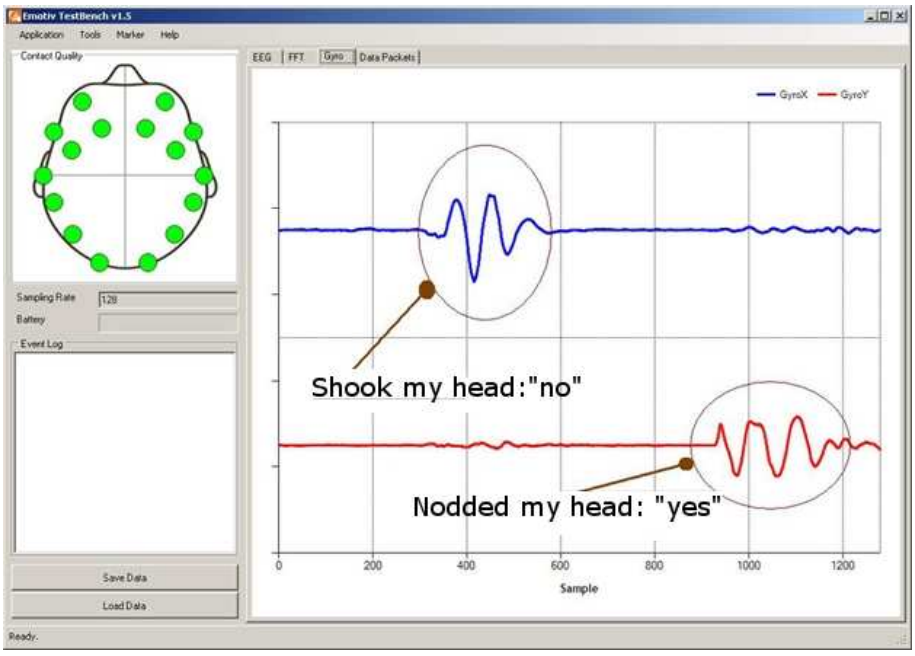

Figure 3.6: Head Movement - Head Movement is the cleanest signal captured by the headset - Image Captured from Emotiv TestBench Application

Chapter 4

Experiments

4.1 Training Emotiv engine and Dial a phone by thoughts

This section explains the experiment performed to create a Brain-Machine Interface to control a mouse cursor in a 2D environment. This experiment is done based on the Emotiv SDK Beta release 1.0.x [9].

In this experiment both Cognitiv and Expressiv events as well as Gyro data are exploited. Two movement actions in different directions, horizontal and vertical, as cognitive actions are trained. The training was performed during two months each training session involves different trials in which each of them performed approximately during 8 seconds. The corresponding events are then recognized by system with their corresponding action power. Imagined horizontal and vertical movements are translated to real horizontal and vertical cursor movements to dial a phone keypad. The keypad user interface is implemented by Visual C++ windows form.

Blink is one of the expressive events that is exploited and translated to the mouse click. Therefore the user can dial the phone just by imagining the horizontal (i.e

Figure 4.1: Brain Keypad - A keypad dialled by brain commands

move left.right) or vertical(i.e up/down) movement to be able to move the cursor over the desired number and by just doing a blink he/she can click on the number and then move on over next digit (Figure 4.1).

It is really important to maintain the quality of the EEG signal and the consistency of the mental imagery associated with the action being trained and performed. This is one the issues during the experiment. The user has to refrain from moving and should relax their face and neck in order to limit other potential sources of interference and additive artifacts with the cognitive EEG signal.

It is possible to add extra actions (i.e having two distinct left and right actions instead of just having one rounded horizontal action), however by adding more actions the complexity of training and also the detection skill of system drops drastically and a lot of extra trainings requires to boost up the detection rate of the system. The reason for this phenomenon is that because this is classification problem, the feature space will be divided to different areas containing training samples for different actions. Therefore once new actions are added to the cognitive training (push, pull, lift, spin), the software will often see that there is an overlap in detections and

as a result may downgrade the training on the existing actions. As the the actions are retrained and more clarity developed in each, the synaptic pathways for each will become more distinct along with the voltage traces for the EEG detections.

As discussed in chapter 3, in addition to the actions, there is also a background state trained and called "neutral". One of the most important things is that with a well trained Neutral State, everything else benefits. One way to do this is to record a neutral state while watching TV, playing a game or having a conversation. This will create a very active neutral state for the detections to differentiate against and make it simpler for the BCI to see the changes in state that indicate different cognitive actions. As the system learns and refines the signatures for each of the actions, as well as neutral, detections become more precise and easier to perform.

As the result, dialling a 5-digit number took 59.4 seconds, that is 11.9 seconds per digit on average. Some portion of this time is due to the delay caused by the Emotiv engine as the response time to the accusation, sampling rate, pre-processing and detection; and some portion of it is due to the accuracy of system in detecting the right direction to move on over the desired digit, that is once the detection is wrong the user needs to try once more which causes further delays, each correct detection is confirmed by a click that is associated with the blink. It worth mentioning that time needed to detect the blink is also a portion of the time reported, whereas blink is an expressive event and its detection is almost instantaneous and pretty accurate, time corresponding to its detection is neglectable, therefore this time shows the average detection time of cognitive events of movement directions.

The second delay can be improved by more training to increase the accuracy (actions' skill rate) of system. However the first delay is an intrinsic property of the Emotiv engine and cannot be improved that much by training.

Chapter 5

Summary and Future Work

In this project we presented a brain-computer interface model that recognizes one task from another in a timely manner and helps us to understand the underling structure of the information concealed in the EEG signals.

We explored the area of brain-computer interface and related fields such as signal processing, cognitive science and machine learning. Different techniques, tools and algorithm in each of these areas that are useful in the analysis of EEG signals are studied and summarized. Algorithms like PCA and LDA for feature extraction and different classifiers like SVM, Neural Network or Markov Models. We performed many experimental studies to demonstrate how different mathematical and software tools can be exploited to accomplish a real brain computer interface.

During the experiments we examined the different capabilities and functionalities of the Emotiv EPOC system [8] as one of the popular platforms for brain-computer interface. We understood how to train the engine by the desired tasks to build a whole BCI system upon the trained engine. It is also carefully investigated how to program by using APIs associated with the EPOC's SDK to get the EEG raw data for further mining by new learning algorithms.

However, this research direction has much to be extended further. As one aspect, the analysis of Emotiv EPOC raw data based on the algorithms explained in the

chapter 2 has a high potential of further studies. This will help to generate a robust and real brain-machine interface controlled based on current data acquisition device. This work requires a long term data recording under the whole cognitive and psychological circumstances to end up with accurate and clean data in advance of applying any analysis.

From the machine learning and pattern recognition perspective, the mathematical part of this work has also a high potential and demand for extension in the future. The algorithms introduced in the second part in the chapter 4 opens many areas of study by looking at non-deterministic and uncertainty of information concealed in the raw data. This part can be coupled by more advance feature extraction techniques, like ICA [4] and non-linear feature learning [13], to cancel noises and get cleaner signals in a smaller and more informative feature space.

In this work we showed that given data recording device, and statistical tools, how to accomplish a brain-machine interface. Most of these works are not restricted to BCI and can be used in a variety of sensing and sensory systems requiring signal detection and estimation. But yes, further research on information fusion, feature extraction, and supervised and unsupervised learning is needed. In addition non-trivial signal recording as well as data aggregation and experimental studies need to be considered to successful system development of this challenging and exciting technology.

APPENDICES

Appendix A

Code Snippets

Listing A.1: Sample code from ExpressivDemo code - [9]

```
        ...
EE_ExpressivAlgo_t upperFaceType =
ES_ExpressivGetUpperFaceAction(eState);
EE_ExpressivAlgo_t lowerFaceType =
ES_ExpressivGetLowerFaceAction(eState);
float upperFaceAmp =
        ES_ExpressivGetUpperFaceActionPower(eState);
float lowerFaceAmp =
        ES_ExpressivGetLowerFaceActionPower(eState);
```

Listing A.2: Cognitiv Detection - [9]

```
void sendCognitivAnimation(SocketClient& sock,
                           EmoStateHandle eState)
{
  std::ostringstream os;
  EE_CognitivAction_t actionType;
  actionType = ES_CognitivGetCurrentAction(eState);
  float actionPower;
  actionPower = ES_CognitivGetCurrentActionPower(eState);
  os << static_cast<int>(actionType) << ","
     << static_cast<int>(actionPower*100.0f);
  sock.SendBytes(os.str());
}
```

Listing A.3: Sample code from CognitivDemo code - [9]

```
EmoEngineEventHandle eEvent = EE_EmoEngineEventCreate();
if (EE_EngineGetNextEvent(eEvent) == EDK_OK) {
  EE_Event_t eventType = EE_EmoEngineEventGetType(eEvent);
  if (eventType == EE_CognitivEvent) {
      EE_CognitivEvent_t cEvt =
          EE_CognitivEventGetType(eEvent);
          ...
  }
          ...
}
```

Listing A.4: Cognitive Action(i.e. push and neutral) Training Session - [9]

```
CognitivDemo> set_actions 0 push lift
==> Setting Cognitiv active actions for user 0...
CognitivDemo> Cognitiv signature for user 0 UPDATED!
CognitivDemo> training_action 0 push
==> Setting Cognitiv training action for user 0 to "push"...
CognitivDemo> training_start 0
==> Start Cognitiv training for user 0...
CognitivDemo> Cognitiv training for user 0 STARTED!
CognitivDemo> Cognitiv training for user 0 SUCCEEDED!
CognitivDemo> training_accept 0
==> Accepting Cognitiv training for user 0...
CognitivDemo> Cognitiv training for user 0 COMPLETED!
CognitivDemo> training_action 0 neutral
==> Setting Cognitiv training action for user 0 to "neutral"
CognitivDemo> training_start 0
==> Start Cognitiv training for user 0...
CognitivDemo> Cognitiv training for user 0 STARTED!
CognitivDemo> Cognitiv training for user 0 SUCCEEDED!
CognitivDemo> training_accept 0
==> Accepting Cognitiv training for user 0...
CognitivDemo> Cognitiv training for user 0 COMPLETED!
CognitivDemo>
```

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] N. Birbaumer et al. The thought translation device (ttd) for completely paralyzed patients. *IEEE Transactions on Rehabilitation Engineering*, 8(2):190–193, June 2000.

[2] C.M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, 2006.

[3] Benedict Carey. Monkeys think moving artifiacl arm as own. *The New York Times*, 29, May 2008.

[4] Pierre Comon. Independent component analysis, a new concept? *Signal Process.*, 36:287–314, April 1994.

[5] A. Delorme, C. Kothe, A. Vankov, N. Bigdely-Shamlo, R. Oostenveld, T. Zander, and S. Makeig. Matlab-based tools for bci research. In D. S. Tan and A. Nijholt, editors, *Brain-Computer Interfaces: Applying our Minds to Human-Computer Interaction*. 2010.

[6] Katie Drummond. Pentagon preps soldier telepathy push. *Wired Magazine*, 14, 2009.

[7] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern classification*. Pattern Classification and Scene Analysis: Pattern Classification. Wiley, 2001.

[8] Emotiv, July 2011.

[9] Emotiv. *Software Development Kit(SDK) User Manual*, beta release 1.0 edition, 2011.

[10] G.E. Fabiani, D.J. McFarland, J.R. Wolpaw, and G. Pfurtscheller. Conversion of eeg activity into cursor movement by a brain-computer interface (bci). *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 12(3):331 –338, sept. 2004.

[11] E. Fetz. Operant conditioning of cortical unit activity. *Science*, 163:955–958, 1969.

[12] F. Galan et al. a brain-actuated wheelchair: Asynchronous and non-invasive brain-computer interfaces for continuous control of robots. *Clinical Neurophysiology. Volume*, 119:2159–2169, 2008.

[13] D. Garrett, D.A. Peterson, C.W. Anderson, and M.H. Thaut. Comparison of linear, nonlinear, and feature selection methods for eeg signal classification. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 11(2):141 –144, june 2003.

[14] B Hjorth. An on-line transformation of eeg scalp potentials into orthogonal source derivations. *Electroencephalogr Clin Neurophysiol*, 39(5):526–30, 1975.

[15] Leigh R. Hochberg et al. Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature. Vol*, 442:164–171, 2006.

[16] L. Kauhanen, T. Palomäki, P. Jylänki, F. Aloise, Marnix Nuttin, and José del R. Millán. Haptic feedback compared with visual feedback for bci. In *Proceedings of the 3rd International Brain-Computer Interface Workshop and Training Course 2006*, Graz, Austria, 9 2006.

[17] Philip R. Kennedy et al. activity of single action potentials in monkey motor cortex during long-term task learning. *Brain Research*, 760:251–254, June 1997.

[18] Sander Koelstra, Ashkan Yazdani, Mohammad Soleymani, Christian Muehl, Jong-Seok Lee, Anton Nijholt, Thierry Pun, Touradj Ebrahimi, and Ioannis Patras. Single trial classification of eeg and peripheral physiological signals for recognition of emotions induced by music videos. In *Proceedings of the International Conference on Brain Informatics*, 2010.

[19] F Lotte, M Congedo, A Lcuyer, F Lamarche, and B Arnaldi. A review of classification algorithms for eeg-based braincomputer interfaces. *Journal of Neural Engineering*, 4(2), 2007.

[20] Shijian Lu et al. Unsupervised brain-computer interface based on inter-subject information. *30th Annual International IEEE EMBS Conference. Vancouver, British Columbia, Canada.*, 20-24, 2008.

[21] Kip A. Ludwig et al. *Employing a Common Average Reference to Improve Cortical Neuron Recordings from Microelectrode Arrays.* Journal of Neurophysiology, September 3rd, 2008.

[22] A.M. Martinez and A.C. Kak. Pca versus lda. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(2):228 –233, feb 2001.

[23] David Millett. Hans berger: from psychic energy to the eeg. *Perspectives in Biology and Medicine, Johns Hopkins University Press*, 44.(4):522–542, 2001.

[24] Mohammad-Mahdi Moazzami. EEG pattern recognition, toward brain-computer interface, 2011.

[25] M Naeem, C Brunner, R Leeb, B Graimann, and G Pfurtscheller. Seperability of four- class motor imagery data using independent components analysis. *Journal of Neural Engineering*, 3(3):208–216, 2006.

[26] Ernst Niedermeyer and Fernando Lopes da Silva. Electroencephalography: Basic principles, clinical applications, and related fields. *Lippincott Williams and Wilkins*, page 140, 2005.

[27] K.G. Oweiss. *Statistical Signal Processing for Neuroscience and Neurotechnology*. Academic Press. Elsevier Science & Technology, 2010.

[28] Nunez PL and Srinivasan R. *Electric Fields of the Brain: The Neurophysics of EEG*. Oxford University Press, 1981.

[29] D. Pyle. *Data preparation for data mining*. Number v. 1 in The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, 1999.

[30] A. James Rowan. *Primer of EEG*. Elsevier Science, Philadelphia, PA, 2003.

[31] Alois Schlgl. *The electroencephalogram and the adaptive autoregressive model: theory and applications*. Berichte aus der medizinischen Informatik und Bioinformatik. Shaker, 2000.

[32] Eric W. Sellers et al. A p300 event-related potential brain-computer interface (bci): The effects of matrix size and inter stimulus interval on performance. *Biological Psychology*, 73:242–252, October 2006.

[33] P. Shenoy, K.J. Miller, J.G. Ojemann, and R.P.N. Rao. Generalized features for electrocorticographic bcis. *Biomedical Engineering, IEEE Transactions on*, 55(1):273 –280, jan. 2008.

[34] Pradeep Shenoy, Matthias Krauledat, Benjamin Blankertz, Rajesh P. N. Rao, and Klaus-Robert Müller. Towards adaptive classification for bci. *Journal of Neural Engineering*, 3, March 2006.

[35] B. E. Swartz and E. S. Goldensohn. Timeline of the history of eeg and associated fields. *Electroencephalography and clinical Neurophysiology*, 106(2):173–176, February 1998.

[36] P.N. Tan, M. Steinbach, and V. Kumar. *Introduction to data mining*. Pearson International Edition. Pearson Addison Wesley, 2006.

[37] Lucy Troup. Electroencephalogram (eeg) and event-related potentials (erp), January 2008. Presentation on CSU CSU Symposium on Imaging.

[38] J. Vidal. Toward direct brain-computer communication. *Annual Review of Biophysics and Bioengineering*, 2:157–180, 1973.

[39] Carmen Vidaurre, Claudia Sannelli, Klaus-Robert Mller, and Benjamin Blankertz. Machine-learning-based coadaptive calibration for brain-computer interfaces. *Neural Computation*, 23(3):791–816, 2011.

[40] Johan Wessber et al. Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. *Nature*, 408(6810):361–365, 2000.

[41] Wikipedia. Parietal lobe — wikipedia, the free encyclopedia. [accessed 2-July-2011].

[42] Wikipedia. Electroencephalography — wikipedia, the free encyclopedia, 2009. [accessed 10-July-2011].

[43] Wikipedia. Event-related potential — wikipedia, the free encyclopedia, 2009. [accessed 10-July-2011].