This Pdf version of my thesis is identical to the official paper copy, except for a few small details: References and indexes are clickable hyperlinks (something impossible to achieve with the current print technology), the abstract has been revised, and a couple of typos that I noticed after printing have been corrected.

B. D.

Politecnico di Milano
Dipartimento di Elettronica e Informazione
Dottorato di Ricerca in Ingegneria Informatica e Automatica

Toward An Integrated P300- And ErrP-Based Brain-Computer Interface

Tesi di dottorato di:
Bernardo Dal Seno

Relatore:
Prof. Matteo Matteucci
Correlatore:
Prof. Luca Mainardi
Tutore:
Prof. Andrea Bonarini
Coordinatore del programma di dottorato:
Prof. Patrizio Colaneri

2009 — XX ciclo

Politecnico di Milano
Dipartimento di Elettronica e Informazione
Piazza Leonardo da Vinci 32 I 20133 — Milano

Politecnico di Milano
Dipartimento di Elettronica e Informazione
Dottorato di Ricerca in Ingegneria Informatica e Automatica

Toward An Integrated P300- And ErrP-Based Brain-Computer Interface

Doctoral Dissertation of:
Bernardo Dal Seno

Advisor:
Prof. Matteo Matteucci
Coadvisor:
Prof. Luca Mainardi
Tutor:
Prof. Andrea Bonarini
Supervisor of the Doctoral Program:
Prof. Patrizio Colaneri

2009 — XX edition

Acknowledgments

This thesis stems from almost four years of work in my PhD course. While it is mine, as my name on the front cover implies, there are a lot of other people whose names don't appear on the cover but that made this thesis possible and influenced the way it is. As the front cover is too small to mention them all, I decided to put the names of (some of) them here. I'm very grateful to all the people that accompanied me in these part of my journey, even to those whose name didn't find a place here, and hope that they could continue to be company to me. Also many circumstances affected my work in the PhD program. Some of them were good, and some were bad, but in the end, it seems to me that their cumulative effect has been to improve the quality of the work.

Many people (and institutions) made my work at the Department of Electronics and Information and my (hopefully) imminent PhD graduation possible. **Politecnico di Milano** and in particular the **Department of Electronics and Information** hosted me; though the bureaucracy involved in working here is still beyond my level of comprehension, the environment has been friendly enough to let me enjoy my job. Thanks to all the people, also those "in the background", who made that possible. The **IIT** (Italian Institute of Technology) payed for my scholarship and some of the instruments I used. **Matteo** supervised my work with many pieces of advice and guidance; though he's been always rather busy with many other things, I'd like to thank him for being present even when he was absent or busy (though that involved looking for him very hard or meeting after midnight on ICQ). **Luca** gave many suggestions for the signal analysis and helped with the articles derived from the present work (and my gratitude will not fade if they should reject the latest articles we submitted). **José Millán** read my thesis draft very attentively and sent me many useful observations that helped me in improving my work. **EBNeuro** provided us with their BE Light electroencephalograph and helped us in interfacing it with the software we used; I'd like to thank in particular **Paolo Giovagnola**, **Romana Sgalla**, **Simone Pola**, and **Vincenzo Roma** for their assistance. Several students helped me in my work with the BCI and the wheelchair; I want to remember in particular **Francesco**, **Gianmaria**, **Andrea**, and **Simone**, whose work contributed directly to the results in my thesis. **Franco** and **Stefano** kindly provided us with EEG data they recorded at the S. Camillo Hospital. Many **students of AirLAB** have participated as subjects in our experiments, at great risk of their life (seriously, no students were harmed in the course of this thesis work, and no students ever complained, not even after graduation); I'm thankful for their patience and collaboration. Many researchers and scientists have also contributed to this work with their discoveries and insights, almost always unwittingly. Most of them have been credited in the references, and I apologize if anyone thinks that their wonderful work should have been cited here and I didn't mention it. A PhD thesis is a big work, but I can only devote a finite amount of time and efforts to it, so there is always something missing. (And that is precisely the reason why you may find many errors in the text).

During my work I've used many software tools and components, and most of them have been made available freely by their authors. I'm grateful to them for their work and for their decision to share it. Here I want to cite the pieces of software that have been most useful or that I've enjoyed the most (in random order): **Linux** (don't know about world domination, but it's pretty dominating all the computers on my desks; by the way, I use **Debian**), **LaTeX** (writing is quite a chore to me, and I'm glad that there are tools like this that help in the technical aspects), **Biosig**, **Weka**, **Emacs** the operating system in an editor (sorry, I tried Vi, but I've not yet learned to do a copy-and-paste operation), **Firefox**, **BCI2000**, **Python**, **Gcc**. Also, I've found many useful resources over the **Internet** (one of the greatest inventions of the last century), which are too many to list (also because I forgot of many of them); I just mention **Google**, which has been very handy in many circumstances, and **Wikipedia**, from which I took some of the images for this thesis, and which is very convenient when you need a quick introduction to something new.

Many more people accompanied me and contributed *indirectly* to this thesis. **My parents** have been always supporting me in many ways. Grazie **papà** e **mamma** per tutto, anche e soprattuto per tutte quelle cose che ho sempre dato per scontato ma non lo sono affatto! **Federico**'s friendship surely had a big role in orienting me towards getting back to the academia and pursue a PhD after a few years of work in a company. Also **Diego** had an important role, probably without knowing. When I entered the Department of Electronics and Information four years ago I didn't know anyone. I've made the acquaintance of many people, also in other departments, and with time many friendships (of different degrees, of course) were born. This list is long (in almost random order): **Simone**,

Rossella, **Luigi**, **Davide**, **Giulio**, **Daniele**, **Lin**, **Ahmed**, **Davide**, **David**, **Andrea**, **Giacomo**, **Nadia**, **Alessandro**, **Roberto**, **Jonatha**, **Pigi**, **Alessandro**, **Simone**, **Ernestina**, **Antonella**, **Marco David**, **Peri**, **Marco**, **Paolo**, **Gianmaria**, **Pedra**, **Alberto**, **Manu**, **Marco**, **Checco** (and all **the others** that I forgot with my mind but not with my heart). Many other friends have been close to me in these last four years of my life, some for a long time and some for a shorter time. I'm very grateful to all of them for their friendship (which — I know — requires a lot of patience), but I don't want to make a long exhaustive list of friends here, both because the day after printing I would realize that I forgot of someone (never write acknowledgments at the last minute!), and probably nobody would care anyway of having their names here. I just mention a few names without further notices, as their respective proprietors should know why they had the venture of being mentioned here (the list is in grouped random order): **Maurizia**, **Davide**, **Irene**, **Viviana**, **Claudio**, **Laura**, **Luca**, **Marilena**, and the **Jedis**.

J. R. R. Tolkien has contributed to this thesis in more than one way. Besides being the author of over than twelve quotations throughout the text, his books have been a great source of inspiration for me. Not for my writing style or my English, as you may notice, but because books like *The Lord Of The Rings* or *The Silmarillion* are books that speak about each of us and our lives. The most beautiful passage I've ever read in a novel is in *The Lord Of The Rings*, and it is the dialog between Frodo and Sam when they are on the stairs of Cirith Ungol (towards the end of the Chapter 8 of Book IV). Two of the quotations reported in my thesis have been taken from that passage. When you read about Frodo and his dangerous journey through the dark land of Mordor, you read about your life and your challenges (and during the long lonely nights when I was writing the first draft of this thesis, I really identified myself with Frodo), and even though the destiny of the world doesn't lie on your shoulders and you're not risking your life, you can feel that your life is greater than the sum of the things you do, or the value of your salary, and the real danger is that your life becomes dull and without meaning. And I've been blessed with the company of many friends who continuously show me how to avoid such danger and remind me of what is really important. And that's saying a lot.

Abstract

This thesis presents a work in the brain-computer interface (BCI) field that makes use of machine learning and statistics techniques. A BCI is a device that bypasses any muscle or nerve mediation, and it interfaces directly with the brain by measuring signals generated by its activity. Potentially, it could be helpful for people that cannot use "standard" interfaces, such as people affected by a disease that impairs limb movements, or people engaged in physical activities. Machine learning is a broad field of artificial intelligence that deals with techniques used to endow a machine (a computer) with the ability to adapt its behavior to different and possibly changing conditions.

The work focuses mainly on BCIs based on potentials discernible in EEG (electroencephalography) recordings, such as P300 and error potentials. A P300 occurs when a subject detects an occasional target (oddball) stimulus in a regular train of standard stimuli. An error potential can be seen when a subject makes a mistake, and, more relevant to BCI applications, when the machine the subject is interacting with does not behave as the user expects. In a P300-based BCI the user is presented with some possible choices, and they are highlighted one at a time. A user directs his attention to one choice, and thus every highlighting of that choice elicits a P300 potential. Error potentials can be exploited as a way to automatically detect (and possibly correct) BCI errors.

After an overview of some methods and protocols already used for BCIs, a couple of novel algorithms to recognize relevant potentials in the brain are presented. In particular, a genetic algorithm — an optimization method that mimics the way natural evolution works — is developed to extract features from EEG data. A mathematical interpretation of the features found by the genetic algorithm is derived, which can be used to identify the most important time intervals in EEG recordings.

The algorithms are tested on real EEG data, coming in part from experiments done specifically for the purpose and in part from other sources. The experimental settings are described in details, with some emphasis on aspects developed in this work. The results are compared with a few methods from other researchers which have been replicated.

For error potentials, experiments are made first in settings different from a BCI, in which subjects perform tasks where the knowledge about error potentials seems more firm in the literature; experiments are then performed in a BCI task. For P300 detection, only data recorded while subjects are operating a BCI are used. Data from persons affected by amyotrophic lateral sclerosis are also tested, and for some subjects classification accuracy is well above 80% in single-sweep mode.

The main testbed for the new algorithms consists in two classical applications: a P300 speller and a P300-driven robotic wheelchair. A P300 speller is a program to write text by selecting one letter at a time, where each selection is made with the recognition of a P300 potential. The robotic wheelchair has been developed for another project and is autonomous: It can receive high-level commands from the user, like "go to the room X". An interface driven by a P300-based BCI is set up to give such commands to the wheelchair. In both applications, error potentials are used to identify mistakes made by the P300 BCI in recognizing the choice of the user; the detection of mistakes made by a BCI should permit to lower the global error rate and, hopefully, the frustration of the user.

Results on offline experiments show that the genetic algorithm performs well and can be effective for a real BCI, and that error potentials are present and detectable in a P300 BCI. Online experiments are done both on the P300 speller and the wheelchair, and they confirm the offline results: The proposed algorithms are robust and fast enough to be used in real-word applications.

An analysis of the performance gain achievable by using error potentials is also presented, based on the novel concept of *utility*. After a critique of the methods used in the literature, where in particular some limits of the "information transfer rate" are showed, utility is presented as a formalization of the benefit that the user receives from the behavior of the BCI. A formula for the performance of a speller is derived, and a discussion of when error-potential detection can be helpful is given. Under some simplifying assumptions, a precise characterization of the performance gain (or loss) is computed as a function of the speller accuracy. Utility can be also extended to different kinds of BCIs and used to choose many project parameters. As a proof of the versatility of utility, a more complex interface is analyzed. Utility predictions are validated through Monte Carlo simulations.

Riassunto

Questa tesi riporta un lavoro svolto nel campo delle interfacce cervello-computer (BCI, dall'inglese *Brain-Computer Interface*) che fa uso di tecniche di apprendimento automatico e statistica. Una BCI è un dispositivo che aggira qualsiasi mediazione di muscoli e nervi, e si interfaccia direttamente con il cervello misurando segnali generati dalla sua attività. Potenzialmente essa potrebbe essere utile per persone che non riescono ad usare interfacce "standard", come per esempio persone affette da malattie che compromettono il movimento degli arti, o persone impegnate in attività fisiche. L'apprendimento automatico è un ampio campo dell'intelligenza artificiale che riguarda tecniche per dotare macchine (computer) della capacità di adattare il proprio comportamento a condizioni diverse e anche variabili.

Il lavoro è incentrato sulle BCI basate su potenziali riconoscibili in registrazioni EEG (elettroencefalografia), come i potenziali P300 e d'errore. Una P300 si ha quando il soggetto riconosce uno stimolo bersaglio raro (*oddball*) in una sequenza di stimoli standard. Un potenziale d'errore è visibile quando un soggetto commette un errore, e, cosa più rilevante per applicazioni BCI, quando la macchina con cui il soggetto sta interagendo non si comporta come egli si aspetta. In una BCI basata su P300 vengono mostrate all'utente delle possibili scelte che vengono evidenziate una alla volta. L'utente fissa la sua attenzione su una delle scelte, e perciò ogni evidenziamento di quella scelta genera un potenziale P300. Potenziali d'errore possono essere sfruttati per rilevare (ed eventualmente correggere) automaticamente errori di una BCI.

Dopo una panoramica di alcuni dei metodi e protocolli usati nelle BCI, vengono presentati un paio di algoritmi innovativi per riconoscere potenziali rilevanti nel cervello. In particolare, viene sviluppato un algoritmo genetico — un metodo di ottimizzazione che imita l'evoluzione naturale — per estrarre caratteristiche da dati EEG. Viene derivata un'interpretazione matematica delle caratteristiche individuate dall'algoritmo genetico che permette di identificare gli intervalli temporali più importanti nelle registrazioni EEG.

Gli algoritmi vengono provati su dati EEG reali, provenienti parte da esperimenti fatti allo scopo e parte da altre fonti. Le condizioni sperimentali sono descritte in dettaglio, soprattutto gli aspetti sviluppati

in questo lavoro. I risultati vengono confrontati con alcuni metodi di altri ricercatori che abbiamo replicato. Per i potenziali d'errore vengono fatti prima esperimenti in cui i soggetti eseguono un compito dove la letteratura sui potenziali d'errore sembra meglio stabilita; quindi vengono eseguiti esperimenti con BCI. Per il riconoscimento di P300 vengono usati solo dati da test in cui i soggetti usano una BCI. Vengono provati anche dati da persone affette da sclerosi laterale amiotrofica, e per alcuni di questi soggetti l'accuratezza è molto sopra l'80% in modalità a una passata.

Il principale banco di prova per i nuovi algoritmi consiste in due applicazioni classiche: uno *speller* a P300 e una carrozzina robotica guidata tramite P300. Uno *speller* a P300 è un programma per scrivere testo selezionando una lettera alla volta e dove ogni selezione è fatta attraverso il riconoscimento di un potenziale P300. La carrozzina robotica è stata sviluppata per un altro progetto ed è autonoma: può ricevere comandi di alto livello dall'utente, come "vai nella stanza X". Viene sviluppata un'interfaccia operata con una BCI basata su P300 per inviare questi comandi alla carrozzina. In entrambe le applicazioni i potenziali d'errore vengono usati per identificare gli errori commessi dalla BCI a P300 nel riconoscere la scelta dell'utente; la rilevazione degli errori commessi da una BCI dovrebbe permettere di abbassare il tasso globale di errori e, si spera, la frustrazione dell'utente.

Esperimenti *offline* indicano che l'algoritmo genetico si comporta bene e può essere efficace per una vera BCI, e che i potenziali d'errore sono presenti e rilevabili in BCI basata su P300. Esperimenti *online* sono effettuati sia sullo *speller* a P300 e la carrozzina, e confermano i risultati *offline*: gli algoritmi proposti sono sufficientemente robusti e veloci da essere usati in applicazioni reali.

Viene presentata anche un'analisi del guadagno ottenibile con l'uso di potenziali d'errore, basata sul nuovo concetto di *utilità*. Dopo una critica dei metodi usati in letteratura in cui si mostrano in particolare alcuni limiti della "velocità di trasferimento dell'informazione", viene presentata l'utilità come formalizzazione del beneficio che l'utente trae dal comportamento di una BCI. Viene derivata una formula per le prestazioni di uno *speller* a cui segue una valutazione di quando la rilevazione dei potenziali d'errore può essere utile. Sotto alcune assunzioni semplificanti, viene calcolata una caratterizzazione precisa del guadagno (o perdita) di prestazione in funzione dell'accuratezza dello *speller*. L'utilità può anche essere estesa ad altri tipi di BCI e usata per scegliere molti parametri di progetto. Come prova della sua versatilità viene analizzata un'interfaccia più complessa. Le previsioni dell'utilità sono verificate con simulazioni Monte Carlo.

Contents

List of Figures

List of Tables

1 Introduction

The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with weary feet,
Until it joins some larger way,
Where many paths and errands meet.
And whither then? I cannot say.
J. R. R. Tolkien – *The Lord Of The Rings*

Men have been using tools and machines of all kinds for quite a long time. Mechanical machines need a human operator that controls them, and for centuries, operators used interfaces like levers, buttons, wheels, pedals... With the progress of electronics, new devices became available, like the pointing devices that are commonplace in today personal computer systems. More recently, speech-based interfaces have been developed, which are certainly more friendly, although they are reliable only in some specific and well-defined environments. Whatever the advancements, all the interfaces used in today devices require the use of some body muscles by an operator. This works fine in general, but there are two situations where the involvement of muscle activity can be a problem: users affected by disabilities, who can lose many opportunities if they cannot use the majority of devices; and users engaged in other activities, like driving.

A brain-computer interface (BCI) is an interface that does not entail muscle movements, but it bypasses any muscle or nerve mediation and connects directly with the brain by picking up signals generated by its activity. The word "computer" in the name BCI means that some numerical processing is needed to interpret the brain activity read by the interfaces, but a BCI can be coupled to any machine through the appropriate effectors, of course.

Electrical phenomena in the brain are known since the 19th century, but it was in the beginning of the 20th century that scientists began to study the activity of the brain with the help of electroencephalography (EEG), the recording of brain potentials through electrodes applied on the scalp. Neurologists used EEG mainly to evaluate neurological disorders in clinics and to investigate brain functions in laboratories, but the relatively recent availability of powerful digital hardware and analysis algorithms opened the door to the use of EEG as a mean of communication, i.e., to BCIs.

There are different kinds of brain activity that can be used in a BCI. Different internal and external events cause different patterns in the brain waves, and many of these patterns have been studied for use in a BCI, as for example, *visual evoked potentials* (VEP), *slow cortical potentials* (SCP), mu and beta rhythms [1]. In this study, we focus on two particular potentials: P300 and error potential (ErrP). The P300 is an event-related potential (ERP) that is visible in an EEG recording as a positive peak at approximately 300 ms from the eliciting event. It follows unexpected, rare, or particularly informative stimuli, and it is stronger in the parietal area. An error potential can be seen when a subject makes a mistake, and, more relevant to BCI applications, when the machine the subject is interacting with does not behave as the user expects.

A BCI is a complex system, which requires an interdisciplinary approach. Yet, different people with different skills and background concentrate on different aspects of the system. Given the engineering background of the laboratory where this research has been developed, Airlab (Artificial Intelligence & Robotics Laboratory) [2] at the Department of Electronics and Information of *Politecnico di Milano* (Italy), the focus of this thesis is on the algorithms for processing the signals from the brain and the architecture of a BCI system viable in the real world.

In the next chapter, after a brief overview of brain physiology, the field of BCI is introduced. Chapter 3 presents the experimental setting used to record EEG data used for the work of this thesis, and Chapter 4 contains a description of the methods used in the following chapter.

In Chapter 5, various approaches to the classification of P300 potentials are presented: from a method derived by the literature to a blind algorithm for feature extraction based on a genetic algorithm. The experimental results are presented, and also an explanation of the inner working of the blind approach is given.

Chapter 6 presents the work on ErrPs: the experiments, the classification method, and the results obtained. Chapter 7 is devoted to assessing the impact of ErrP detection on the performance of a BCI. In particular, after a critique of information transfer rate, a different task-oriented approach is introduced and a precise characterization of when ErrP detection is helpful is given.

Chapter 8 closes the work with some final remarks.

2 The Human Brain And Brain-Computer Interfaces

It has not been hard for me to read your mind and memory. Do not worry!
J. R. R. Tolkien – *The Lord Of The Rings*

Some notions of brain physiology are needed in order to understand how BCIs work. Detailed information on the matter is better found in medical texts (see for example [3]); here, only the most basic information is given, which should be enough to understand the rest of the thesis. After a brief overview of brain physiology, a sketch of the status of the BCI field is presented.

2.1 Brain Physiology

The human brain is probably the most complex object in the known universe. It is responsible for many important functions: body movement, memory, thought, emotions, sensing, and more. The study of the brain is very fascinating and has a long history, but most of what we know today has been discovered in recent years, as technology advancements permitted to study the brain functioning *in vivo*.

2.1.1 Brain And Neurons

Figure 2.1 shows the structure of the human brain. The biggest part of the brain is the *forebrain*. It is responsible for the highest intellectual functions, as thinking, planning, memory (in the *hippocampus*), etc. The cerebral cortex, the external layer of the forebrain, is involved in the most complex functions. It has many fissures (called *sulci* when small) so that it appears to be folded. A longitudinal fissure divides the cortex into two hemispheres; smaller fissures delimit four lobes in each hemisphere: frontal, parietal, temporal and occipital. The two hemisphere are connected at the base by the *pons*. The *cerebellum*, the lower back part of the brain, is responsible for coordinating motor activity and sensory perception.

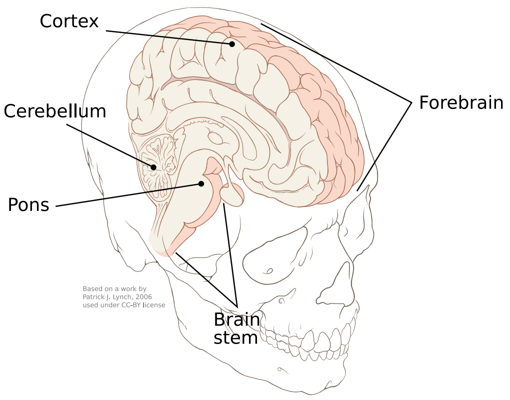

Cortex
Forebrain
Cerebellum
Pons
Based on a work by Patrick J. Lynch, 2006 used under CC-BY license
Brain stem

Figure 2.1: The human brain. Only a few structures are named here.

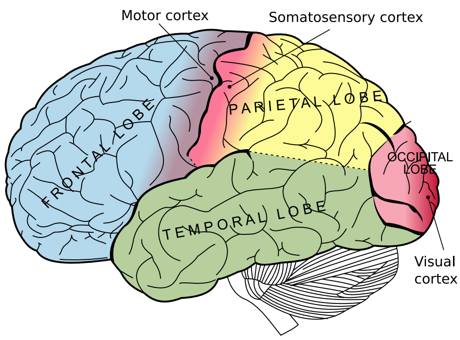


Figure 2.2: The human cerebral cortex. Its lobes and some functional regions are highlighted.

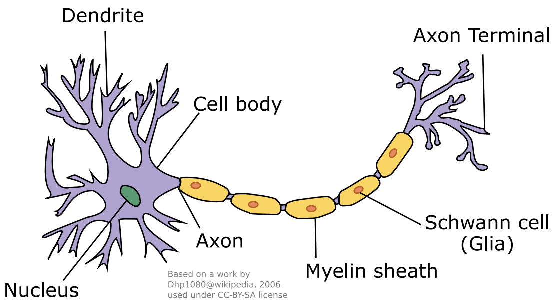


Figure 2.3: A neuron and its parts

The cerebral cortex is the part of the human brain that differs more from the brains of the other mammals, and it is the place where the functions that set humans apart from the other animals are performed. On the cortex (see Figure 2.2) it is possible to identify several regions, each characterized by a specific function. For example, the motor cortex (in the back of the frontal lobe) is responsible for voluntary movements, the somatosensory cortex (in the front of the parietal lobe) receives sensory information from the body, and the visual cortex (in the occipital lobe) receives input from the eyes. In general, each hemisphere controls and receives its input from the opposite side of the body.

The basic elements of the neural tissue are the neurons. There are many neurons in the human brain, some 100 billion of them. A typical neuron is depicted in Figure 2.3. Neurons are connected among them by long fibers that protrude from their body, *axons* and *dendrites*. Electric signals travel on these fibers from the extremes of dendrites toward the neuron body, and from the body toward the extremes of axons. The axon of a cell is connected to the dendrites of other cells, in junction points called *synapses*. Actually, a synapse is a gap between two neurons; the signal travels across this gap by means of chemical substances called *neurotransmitters*. When a signal travels along an axon of a neuron and reaches a synapse, neurotransmitters are released in the synaptic gap, and they bind themselves to receptors on the surface of the other neuron connected by the synapse. The binding of neurotransmitters to receptors modifies the electric potential of the neuron membrane. When the membrane potential reaches a threshold, an impulse begins (*action potential*), which travels along the axon to the synapses at its end. The action potential is a double change of the electric potential across the neuron membrane, caused by the exchange of ions through the membrane. The presence of an insulating sheath, *myelin*, around an axon speeds up the conduction of impulses. The nonlinear behavior of neurons (due to the threshold that controls the firing) together with the fact that each neuron is connected with thousands of others is the cause of the complex behavior shown by animals and in particular humans.

There is another type of cells other than neurons in the brain: glial cells. They perform support function for the functioning of neurons; for example, myelin is provided by glial cells. Although they do not have action potentials, their activity can influence the information processing performed by neurons. A clear example is given by the multiple sclerosis, which is a degenerative disease that causes the destruction of the glial cells responsible for the myelin sheath; its impact on the functioning of the neurons is dramatic, and patients may loose muscle control and suffer of cognitive impairments. Glial cells are also sensitive to neurotransmitters and it seems that they may play a role in learning and plasticity [4].

2.1.2 Watching The Brain

There are several methods to observe the brain activity. They can be divided in two classes: methods that sense the neuron activity directly, and methods that measure blood flow, which is highly correlated with local activity. In the first class, electroencephalography (EEG) makes use of electrodes on the scalp to sense the electrical fields caused by neurons. EEG is the oldest technique of recording the brain activity, it is simple to use and relatively cheap; it has very good time resolution and not-so-bad spatial resolution. Magnetoencephalography (MEG) is sensitive to the tiny magnetic fields induced by the electric currents in the brain. MEG equipment is bulky and very expensive, and it is sensitive to electromagnetic interference, yet it has good spatial and temporal resolution; it has been used for BCIs [5], but its use is very limited. The limit to EEG resolution is given by the tissues present between the electrodes and the brain surface; electrocorticography (ECoG) is an EEG recorded from electrodes placed directly on the brain surface. For evident reasons, it is an unlikely candidate for everyday BCIs, although research involving this technique is being carried on with good results [6, 7, 8]. Some studies have been done with implanted electrodes, focused on small regions in the brain, with the hope that future technological advances would make the implantation of electrodes safer and more stable in time [9, 10, 11]; more experiments of this kind have been done on animals [12, 13]. While this technology is still evolving, and there are interesting developments [14, 15], such practice requires complex surgery, and hence it has several drawbacks, which have to be carefully considered before use.

Neuron activity needs energy to be sustained, so blood flow increases in more active regions. Techniques for brain imaging have been developed that exploit this phenomenon. Changes in blood flow happens over seconds, so imaging methods that detect such changes are slow when compared to direct approaches like EEG. Positron emission tomography (PET) [16] is a way to measure neuron metabolism through the injection of a radioactive substance in the subject; it is used in clinical practice, but not for BCIs. Functional magnetic resonance imaging (fMRI) measures the metabolism by comparing the concentration of the oxygenated hemoglobin and the deoxygenated hemoglobin, which respond differently to a magnetic field. It needs an expensive and bulky equipment, but it gives access to the whole brain and not just its surface. The reconstruction of the 3D map of the brain requires heavy processing of the data, and this is not done in real-time, normally. Yet, experiments with a fMRI-based BCI have been done [17]. Near-infrared spectroscopy (NIRS) measures the reflection of infrared light by the brain cortex through the skull. It is a way to measure changes of oxygenated hemoglobin and deoxygenated hemoglobin, which is due to changes in the activity of the brain cortex and, consequently, of blood flow. It is a relatively young technique of brain imaging, and it has been tried in BCIs with good results [18], but time will tell how its potentials will be developed.

The different techniques to record the brain activity have different merits and different shortcomings. Invasive techniques (electrodes inside the skull) have the highest signal-to-noise ratio, but they are invasive, indeed. MEG has very good resolution, both in time and space, but MEG equipment is very bulky and expensive, and it must be used in a room shielded against magnetic interferences. Similarly, fMRI has good spatial resolution, but its equipment is bulky and expensive; moreover, fMRI has relatively long response times. NIRS suffers from the same problem of low time resolution as it measures blood flow, and its space resolution is not good either; yet, from the viewpoint of the user, it is the simplest system among those mentioned. EEG has very good time resolution, and a spatial resolution that demonstrated to be good enough for BCIs; it is relatively cheap (while EEG equipment still does not sell at bargain prices), and not difficult to use, although it requires some time to put the electrodes in place and having conductive paste in one's own hair is not pleasant. Anyway, EEG seems to be the most wide-spread system for interfacing with the brain, and it is also used at Airlab. For this reason, the rest of this chapter makes reference almost only to it, and its principles and practices are here described in some details.

The electric potential on the scalp measured by EEG devices is generated mostly by neurons located in the brain cortex. Each electrode is influenced more by the region of brain just underneath the electrode, and by using more electrodes, it is possible to build a map of the potential of the whole cortex. The signal picked up on the scalp is very weak, in the order of tens of microvolts. Very sensitive and high-quality amplifiers are needed in order to record the signal without adding too much noise. Modern electronic technology enables to build relatively cheap and compact devices that exploit digital electronics (see Figure 2.4), which feed EEG data to a personal computer instead of driving a pen on paper as in the traditional ones. Sometimes, they apply also some form of filtering, as band-pass filtering to remove unwanted

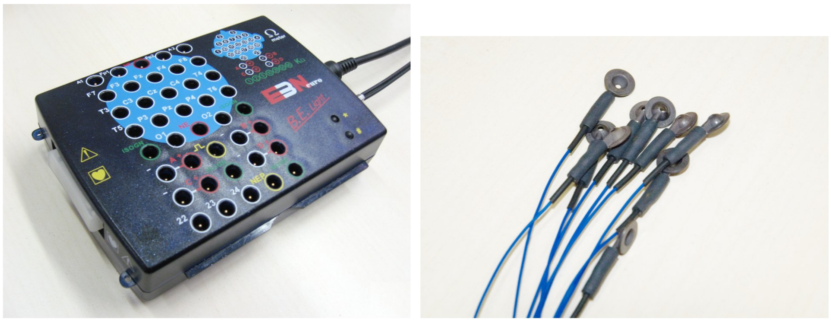


Figure 2.4: A modern digital EEG amplifier (left) and some EEG electrodes (right).

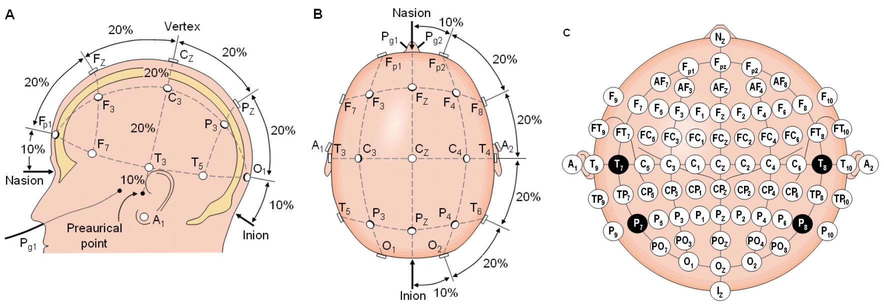


Figure 2.5: A and B: the 10-20 system; C: the 10-10 system. Odd and even numbers are used respectively for the left and right sides, and z for the center; *Fp* = frontal polar, *F* = frontal, *C* = central, *T* = temporal, *P* = Parietal, *O* = occipital. From [19, Web version].

high frequencies and drifts, or notch filtering to remove the pervasive interference from the power line. EEG electrodes are small disks placed on the scalp (see again Figure 2.4); they are made of a metal, normally silver or gold, that does not interact with skin. A light abrasion of the skin (to remove the outer layer of dead cells) and the use of a conductive gel or paste are required to reduce their impedance to acceptable levels (3–5 kΩ are considered good).

Electrodes are usually placed in precise locations according to international conventions. In this way, electrodes are placed always at the same places for a given subject, and it is possible to compare and communicate results between different institutions and clinicians. There are two widely-used standards, the 10-20 and the 10-10, shown in Figure 2.5. Both define the electrode positions in terms of the distances between two fixed point on the skull: *nasion* (the delve at the top of the nose between the eyes), and *inion* (a small bony bulge at the base of the skull, in the back). The names of the systems come from the fact that the electrodes are placed at the intersections between lines whose distances are 10% or 20% of the distances between the reference points. The 10-10 uses only 10%-distances, so it is a superset of the 10-20 system. Each position has a name, and most of the electrodes common to both systems keep the same names.

It is known from physics that an electric potential is defined only with respect to some zero level, i.e., a reference. This is true also for EEG. There a few different ways to define a reference in an EEG acquisition. One electrode can be selected as the reference potential, and the difference between the potentials of the other electrodes and the reference electrode is measured (*common reference*). Typical places for positioning a reference electrode are the mastoid (the bony prominence just behind the earlobe), the earlobe, and the forehead. Sometimes the average of electrodes placed at both mastoids or earlobes is used, because of the symmetry of this setup; this average can be computed numerically on the digitized data, or the electrodes can be physically linked by a wire. After an acquisition, it is also possible to change the reference electrode by subtracting its signal to all the other ones. A variant is to subtract the average of all signals to each of them (*average reference*). Both reference schemes just illustrated use one reference for all signals (*monopolar channels*). An alternative scheme is to use electrodes in pairs, so there is one reference for every recorded channel; this configuration is called *bipolar*. The set of the recorded electrodes together with the reference scheme define a *montage*.

2.2 BCIs: From History To The Present

Electrical phenomena in the brain are known since the 19th century, but it was at the beginning of the 20th century that scientists began to study the activity of the human brain with electroencephalography [20]. Different types of waves, corresponding to different frequency ranges, were discovered and associated to different activities or states of mind of the subject (e.g., sleep, relax, anxiety, physical activity). Neurologists used EEG mainly to evaluate neurological disorders in clinics and to investigate brain functions in laboratories. Some studies explored the possibility of using EEG for therapy, as a support to biofeedback practice, but results are still controversial [21].

More recently, as powerful digital hardware and new analysis algorithms have become available, EEG has been investigated as a mean of communication, i.e., as an input to a BCI. Jacques Vidal presented a BCI system in 1973 [22], which was built at the University of California, Los Angeles; it is the first BCI on record. Emanuel Donchin and colleagues built a BCI in the 1980s, and in the next decade many more people were doing research on BCIs. Results are very promising and exciting, although both the state of the art and the foreseeable short-term developments are still very far from what has been envisioned in some movies (probably "The Matrix" is the most famous).

Different research groups use different names for the systems they build: for example, brain-machine interface, direct brain interface, thought translation interface. The term BCI seems to be the most common and wide-spread, and for this reason it will be always used in this work, even when referring to systems named differently by their inventors.

While the idea of controlling a device directly with thought[1] is interesting and challenging by itself, with many possible applications, an important driving factor for this type of research is the application of BCIs in helping disabled people. There are thousands of people that become paralyzed in the world each year, whether by trauma or by disease. In many cases, paralysis is complete, mostly because of degenerative diseases like amyotrophic lateral sclerosis or muscular dystrophies. Disabled people are the most likely to benefit by the use of BCI, because BCIs, as far as the state of development is now and for the near future, are rather slow, and however sad it may seem, the most needing people are the one that need the less to help them. Communication rates of a few bits per minute are the norm, but such a low bit rate is still very useful for people who have no muscle control whatsoever, both to control the environment (e.g., switch lights on or off) and to communicate via a computer (predictive algorithms can be used to produce 1–2 words/min). But a BCI would not benefit paralyzed people with residual movement capabilities, as exploiting such capabilities with custom keyboards, switches, eye-tracking devices, etc. is more effective. Although useful for some people, BCIs are mostly used in laboratories, with very few cases of use in everyday life. The reason is that there are many aspect of BCIs that must be improved still (more on this later).

[1]The word "thought" is at best an approximation of the actual way of using a BCI, as explained later in more details.

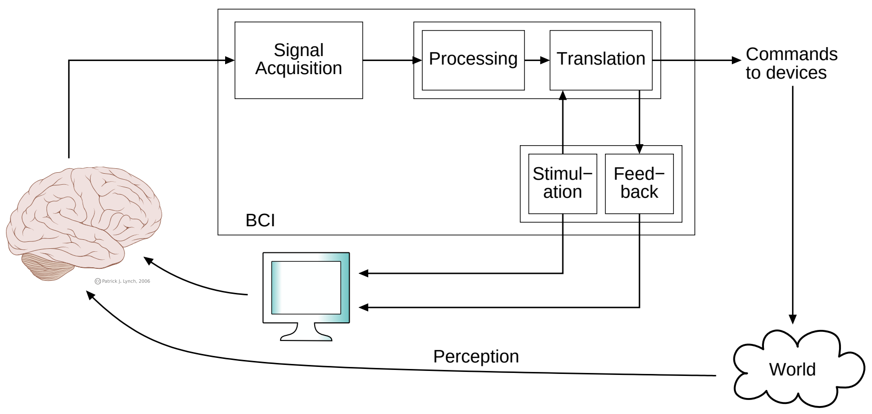


Figure 2.6: Architecture of a BCI

The basic architecture of a BCI is illustrated in Figure 2.6. There are three main components: signal acquisition; signal processing and translation; and feedback and/or stimulation (these latter elements may be missing in some settings). Signal acquisition is done with one of the techniques mentioned before (see Section 2.1.2). The exact acquisition parameters (e.g., electrode placements for EEG) depends on the kind of brain activity that is going to be used. The brain activity of interest could be generated by the conscious will of the user, or it may come from an unconscious mechanism — but still reflecting a conscious choice — in response to stimulations by the BCI. This difference allows us to make a distinction between *endogenous* and *exogenous* BCIs respectively. Also, the type of input discriminates between *dependent* and *independent* BCIs, i.e., between BCIs whose functioning depends on the activity of muscles and those which do not. More important, from the point of view of the user, is the distinction between *synchronous* and *asynchronous* (also *self-paced*) BCIs; here the discriminant is whether the BCI or the user sets the pace. An endogenous, asynchronous BCI lacks a stimulation component, while it exist in exogenous and synchronous BCIs.

Facial muscles and their nerves produce strong electrical signals, much stronger than EEG, and they can be picked up by BCI sensors and appears as artifacts in the recorded tracks. Ocular region is a great source of artifacts, with movements of eyes, rising of eyebrows, and eye blinks. Tongue and throat (swallowing) are another source, and heart beat can also show up in the EEG of some subjects. Signals from muscles are orders of magnitude stronger than EEG signals, and hence disrupt performance [23]. But they could also be used by the BCI, so that a healthy user might in fact control the interface using his own eyebrows, for example. This would invalidate any result or conclusion drawn from such an experiment with regard to BCI effectiveness. BCI users are normally given instructions to remain still, not to blink or to try to blink only at certain times, and so on, but artifacts crop up anyway. There are different techniques that can be used to identify and remove artifacts; some are the same used for classifying features into the supposed user's response [24, 25, 26].

The acquired signal is digitized and further processed. The goal of processing is to enhance the signal, for example, by removing artifacts, or by isolating the relevant components in the raw signal. There is no general way to do this; techniques are chosen based on the particular brain event to be processed and the personal taste of the designer. Yet, some standard techniques are applied often, like band-pass filtering, frequency analysis, averaging, or spacial filtering. The same applies to the features used for signal classification; features may be, for example, intensity of spectral power components, autoregressive parameters, peak values, wavelet coefficients. Salient features isolated by signal processing are then translated into actual commands. This translation could be anything from a plain classifier for binary outputs to a very complex model that controls the trajectory of a robotic arm. The user of a BCI may receive a feedback directly from a BCI, or the feedback may come from the effects of the devices driven by a BCI. In some settings, this feedback is useful also to train the BCI user.

Brain responses and EEG potentials change with users, therefore machine learning and other adaptive techniques are important, as they permit to fit a BCI to a particular user and to the changes of user's response with time. While it is possible for a user to learn and adapt to a BCI, every help by the machine is useful, as user learning may require long training sessions; moreover, control over some EEG features cannot be achieved satisfactorily by all subjects.

2.2.1 Signals From The Brain

Over the years, neurological studies have found many phenomena taking place in the brain. Among all the phenomena, some can be used to detect the intentions or the choices of a subject, and these are the ones that can be useful for a BCI. In this section, the phenomena employed in BCI studies are described, along with the way they are used in a BCI.

Although all research on BCIs is based on some well-known brain aspect, attempts to work disregarding any knowledge of brain have been made. In [27] a direct approach is depicted, where the user is required to concentrate on one of two commands (actually “yes” or “no”), without reference to stimuli or previously-agreed strategies. This BCI uses only general-purpose techniques (independent component analysis, genetic algorithms, and support vector machines). While results are not as good as other kinds of BCIs, it may open interesting routes for the future.

It is possible to control an external device with voluntary EEG modulation by training the user with techniques derived from biofeedback (e.g., slow cortical potential described below), but most work in the field focuses on BCIs where the user has to perform precisely defined tasks, which can be paying attention to particular stimuli (as in the case of visual potentials, P300s, and ErrPs described below), or thinking of moving a part of their body (motor imagery, see below), or performing tasks like mentally manipulating numbers, words, and geometric figures [28].

Visual Potentials

The *visual evoked potential* (VEP) (also *visual evoked response*) is the electrical potential recorded on the visual cortex in response to stimulation of the subject’s visual field. A VEP is the effect of the processing within the visual cortex, and it changes with the changing of the stimulus.

A possible setup of a BCI involving VEPs consists in different symbols displayed on a screen, where each symbol is highlighted by modulating their luminance at different moments or in different ways; the user should look at the symbol he wants to select. VEPs can be viewed by averaging many recordings synchronized on the stimulus onset.

The Vidal’s BCI [29] (the first BCI ever built cited above) was based on VEPs. Users were asked to navigate through a maze controlled by a computer. A stimulus consisting of a checkered diamond with four points placed at the corners was shown for a brief time. By fixating one of the four points, the user indicated the direction of the next movement. The system discriminated between four possible VEPs and updated the position in the maze.

SSVEP (Steady State VEP) is a particular VEP, which occurs in response to light flickering at some fixed frequency. The response in the brain has the same frequency of the stimulus. In a BCI [30], a panel contains two or more symbols, each representing a possible choice for the user, which flicker at different frequencies. When the user look at a particular symbol, an SSVEP is elicited. An analysis of the frequency components in EEG permits to identify the symbol. No particular training is required, as evoked responses are innate, and accuracy is high [31].

These BCIs are examples of exogenous and dependent BCIs, and they basically detect gaze direction. They can be viewed as another mean for detecting messages carried in the brain normal output pathways (in this example, the gaze direction), so they may have limited value for assistive technologies.

Slow Cortical Potential

Slow cortical potential (SCP) is a shift in the potential of the whole cerebral cortex, in the frequency range below 1–2 Hz. Subjects can learn to self-control their SCPs when they are provided with visual or auditory feedback of their brain potential. A positive or negative change in the potential registered over a few seconds period can be used to select between two choices or commands (more choices can be allowed by using a binary selection tree). This is what Birbaumer and colleagues at University of Tübingen (Germany) have used in their *Thought Translation Device* [32]. Even a few months of training is needed to achieve an accuracy sufficient for communication (yet far from 100%).

Motor Imagery

Mu (8–12 Hz) and beta (12–26 Hz) rhythms are correlated with motor cortex idleness. Body movements or even imagined movements (*motor imagery*) are typically associated with a decrease in mu rhythm and an increase in beta rhythm, particularly in the hemisphere contralateral to the movement. So motor imagery (or its absence) can be used to create a binary signal [33]. In a slightly different approach, two (or possibly more) different actions are imagined by the user, and the BCI system tries to discriminate between the actions by comparing features of the signals registered over the motor cortex [34]. In other works, additional discriminating tasks are activities not related with imagined movements, like performing mental computation or visualizing geometric figures [35]. More interestingly, this latter work uses a threshold of confidence for classification, and whenever the threshold is not reached, the system does not take any action.

Motor imagery is probably the most popular type of BCI. Possible reasons for this fact are that is a protocol that require a not very long training phase, and discrimination can be made with simple algorithms like computing a Fourier transform. Yet, training on the part of the user is always needed, as it is not easy to concentrate on an imagined movement without actually doing it. After a while, users report that they no longer think of moving body part, but they produce the right patterns automatically; this ease of use is beneficial for perspective users, but it requires time. Simple algorithms are seldom used; more often, they are replaced by more sophisticated ones in reality.

Several groups around the world make use of motor imagery. Here, only a quick overview of the work of some of them is given. The Maia project (Mental Augmentation through Determination of Intended Action) [36] was a European project (partners were the Swiss IDIAP, the Italian Fondazione Santa Lucia, the Belgian Katholieke Universiteit Leuven, the Finnish Helsinki University of Technology) that terminated a few months ago, and which aimed at developing an EEG-based BCI for controlling autonomous robots. It was based on the principle that the human user imparts just high-level commands through a BCI, while an autonomous robot deals with all the low-level details [37, 38]. Behaviors like obstacle avoidance and wall following were built in the autonomous control of a wheelchair, while the user just gave simple commands like "go straight", "go left". Motor imagery was used; cortical potentials were estimated from scalp EEG through a mathematical model [39], and band energies were used to discriminate different imagined movements. Interestingly, very high frequencies ($> 100\,\text{Hz}$) were also used in the classification process [40]. Haptic feedback was given to the user [41], and detection of error potentials (see Section 2.2.1) in the user helped to lower the overall error rate of the system. They showed that it is possible to drive a wheelchair with a BCI, also in live demos, though the noisy environment lowered the performance dramatically.

The group lead by Gert Pfurtscheller at Technische Universität Graz have been working with motor imagery for several years. They developed a system that records EEG in three points and discriminates between three imagined movements: left hand, right hand, and feet. Spectral features are used in classification. The first Graz BCIs were synchronous [42], but in more recent works, as the virtual reality project [43] or the restoration of movements in quadriplegic patients through functional electrical stimulation [44], asynchronous protocols are used.

The BCI research group at the Wadsworth Center (in Albany, NY, USA) uses mu and beta rhythm over sensorimotor cortex to move a cursor in one or two directions. New users are advised to use motor imagery for producing changes in relevant wave rhythms, and they learn to control the cursor in training sessions [33, 45].

The Berlin BCI group, a collaboration between the Fraunhofer FIRST institute and the Charité Universitätsmedizin in Berlin, have developed various BCIs based on motor imagery with the goal of reducing the time the user spends in learning; this is achieved by employing many machine learning techniques, but also by using a large number of electrodes (typically 64 or more) [46, 47].

Motor imagery has been used in experiments with implanted electrodes as well, such as those on human by Donoghue [11] and Kennedy [10], or on animals by Nicolelis and Chapin [48, 12, 15] and Schwartz [15, 49]. As expected, with implanted electrodes it is possible to obtain better results than with EEG-based motor-imagery BCIs. Time will tell if the drawbacks of an invasive approach can be reduced by new technologies and balanced by the improved performance.

Event-Related Potentials

Event-related potentials (ERPs) are components found in EEG recordings generated in response to specific stimuli or events. In general, ERPs do not stand out from the background EEG, but they can be detected by averaging many recordings time-locked to stimulus presentation. This averaging cancels out the background activity, which is not synchronized with the stimulus, and leaves only the ERP. The problem with averaging is that ERPs may change with time and between sessions, and for this reason the so-called *grand average* may not capture the whole complexity of the phenomenon. Examples of ERPs are *P300* and *error potential*; they are described below in more details, as they play a central role in the present work. Other examples are *contingent negative variation* [50], a negative shift in the potential over the fronto-central region visible when subjects are expecting a stimulus after having been given a warning signal, or readiness potential (also called *Bereitschaftspotential* by the German-speaking scientists who discovered it), which is a negative shift in the potential of the motor cortex just before a voluntary movement. Readiness potential has been used in BCI studies [51, 52], and it has an interesting application: shortening reaction times of human operators controlling time-critical devices. Of course, confidence in detection of readiness potentials should be very high for such an application, so as to avoid potentially catastrophic false positives for the sake of gaining just a few hundredths of a second.

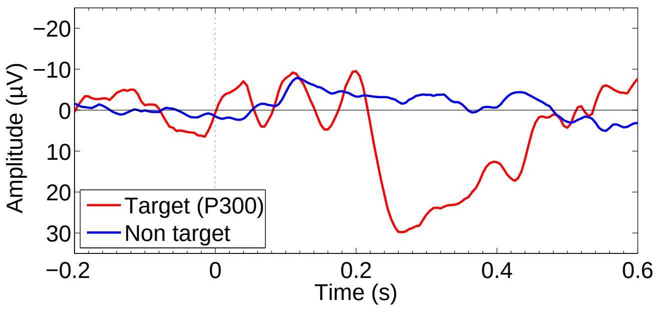


Figure 2.7: Plot of a P300. Averages over many trials in an oddball paradigm are shown. Onsets of stimuli are at time 0.

P300

A *P300 potential* is an ERP that occurs when a subject detects an occasional target (oddball) stimulus in a regular train of standard stimuli. Typically, the recording of a P300 presents a positive peak at about 300 ms (hence the name) after the oddball stimulus (see Figure 2.7[2]). Stimuli can be visual, auditory, or even tactile. The exact shape of the P300 depends on the characteristics of the stimuli and their presentation and varies with the subjects. As the P300 has been known for many years [53], it has been extensively studied. A negative component, a peak at about 200 ms after the stimulus, has been observed to precede the positive peak. The intensity of the P300 is positively correlated with the salience of the stimulus, negatively with probability (less probable stimuli elicit stronger P300s). Also, when stimuli are difficult to discriminate, P300 is smaller. Subjects must be conscious of and attentive to stimuli for a P300 to be elicited, and the more the attention, the bigger the P300. Often subjects are suggested to count the target stimuli, so as to keep their attention high.

As far as BCI applications are concerned, the exact shape of the P300 is not so important, as long as there is a way of detecting it. And given that the P300 varies much between different subjects and even between the same subject in different times, adaptive algorithms are needed. This is true for all ERPs, not only for the P300.

The P300 has been widely used in BCIs, with many variations, but in all cases the paradigm is basically the same: The BCI system presents the user with some choices, one at a time; when it detects a P300 potential the associated choice is selected. The user is normally asked

[2]Many EEG devices show tracks with the positive axis pointing down. This convention is followed in the present work.

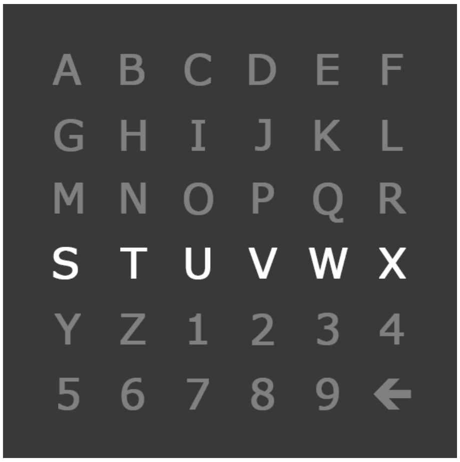


Figure 2.8: The visual stimulator for a P300-based BCI speller. Lines and rows are flashed one at a time.

to count the number of times the choice of interest is highlighted or presented, so as to remain concentrated on the task. As the P300 is an innate response, it does not require training on part of the user, and although individual response changes with time, as said before, it seems that it is stable enough for an appropriate classifier to be able to work even after a long time [54].

In [55, 56], Donchin and colleagues described a system to spell words based on the P300. They built the first P300-based BCI in 1988: A grid of letters and symbolsis presented to the user, and entire columns or rows are flashed one after the other in random order (see Figure 2.8 for an example). Flashing is repeated several times, and all the EEG recordings related to the same letters are averaged together; the resulting signals (one for each letters) are compared to find the one more likely to contain a P300 and hence the letter selected by the user. Classification is made through stepwise discriminant analysis (SWDA) applied to the averages of samples (but other methods are also used). The number of times a letter has to be flashed before calculating a P300 score is predetermined for each user so as to get a good trade-off between speed and accuracy.

At the National University of Singapore a wheelchair controlled through a P300 BCI is under development [57]. The wheelchair is autonomous, but its movements are constrained to predefined paths; the user selects a destination and the chair drives there. If an unexpected situation occurs, the wheelchair stops and waits until the user decides what to do. Classification of P300 events is done by a support vector machine (SVM) fed with samples of the recorded EEG and its time derivative. Every stimulation is processed and classified as target or non-target independently from each other; stimulation is repeated several times for every possible choice, and classifications of stimuli referring to the same choice are aggregated until a threshold is reached.

In [58], one of the winning groups of the BCI Competition 2003 [59, 60] for the data set *IIb* describe the method they used. It is cited here because that data set has been used also in the present work. Data was provided by the Wadsworth Center and consist of EEG recordings made with a P300 speller like the Donchin's one described above. The winners achieved 100% of correct classifications with only 5 repetitions, i.e., after each column and each row has flashed for 5 times. The authors trained an SVM on the raw samples of the digital EEG, almost without preprocessing. SVMs are described in more details in Section 4.2.2; they can be thought as a way to find a separation surface in the space of EEG samples (with a non-linear distance metric). In a sense, SVMs classify examples basing on similarity, although the similarity metric is computed in a transformed space. So, it is somewhat surprising that such a method works without extracting specific features of the signal and without any *a-priori* knowledge of the problem. From the analysis of the method, another fact seems interesting. Of the 64 EEG channels in the data set, the winning group used electrodes placed on the central line, where the P300 is typically more pronounced, but also electrodes over the motor cortex and the visual cortex. This may suggest that other phenomena, besides the P300, are used to discriminate the target letter.

In [61] tests have been made both with healthy and impaired subjects. The subjects control a cursor by choosing among four commands (up, down, left, right) via the P300. Single-sweep detection is performed; independent component analysis (ICA) is used to decompose the EEG signal, a fuzzy classifier identifies a candidate P300 component among the ones extracted by ICA, and a neural network classifies it as target or nontarget. The system is more effective with healthy subjects, though no exact reason could be pinpointed.

Although a P300 is delayed, and hence it can be detected only after (at least) a few tenths of seconds, stimulations can follow one after the other rapidly. This means that a P300 responses may be superimposed with the next stimulation, but this does not seem to be a problem.

By looking in the literature at what kind of BCIs the various groups around the globe are working on, it seems that motor imagery is the most widely used. Probably, the reason for the popularity of motor imagery is that in a P300-based BCI, the interface gives the timing, while it seems more natural if the user sets the pace. Yet, it is not entirely true that a P300 BCI imposes timing constraint to the user; if provided with a confidence threshold, or other ways to assess user attention, a P300 interface may just sit idle until the user turns his attention to it. Also the time needed to present the user with enough stimulations to activate the interface is similar to the time needed to operate a motor imagery BCI, if we take into consideration that a binary or ternary motor-imagery BCI conveys much less information per selection than a P300 BCI. For example, in [62] a 3-class motor imagery is used and the classifier chooses among the three classes 128 times per second, but such classification is used to move a cursor on the screen and drive a virtual keyboard; the speed of the keyboard is about 2 correct letters per minute, which is comparable with the performance of a P300 speller. In [35], a 3-class BCI performs a selection every 0.5 s, but the BCI is used to drive a mobile robot, a task that does not involve many switches between commands — as the authors observe; from the graphs and the data in the cited work it is possible to estimate that the tasks involves about 20–25 switches per minute, each conveying about 1 bit of information: Again, comparable to a P300 speller. This quick comparison does not imply any judgment regarding which one is the best BCI; the choice between different kinds of BCIs must take into account many more aspects, as the requirements of the task, the preferences of the user, and the actual speed achieved by the individual user.

Error Potential

An *error potential* (ErrP) is an event-related potential that is present when a subject makes a mistake, and, more relevant to BCI applications, when the machine the subject is interacting with does not behave as the user expects. Studies in the late 1980s made independently in Germany and Illinois (USA) [63, 64] found a response in EEG recordings shortly after the subject making a mistake. Subjects were asked to press different keys in response to different stimuli, within a time constraint. Both groups found a negative shift in the electric potential over the fronto-central region (from Fz to Cz of the 10-20 system) occurring 50–100 ms after an erroneous response. For this reason, it was called *error negativity* (Ne) or *error-related negativity* (ERN). A subsequent positive shift located in the parietal region was found only by the German group [63]. This positive shift has been called *error positivity* (Pe), and its maximum occurs between 200 and 500 milliseconds after the error. The delay figures above are only indicative, as delays are influenced by the characteristics of the particular task; in fact, different authors have

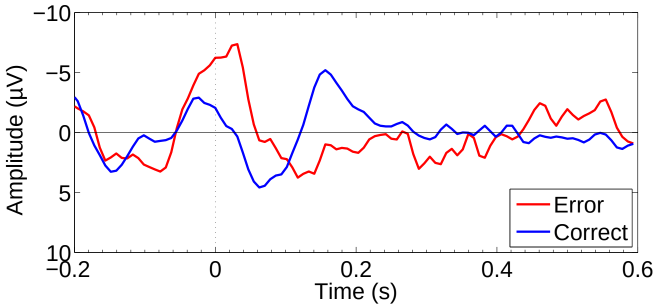


Figure 2.9: Plot of an ErrP. Averages over many trials are shown. Signal are synchronized on the key-pressing event, at time 0 in the plot.

found different results. Figure 2.9 shows an example of an ErrP recorded in an experiment at Airlab.

Many experiments have been done, with different types of tasks and different parameters (e.g., timing). One variant is the use of a go/no-go task, where the subject is asked to either press a key or to do nothing depending on the stimulus type. A high variability in shape, size, and delay of the Ne and Pe components has been observed. The two components are thought to be the effect of different underlying mechanisms, whose nature is not yet certain. Ne has been shown to be smaller when time pressure increases, while Pe remains constant. When there are more than one possible error (e.g., the subject has to push one among four buttons), the severity of the error is correlated with Ne amplitude, while, again, the Pe is not affected. Moreover, the Pe is smaller with higher error rates, while the Ne amplitude is independent of error rate. The most likely hypothesis explaining the Ne is that the Ne is elicited when a mismatch is detected between the neural representations of the correct and the wrong responses [65]. A study of Nieuwenhuis et al. [66], where subjects are asked to control saccades (eye movements), shows that the Pe is higher for conscious errors, i.e., when eye movements are big enough for the subject to be aware of them. It seems that the Pe is linked with a conscious perception of errors, but it is also possible that it is the effect of a learning process or an emotional reaction.

The Ne has been found also in experiments where subjects do not know if they have made a mistake until they receive a feedback about their performance, as in a task where subjects are asked to estimate a time interval [67]. Apparently, this phenomenon is still an ErrP, and it is possibly connected to a learning process (a hypothesis made also for the “standard” ErrP). But a very similar ERP has been seen even in gambling tasks [68].

Researchers at the Wadsworth Center [69] investigated the presence of ErrPs in a BCI application (cursor movement by mu and beta rhythms) and found a positive peak at Cz 40 ms after the end of erroneous trials. Although the features of this ERP are rather different from the ErrP mentioned above, it has possibly an interesting application: automatic detection of errors made by the BCI in recognizing the user’s intent.

The Berlin BCI group performed experiments with ErrPs; apparently they reported only works where subjects had to press the right key in response to a visual stimulus [70, 52], although they reported “online use of the error potential” as a line of research in a later article [47].

Millán and colleagues worked on the possibility to automatically detect ErrPs due to BCI mistakes to improve their performance [71]. They first made experiments with a simulated BCI, i.e., subjects interacted with the system by pressing keys instead of using brainwaves [72]. Again, they found an ERP with different features from the “classical” ErrP, where users did mistakes. They found a negative peak around 270 ms after feedback, a positive one between 350 and 450 ms, and another negative peak around 550 ms. More interestingly, they trained a Gaussian classifier to recognize ErrP (reaching an accuracy of about 80%). In further work [73, 74], Millán and colleagues found ErrPs in simulated BCI experiments. Six users where asked to drive a discrete BCI using 2-class motor imagery, and the system was programmed so as to give the correct feedback with 80% probability. The classifier for ErrPs worked with an accuracy of 70–80%; the authors estimated that using ErrPs should double the bit rate of that BCI for the subjects of the study. They trained a classifier for the motor imagery in parallel with the classifier for ErrPs; they estimated the bit rate of the BCI on new session data with and without ErrPs detection and found about a doubling of the performance. Online detection in a real BCI has been recently reported [75].

In a work for detecting errors by Bayliss et al. [76], the authors found that a P300 is elicited when the BCI selects the right element. While this is not an ErrP, it permits a similar approach to error correction.

2.3 Some Considerations

After what has been said, it should be clear that the building of a BCI is not a simple task. The performance of an independent BCI can reach some 20 bits/min of information transfer rate [1]; for communication, this figure could be improved by the use of word and letter prediction, so as to produce 1–2 words per minute. Results presented in most of the literature have been obtained in laboratory environments, and the time when a BCI could be used at home without problems has not come yet. Trials have been done at home [77], but oversight of a technician is still needed, both for electrodes (which cannot be worn for long without maintenance) and the system.

Difficulties are many, and there is work to do in many areas. Selection of EEG electrodes, for example, is not always a straightforward choice; while physiological considerations may suggest some locations, the optimal configuration changes from subject to subject. Dry electrodes, i.e., which do not need any conductive paste, are desirable, and different research groups are working on it (see for example [78]), but they are still at a prototypical stage. Signal processing is an open problem; many methods have been used, coming from different research areas: signal analysis, data mining, statistics, machine learning. No technique has proven to be "the best". What works for a subject may not work for another, or in different settings, but the field is still young, and only time can say what works best.

Although BCI systems have been proven to be useful, there are many limitations in the approaches found in the literature. In several systems, the selection of the interaction protocol and the extraction of features from brain signals are made as a preliminary step in the development of a BCI. These are probably the most important points for the effectiveness of a BCI, so a better approach would be to adapt them to each single user. This is something that the work in Chapter 7 wants to help to achieve.

All BCI systems in the literature have a training phase, so they adapt to new users, but online adaptation, i.e., adaptation to changes of signal during use, is far from common. EEG signals (like other physiological signals) typically display variations linked with the more disparate factors, like, e.g., time of day, hormonal levels, emotions, fatigue, illness. A BCI should adapt to these changes to maintain its effectiveness [79, 80]. This means that a software module should supervise the use of the BCI and adjust its behavior, but, also, some method to give a feedback to such supervisor module is needed.

In general, this kind of adaptation is difficult to implement, as many components must interact effectively (the BCI, the supervisor, the feedback provider), and the system must be programmed beforehand to cope with situations not always well defined.

Adaptation of a BCI to a user is always done with the goal to obtain a higher performance, i.e., a higher correlation between the system response and the user's intention. This optimization is applied to several components and parameters of a BCI, but they are often optimized separately. A global optimization, i.e., the optimization of all aspects in parallel, should deliver better results, but for sure it is computationally more expensive, and this is why it is rarely done in practice. We can expect that if computers continue to get more powerful as they have done in the past, global optimization approaches will be more common in the future.

Many protocols require users to adapt themselves to the BCI and learn how to use it [81, 23]. Although this learning phase may last for many months, it is a prerequisite for BCI use and it is considered more or less concluded at that point. But humans — and the human brain — continually adapt themselves to new conditions and change with time, even after the initial training phase and during everyday use of a BCI. We all have experienced such phenomenon in our everyday activities. It could take the form of learning, i.e., an improving of performance, or habituation, i.e., a decrease of the response. A decrease of the brain response is very likely to negatively affect BCI performance, but even learning might have a negative impact if the BCI cannot adapt to changing conditions. It is important to note that these issues are about long-term effects, so a more extensive use of BCIs is required to talk about them beyond mere speculations. Only when BCIs will exit the laboratory to enter the ordinary clinical practice it will be possible to see and understand what happens, and also try to develop BCIs that incentive or take advantage of benign adaptation.

A final note. As already said, a BCI should be most useful to severely ill people, but often tests are made on healthy people because it is easier. On the one hand, that is regrettable, as less data are available on how well a BCI works with disabled people than with healthy people. On the other hand, making initial trials on patients may frustrate them or create too high expectations: Some balance is needed.

3 Experimental Setting

And they made letters and scrolls and books, and wrote in them many things of wisdom and wonder in the high tide of their realm, of which all is now forgot.

J. R. R. Tolkien – *The Silmarillion*

'A safer seat than many, I guess,' said Legolas.

J. R. R. Tolkien – *The Lord Of The Rings*

The work of this thesis is focused mainly on the detection of two different ERPs (event-related potentials) in EEG data: P300 and ErrP (error potential). We performed a number of experiments to acquire data to test the algorithms developed for P300 and ErrP detection described in the next chapters. Moreover, the work of the present thesis is a part in a bigger work whose long-term goal is to provide BCI-based assistive devices for paralyzed people.

This chapter describes the equipment and the software used for the experiments done at *Airlab* (the Artificial Intelligence & Robotics laboratory at *Politecnico di Milano*, where this thesis has been developed).

3.1 EEG Acquisition

The core component of the EEG acquisition hardware is the unit that amplifies and digitizes the EEG signals, and feeds them to a computer: In our case, the amplifier unit is *BE Light* by EBNeuro [82], which is shown in Figure 3.1. It has 28 channels: 21 monopolar, 4 bipolar, and 3 monopolar with a separate reference (polygraphic). The maximum sampling rate is 8 kHz.

The unit is connected to a laptop through an optical fiber cable and a custom PC Card adapter. The use of optical fiber electrically isolates the computer from the amplifier and the subject. This is a standard procedure to ensure the respect of safety standards and avoid to endanger the subject. In our experiments, the computer is used also to stimulate the subject and lies very close to him. For this reason, the

Figure 3.1: EBNeuro *Be Light* EEG amplifier

computer is disconnected from the main power and runs on its internal batteries for all the duration of the experiments.

We used the same montage in all our experiments. As the potentials we are interested in, P300 and ErrP, are stronger on the midline of the head, the four monopolar channels Fz, Cz, Pz, Oz are used (see Figure 2.5 at page 8 for their positions). The reference is the right mastoid, and a mass electrode is placed on the forehead. Finally, two bipolar electrodes are placed near the right eye and measure the *electrooculogram* (EOG), i.e., a potential correlated with eye position; in the EOG channel are visible both eye movements and blinking. All electrodes are Ag/AgCl cup electrodes. They are applied by following the standard procedure: rubbing of the skin, application of the adhesive and conductive paste, and check of the impedance levels.

The BE Light applies a high-pass filtering at 0.1 Hz to all EEG and EOG channels, so that DC components and slow drifts are removed. Frequencies above 1 kHz are also removed by an anti-aliasing filter (the internal sampling rate of the ADC is 8.2 kHz), and a digital low-pass filter is applied before the signal is downsampled to the desired frequency. The effect of all the filters is that the recorded digital signal contains all the frequencies between 0.1 Hz and 0.45 times the selected sampling frequency (or up to 1 kHz for very high sampling frequency). The amplifier does not remove the 50 Hz power line interference, but experiments have shown that keeping cables, sockets, and AC adapters at least 0.5–1 m away from the subject is sufficient to have only negligible effects on the signal quality.

We used a frequency of 512 Hz for all our experiments, so the useful band according to the specifications is 0.1–230 Hz. Although both P300 and ErrP do not have significant components above 10–20 Hz, we preferred to record data at a higher frequency for possible future studies. After all, it is very easy to discard unwanted high frequencies or to downsample a signal with digital processing techniques, but it is impossible to recover frequencies filtered during recording.

The amplifier comes with a software suite, *Galileo*, which handles all the different tasks related to EEG acquisition, like controlling acquisition parameters, displaying recorded data, performing some data analysis, handling a database of recordings and subjects. Galileo has been thought for clinical practice, and it has more features than we use. Our main use of Galileo is to record data and export them for further processing. But the visualization of EEG data in real time is useful to verify that everything is working properly, and a quick look at the data just after the end of an acquisition permits to identify potential problems and possibly warrant the repetition of the acquisition. Real-time visualization has been also useful to instruct the subjects about avoiding movements and eye blinks, as they can appreciate the effect of their actions on the EEG tracks.

Galileo does not provide any facility to feed the EEG data to another application in real time. It writes data in a file, but it does so in batches of 4 s, so this feature cannot be used to build a real-time BCI. Therefore, we performed all our classification experiments *offline* initially. Data were acquired in Galileo, and only after the end of each acquisition session they were exported for analysis. Nonetheless, care has been taken that all the algorithms for processing and classification of potentials could be applied also online.

After having accumulated experience with EEG signal processing, and having tried the algorithm offline, we modified the system to work also online. The first element we needed for online processing was a way to get the EEG data from the amplifier. To this end, we wrote a plugin that runs inside Galileo and feeds the EEG data into a software pipe, where any other program can read them. Data are made available to the plugin in batches of 62.5 ms, with a delay of about 1–2 tenths of a second, which is low enough for real-time process (thanks also to the synchronization mechanism described in the next section).

3.1.1 EEG Synchronization

When doing experiments with ERPs induced by external stimuli, it is very important to have the EEG recording synchronized with the stimuli. The problem is that in the chain from the EEG electrodes to the application running on the PC, there are several elements (the amplifier unit, the PC Card adapter, the driver of such adapter), and each of them uses internal buffers and introduces a delay. Delays are

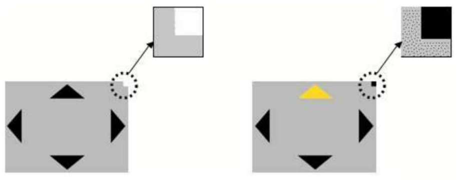


Figure 3.2: Detail of the colored square used for stimulus synchronization

present also in the stimulation chain. If stimuli are generated with a PC, after an application has issued a command for drawing the screen, time passes before the video driver, the video card, and finally the display update the current image.

If the stimulator application communicates the instants of stimulation to the EEG recording application, stimuli and EEG tracks are off by a noticeable amount of time. If both EEG recording and stimulation are done by the same computer all the delays add together, and the sum should lie in the range of the hundredths, possibly tenths, of second. If a computer records the EEG while another one stimulates the subject, the clocks of the two computers must be kept synchronized, or some other mechanism must be devised to mark the EEG with the instants of stimulation.

It is worth noticing that the actual offset of the stimuli on the EEG tracks is not very important for ERP analysis, as what really counts is how much the offset varies from stimulus to stimulus (as long as it is not very large). But many of the delays mentioned above are stochastic, as they depend on the task scheduler of the operating systems, or are likely to depend on the parameters of EEG acquisitions, like sampling frequency or the number of channels.

Because of all these reasons, and because the EEG recordings of our experiments were done with the Galileo software, which has no built-in mechanism to interface with external programs, we devised a synchronization mechanism that does not depend on a direct communication between the stimulator application and the EEG recording software. The idea is simple: The visual stimuli are picked up by the amplifier unit together with the EEG channels.

Figure 3.2 shows a screen shot of a visual stimulation used in one of our early P300 experiments. In the top right corner of the screen, a small square is visible, and it changes from white to black on the onset of each stimulus. A small optical sensor is placed on the display used for stimulations in correspondence with the square, and the signal from

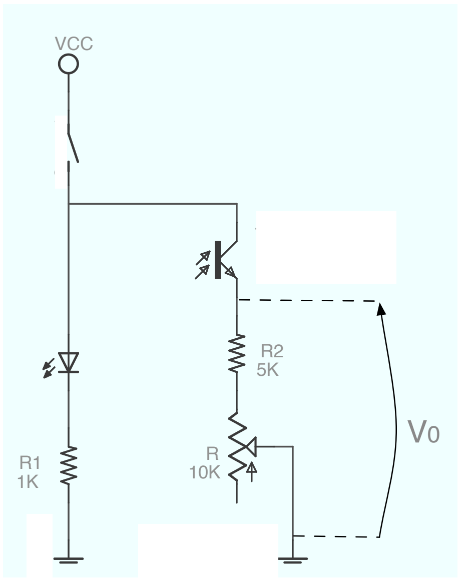


Figure 3.3: Scheme of the electronic circuit of the synchronization sensor

the sensor is fed to the EEG amplifier. The application that generates the stimuli also controls the color of the synchronization square, and care is taken that the stimuli and the square are drawn very closely in time (more on this later). All the other delays are removed, as the EEG and the synchronization signal travels together from the amplifier to the processing application.

The scheme of the electronic circuit used for connecting the optical sensor to the EEG amplifier is shown in Figure 3.3. It is very simple: the optical sensor is a phototransistor, and the current induced by light is converted in voltage by a resistor. The value of the resistance is such that the output voltage is suitable for the input of the EEG unity. As it depends on the luminance of the screen and the actual voltage of the power source, a trimmer is put in series to a fixed resistor to adjust the value. Power is provided by a lithium battery, so the insulation of the EEG unit is preserved. There is a switch to turn the circuit on and off, and an LED is used to signal the on/off state of the circuit. The phototransistor employed is directional, but its case is transparent and therefore the transistor is subject to the interference of external light sources. For this reason, a black sheet is put over the phototransistor and the synchronization square.

Because the circuit is very simple, the digitized signal require some software processing to identify the transitions of the synchronization

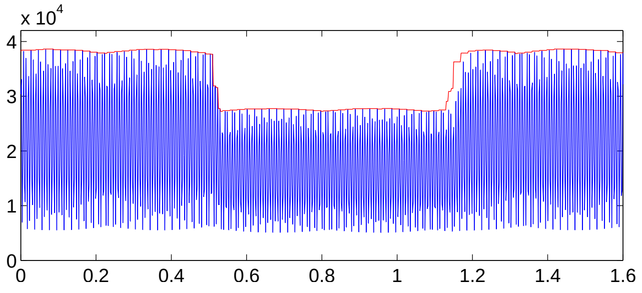


Figure 3.4: Synchronization signal in blue, and its envelope in red

square and hence the time instants of the stimuli. Figure 3.4 shows a typical synchronization signal recorded by the EEG unit when the synchronization square turns alternatively black and white. The sensor is connected to one of the 3 inputs of the BE Light that have a reference different from the one used for the subject, and allow DC coupling, i.e., no high-pass filter is applied. This is very useful, as the high-pass filter present on the standard EEG inputs would have altered the shape of the signal and made the detection of transitions harder. The intervals where the signal is stronger correspond to white, as white generates more current in the circuit than black.

The transitions are clearly visible in Figure 3.4, but there is a lot of noise when the screen is white. This is probably due to the refresh of the pixels in the LCD display. The simplest way to find transitions in a square wave is to compare the signal with a threshold, placed halfway between the maximum and the minimum. The noise may cause many false positive, and indeed it happened in our first attempt to find transitions, as the brightness of the screen was lower and the effect of the noise was stronger. Although tinkering with the screen brightness control helps, a more robust solution is computing the envelope of the synchronization signal before the comparison with the threshold. A simple function that find the maximum within a sliding window does the trick, and the result is shown in the figure (red line). The lag of the red line with respect to the actual signal at the rising front is not important, because only the falling front is used. As it is apparent from Figure 3.4, the transitions from white to black are faster and cleaner than those from black to white, at least for the display we used. For this reason, stimulus onsets are synchronized with the first kind of transitions in all our experiments.

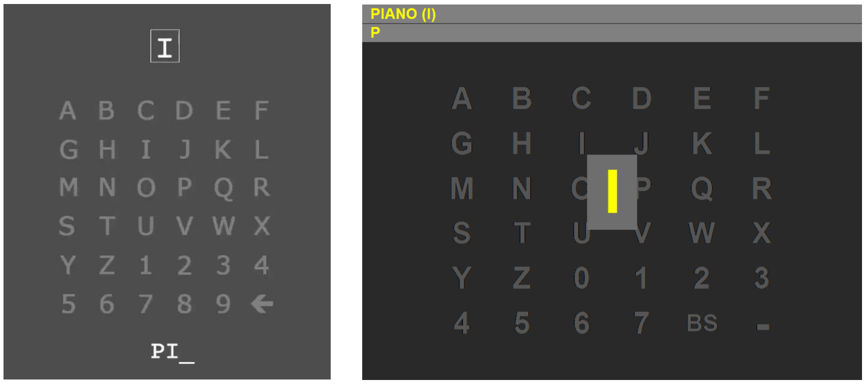
We verified that the synchronization system is adequate for ERP experiments by measuring the time offset between the appearance of the stimulation and the color switch of the synchronization square in the stimulation programs. For this purpose, we assembled a second optical sensor and placed it on screen in correspondence with the area of a stimulus. We recorded the two sensors at the same time, and by analyzing the resulting tracks we found that the maximum offset had been one hundredth of a second. That was very good, as features in ERPs are longer by at least a magnitude order.

As some experiments for ErrP require the pressing of buttons, we have verified that the response times of the program controlling such experiments are acceptable. We modified a USB keyboard by connecting one pole of a double-pole switch in parallel to a key, while the other pole of the switch was connected to an input of the EEG amplifier. In this way we could operate a key of the keyboard and have an exact reading of the time of that event with respect to the synchronization signal generated by the usual square. We measured the times of key pressings within the stimulation application and compared the two sequences of measures, one coming from the analysis of the tracks recorded by the EEG amplifier, the other from the program generating the stimuli. The result was that the response time to key pressing was always below two hundredths of a second, and often much less than that.

3.2 P300 Speller

We developed a classical BCI based on P300, the P300 speller, and integrated the use of ErrP in it. We developed two versions of the speller, one for offline use and a later version for online use. They are the same in concept, and they only differ slightly for appearance and timings; these differences are due to the fact that the two interfaces have been developed in different environments, and we preferred to use the parameters that simplified the development the most. We have not yet made any study to find out which one is the most effective way to present the stimuli and the feedback.

Our P300 speller is very similar in the appearance and in functioning to the one described by Donchin [56]: 36 symbols are disposed on a 6×6 grid, and entire rows and columns of symbols are flashed one after the other in random order. The grid of symbols is visible in Figure 3.5: There are the letters from the alphabet, digits, and the *backspace*, represented by the small arrow in the right bottom corner. The intensification of rows and columns lasts for 100 ms and the matrix remains blank for 100 ms between two consecutive flashes (125 ms were used for the online version). Each row and column is flashed exactly



(Offline) (Online)

Figure 3.5: Graphical interfaces of the P300 spellers used in the experiments. They both show the moment of the letter feedback used for ErrP-based confirmation.

once in the first 12 stimulations; then another round of 12 stimulations is repeated, with flashing of rows and columns done in a new random order, and this procedure is repeated for a total of 5 times. Each block of 12 consecutive stimulations is called a *repetition*, and there is no pause between repetitions.

After the fifth repetition, the P300 system detects the row and the column that are more likely to have elicited a P300, and selects the letter at their intersection. After a pause of 1 s, the letter is presented to the user in the rectangular frame in the top part of the interface for the offline version, and in a big rectangle that pops up in front of the grid for the online version (see Figure 3.5). The letter is also concatenated to the text which is at the bottom in the offline version and at the top in the online one. The presentation of the letter should elicit an ErrP if the letter predicted by the P300 system is different from the one the user intended.

An ErrP detection system figures out if any ErrP is elicited by the presentation of the selected letter, and in that case it overrides the P300 speller and cancels the last spelled letter. After a 2–3 s pause (this parameter is tuned to each subject's requirements), the speller starts a new series of stimulations for the next letter. A *trial*, in this context, is the whole series of 60 row/column flashes together with the feedback of the speller selection made for each letter, i.e., a single trial is composed of 60 P300 stimulations and 1 ErrP stimulation (a trial is about 15 s long).

During the training phase, i.e., when there is no trained classifier, and during offline experiments, subjects are told the letter they must pay attention to at the beginning of each trial, and no automatic error correction is done. In order to elicit also ErrPs, the letter feedback is chosen randomly so as to match the correct letter in 80% of the cases.

3.2.1 BCI2000

BCI2000 [83, 84] is a general-purpose software system developed at the Wadsworth Center of the New York State Department of Health in Albany, New York, USA, for brain-computer interface (BCI) research. It is made available complete with source code at no cost to research institutions, under a license that has some restrictions but permits to do research freely. It has been chosen as the platform to base our speller on because we deemed that it could speed up the development of the systems (if that has been the case is not clear, as BCI2000 is not as modular and flexible as it could be).

BCI2000 is composed by four processes which perform different tasks. A source process acquires data from an EEG device; a signal processing process performs all the data analysis and classification; an application process interacts with the user and possibly external devices; a supervisor process coordinates the other processes. The processes exchange data together with state and control signals via TCP/IP sockets. Different applications and data analysis algorithms are implemented in BCI2000; this could help in bootstrapping the development of new custom BCIs.

We developed three main components: a source module that acquires EEG data from the Galileo plugin described above, an application derived from the built-in P300 speller, and a dual-classifier processing module to handle both P300 and ErrP classification. The source module just feeds the EEG data read from the Galileo plugin to the rest of BCI2000 system. The application module implements the P300 speller with ErrP-based error correction, as described above. The processing module breaks the EEG signals in epochs synchronized on the stimulation instants recovered from the synchronization signal (see Section 3.1.1); it then processes the data and performs the classification of the epochs according to two separate processing chains, one for P300s and one for ErrPs (the two classifiers are described in Sections 5.4 and 6.1.1, respectively). The application module activates the classifiers in turn by changing a global BCI2000 state, and uses their output to select letters or cancel a selection.

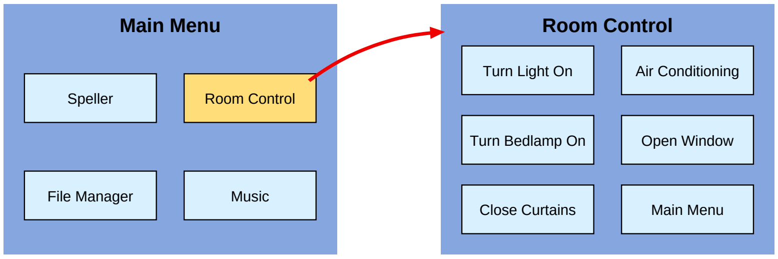


Figure 3.6: P300-based menu interface for control

3.3 P300 Menu System

The P300 speller is a part of a bigger system, based on a P300 BCI, that is currently under development. The objective of this bigger work is to develop a prototype of an assistive device that permits both communication and control, where the communication part is based on the P300 speller already described, and the control part is based on an interface with hierarchical menus selected by means of P300 detection. Each menu entry can lead to another menu, or perform an action, like running a program or operating an external device.

This system is similar to the P300 speller in many respects, as both work by eliciting P300 potentials in the user and detecting them, but there are differences in stimulation mechanisms (single entries are flashed for the menu, while entire rows and columns are flashed for the speller), and different is also the kind of attention that a user may pay to the task. These differences might elicit different variations of P300; this an interesting point for research.

Figure 3.6 shows the menu interface and how it works. An initial menu contains entries for different applications: the P300 speller, an interface for controlling some devices in the room, a file manager, and an application that plays music in the background. Entries in the current menu are highlighted one at a time, and, after a number of repeated stimulations, the detection of a P300 determines that the corresponding entry has been selected by the user. If the user selects “Room Control”, a new menu appears with one entry for each of the controllable devices, plus an entry to go back to main menu. The first entry, “Turn Light On” invokes an external application on the BCI computer, which acts on an external interface connected with some relays or a more complex device operating the main room light. The second entry, “Air Conditioning”, shows a new menu with commands for controlling temperature and fan speed. The other entries behave in a way similar to the first: They run an external program that operates the indicated device.

When a P300 is detected by the BCI, the corresponding menu entry is selected and this is shown on the screen by a specific highlighting. This can be used for detecting errors made by the BCI by means of ErrP. The effectiveness of this option should be checked on the individual users, as the impact of ErrP detection on the overall performance changes dramatically with its accuracy.

The use of P300 is a natural choice for a menu-based BCI like the one just depicted. The resulting interface can be consistent both in appearance and usage. It could also be extended to use a different mechanism for stimulation; for example, short audio clips reading the menu entries aloud could be used for people with visual difficulties. Although it is possible to use motor-imagery for making selections, e.g., by moving a pointer or a cursor on a computer screen, it is not easy to design such a graphical interface, and lack of precision in the control of the cursor limits the number of possible choices. P300 BCIs are effective even when the number of menu entries is big, as the P300 speller witnesses. Yet, the versatility of a menu-based interface permits to launch motor-imagery-based application when they are more suitable than P300-based ones. The fact that P300 is an involuntary and automatic potential may help the acceptance by potential users, as users do not have to acquire new skills. This point must be confirmed with experiments with potential users, though.

3.4 Autonomous Wheelchair

The idea of P300-based menus can be extended to more complex contexts, like giving commands to a robotic arm or driving a robotic wheelchair. For the wheelchair, a menu can contain a list of different locations, and the wheelchair reaches the selected location without further intervention of the user. Given the limitations of BCIs, in particular latency and information transfer rate, a wheelchair require a good degree of autonomy in order to be operated through a BCI. In this way, a user needs only to give high-level commands to such a device and not care about exact and precise movements.

Figure 3.7 shows the autonomous wheelchair of the *LURCH* (Let Unleashed Robots Crawl the House) project that is being developed in a different — but related —, project at Airlab. This project aims to provide an impaired person with a way to move autonomously in indoor environments [85].

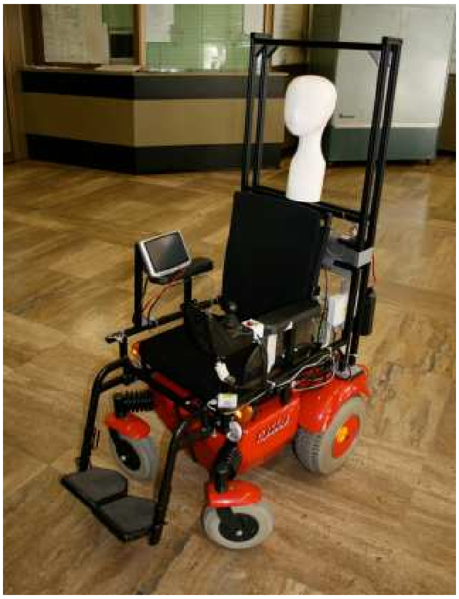


Figure 3.7: The wheelchair of the “Lurch” project used at Airlab

Our main motivation for applying a BCI to a wheelchair is to test a BCI in a more “noisier” environment and assess its robustness. Here, there are two sources of noise: When sitting on a wheelchair, a subject is very likely to get distracted by the surrounding environment, and his attention may be lower; secondly, the movement of the wheelchair is another source of distraction and may modify the shape of the potentials used by the BCI or induce artifacts.

The actual usefulness of a BCI-controlled wheelchair is debatable, and depends on the one hand on the needs of disabled subjects, and on the other hand on the accuracy that the BCI reaches. With low accuracy, people with residual movements may prefer joysticks, buttons, or other assistive devices to drive the wheelchair; more severely paralyzed people may simply not need to drive a wheelchair, as their priorities are more likely to be communicating with friends and caregivers, or operating objects like a lamp or a radio. Although there have been progresses in brain-actuated wheelchairs (e.g., the live demo of the Maia project in Leuven in 2007), we are not aware of any such experiment that involves totally paralyzed people driving a wheelchair through a BCI. Therefore, it is not possible to predict exactly whether or how much a BCI-controlled wheelchair will be accepted.

3.4.1 Wheelchair Architecture

The aim of the LURCH wheelchair project is to provide navigation assistance in a number of different ways, such as assuring collision-free travel, aiding in the performance of specific tasks (e.g., passing through doorways), and autonomously transporting the user between locations, while at the same time keeping the cost of the whole system as low as possible (the total cost of the framework for indoor environment, wheelchair not included, is less then five thousands of euros, which is cheap with respect to other works).

The wheelchair used for the project has been equipped with two embedded PCs, a video camera, an odometry system, an inertial measurement unit (not used in indoor environments) and two Hokuyo laser range finders (used for obstacle avoidance — on the way to be replaced with sonars); this equipment provides a self-localization capability and a safe navigation ability to the wheelchair. The smart wheelchair navigates, deals with all the low-level details, avoids obstacles and reacts to emergency situations, while the user decides where (and when) to go in a high-level fashion (e.g., "go to the kitchen"). In order to meet the variable requirements of disabled people, the system has been designed so that it can be modified simply and adapted to different users' needs. In particular, the user has the opportunity to choose among autonomy levels and three different interfaces: a joystick, a touch-screen and a BCI.

The LURCH system has been designed to be easily adaptable to different kinds of electric wheelchairs. Figure 3.8 outlines a scheme of LURCH. As it is possible to notice from the image, our system is completely separated from the wheelchair, and the only gateway between LURCH and the vehicle is represented by an electronic board that intercepts the analog signals coming from the joystick potentiometers, and generates new analog signals to simulate a real joystick and drive the electronics of the joystick interface board. In other words, we do not integrate our system with the wheelchair at the digital control bus level, but instead we rely on the simulation of the signals from the joystick in the analogue domain. Though this choice could seem awkward, its motivations are twofold: First of all, it is often hard to obtain the proprietary communication protocols of the wheelchair controllers, or to understand how they exchange data with motors and interfaces; second, this solution improves the portability to different wheelchair models, since it avoids a direct interaction with the internal communication bus of the wheelchair.

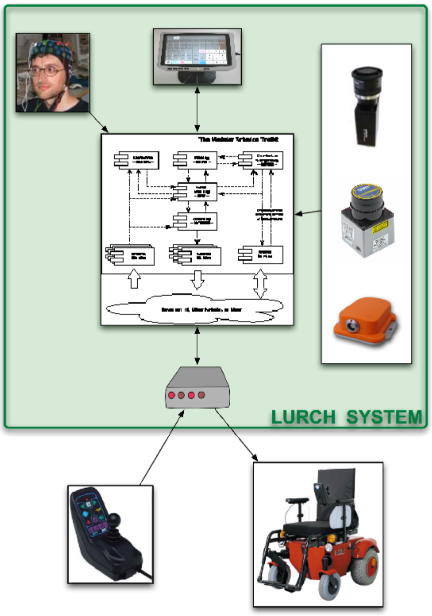


Figure 3.8: General scheme of LURCH system.

LURCH has been designed by adopting a modular approach (as proposed in [86]). There are three main modules:

- *localization module*: it estimates the robot pose with respect to a global reference frame from sensor data, using a map of the environment;
- *planning module*: this module selects the most appropriate actions to reach the given goals, using knowledge about the environment and the robot, while respecting the given task constraints;
- *control module*: it contains all the primitive actions, typically implemented as reactive behaviors that can be executed by the robot.

The localization algorithm makes use of a video camera pointing up and some *fiducial markers* placed on the ceiling of the environment, since this allows to avoid occlusions, and provides an accurate and robust pose estimation. Of course, this restricts the use of LURCH to indoor environments. Usually, a fiducial marker is a planar patch with a known shape that contains some encoded information. In this work,

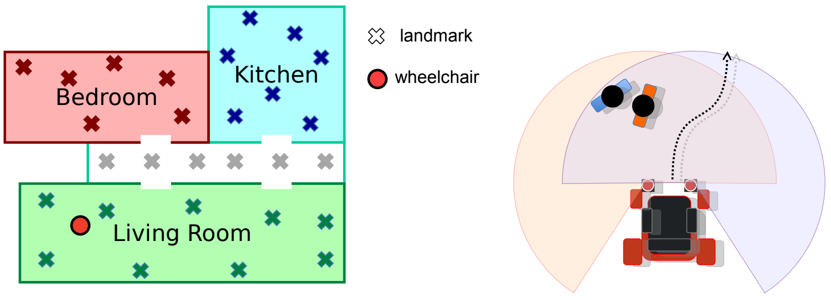


Figure 3.9: LURCH: Example of localization and obstacle avoidance.

the ARToolKitPlus [87] system is used, where the markers are squares with a black border, and the information is encoded in a black and white grid in the inner part of the square. The marker identification process is composed by three steps: identification of possible markers in the image captured by the camera, rectification of the image, and comparison of the information represented in the markers with the database of known landmarks. If a marker is recognized, with the knowledge of its dimension, it is possible to estimate its 6-DoF position and orientation of the camera and hence of the wheelchair. In indoor environments, it is generally sufficient to know the 3-DoF pose of the wheelchair; thus, we decided to simplify the problem, improving in this way the robustness and the accuracy of the localization algorithm.

The position and orientation of the markers in the environment is registered through an initial calibration process. This involves driving the wheelchair in a new environment where the markers have been placed so as too acquire images containing more than one marker; a calibration algorithm processes these images and computes the relative positions between the markers [88].

The trajectory planning is obtained by SPIKE (Spike Plans In Known Environments), a fast planner based on a geometrical representation of static and dynamic objects in an environment modeled as a 2D space (see [86] for more details). The wheelchair is considered as a point with no orientation, and static obstacles are described by using basic geometric primitives such as points, segments and circles. SPIKE exploits a multi-resolution grid superimposed to the environment representation to build a proper path from a starting position to the requested goal by using an adapted A^* algorithm; this path is finally represented as a polyline that does not intersect obstacles. Moving objects in the environment can be easily introduced in the SPIKE

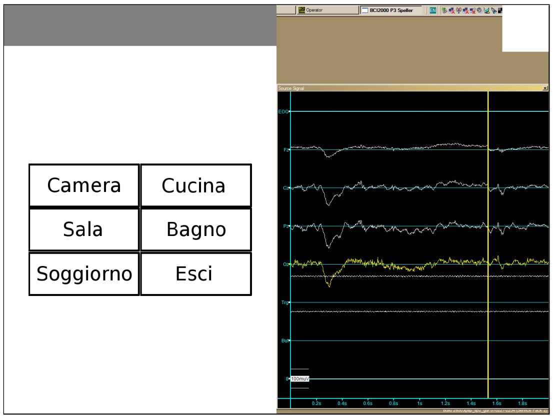


Figure 3.10: Screenshot of the graphical interface used to drive the wheelchair. Possible selections are destination rooms (in Italian).

representation of the environment as soon as they are detected, and they can be considered while planning. Finally, doors or small (with respect to the grid resolution) passages can be managed by the specification of links in the static description of the environment.

The control module is MrBRIAN (Multilevel Ruling BRIAN) [89], a fuzzy behavior management system, where behaviors are implemented as a set of fuzzy rules whose antecedents match context predicates, and consequents define actions to be executed; behavioral modules are activated according to the conditions defined for each of them as fuzzy predicates, and actions are proposed with a weight depending on the degree of matching (see [89, 90] for more details).

3.4.2 Wheelchair And BCI

We chose an incremental approach to the integration of the LURCH wheelchair with a BCI. The first integrated system, reported here, was developed with the aim to be very simple yet fully functional; for this reason, the communication between the BCI and the wheelchair has been kept to the minimum. The BCI permits to choose among six possible destinations, and it runs independently of the behavior of the wheelchair: After every round of stimulations, it selects the most likely target and sends it to the planning module of LURCH. This has the advantage that in case of error the user has a chance to select the correct target immediately, but at the cost of continuously focusing on the target while the wheelchair is moving. While this setting is not the most convenient from the point of view of usability and user comfort, it is very useful to test the robustness of the BCI as a communication system, as it has to work in a condition where the user can be distracted by the changing environment, and also artifacts may arise due to the movement of the user with the wheelchair.

A screenshot of the BCI graphical interface is shown in Figure 3.10. The possible destinations (not highlighted) are show on the left side, while on the right the EEG tracks are visible. The white square in the top right corner is the one used to synchronize the stimuli with the EEG (see Section 3.1.1).

3.5 Software Tools

For the work of the present thesis, many different pieces of software have been used; they are described in this section very briefly. Two pieces of software used for data acquisition have been already described: Galileo and BCI2000. For the offline experiments that elicit P300s and ErrPs, we have written a few custom programs; these programs display stimuli on a computer screen and record the stimulation sequence together with some ancillary information in a text file. In some ErrP experiments (described in Chapter 6), our programs record also the timing of buttons pressed by subjects.

Data processing has been done with Matlab [91], a tool for numerical programming. EEG data has been first exported from Galileo in EDF format [92, 93], and imported in Matlab with the help of functions from the BioSig toolkit [94, 95]. Weka [96, 97], a collection of machine learning algorithms written in Java, and LibSVM [98], an implementation of SVMs, have been used for classification. Matlab functions have been written which find stimulation instants on the synchronization track, segment EEG data according to the synchronization signal and the information stored by stimulation programs, and perform all the further processing steps according to the algorithms described in the next chapters.

4 Methods

We are met to discuss our plans, our ways, means, policy and devices.

J. R. R. Tolkien – *The Hobbit*

As explained in Section 2.2, a BCI must decide whether a recorded signal belongs to two or more possible classes. For example, a P300-based BCI classifies each stimulus as target or nontarget; a motor-imagery-based BCI classifies time segments as left-hand, right-hand, or foot movements. Normally this is accomplished in a two-step procedure. In the first step, the EEG signal is processed and some relevant features are extracted or computed; this is called *feature extraction*. In the second step, these features are fed to a procedure to make a decision; this is called *classification*. Both steps can be considered as mathematical functions; the first one maps real-world objects (or their representation) into a feature space where (hopefully) samples from different class are more separated. The second function assigns labels (classes) to elements in the feature space.

In this chapter, the algorithms used to process EEG signals and to recognize (i.e., classify) ERPs are presented.

4.1 ARX Models

As said before (see Section 2.2.1), event-related potentials (like P300) are buried in the ongoing EEG. Methods are needed to extract the interesting part of the EEG (the P300, in our case) from the recorded signal. In this section, a method is described that has been tested in the work of this thesis: ARX (AutoRegressive with eXogenous input) models; such a model has been already used for the detection of ERPs in EEG [99, 100].

Figure 4.1 shows the block diagram of an ARX model. In this diagram, the signal y results from the superposition of a stochastic signal and a deterministic one: The first (the upper one in the figure) is the output of a process with a white noise e as input; the second (the bottom one) is the output of an ARMA (AutoRegressive, Moving Average) system

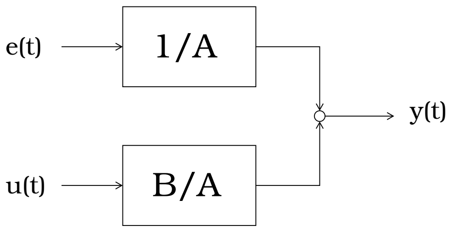


Figure 4.1: Block diagram of an ARX model

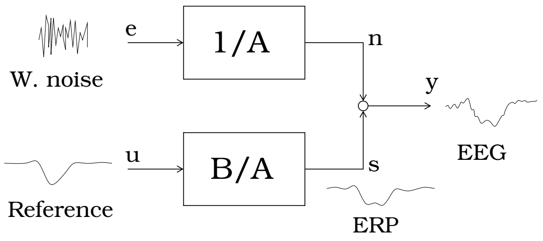


Figure 4.2: Usage of an ARX model for the modeling of ERPs

with a fixed input $u(\cdot)$. In formulas:

$$y(t) = \sum_{k=d}^{q+d-1} b_k u(t-k) - \sum_{j=1}^{p} a_j y(t-j) + e(t) \ , \tag{4.1}$$

where a_j and b_k are the coefficients of the ARMA system, q and p are their orders (i.e., the number of coefficients), and d is the delay between the input and the output. Equation (4.1) can be written also as

$$A(z)Y(z) = B(z)U(z) + E(z) \tag{4.2}$$

by using the Z transform [101]; $Y(z)$, $U(z)$, $E(z)$ are the Z transforms of $y(\cdot)$, $u(\cdot)$, $e(\cdot)$ respectively, and

$$A(z) = 1 + \sum_{j=1}^{p} a_j z^{-j} \tag{4.3}$$

$$B(z) = \sum_{k=d}^{q+d-1} z^{-k} \ . \tag{4.4}$$

When applied to extraction of an ERP from an EEG recording, the recording $y(\cdot)$ can be seen as a superposition of two contributions:

$$y(t) = s(t) + n(t) \tag{4.5}$$

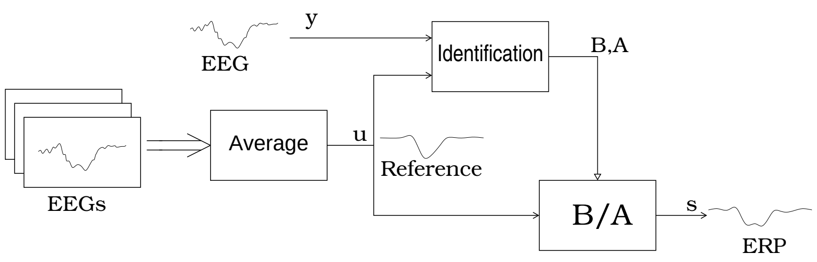


Figure 4.3: Data flow in using an ARX model for ERP extraction

where $s(\cdot)$ represents the ERP component, and $n(\cdot)$ represents the noisy component, i.e., the background EEG. The two components are modeled by the two blocks in the ARX models (see Figure 4.2); more precisely, the AR block models the ongoing EEG, an approximation justified by several studies [102, 103], while the ARMA block filters the reference signal to get the ERP part. The reference input to the ARMA block is a pattern that resembles the characteristics of the ERP to be detected. It is usually obtained by averaging many EEG recordings where the ERP is supposedly present. The rationale behind this is that the ERP component $s(\cdot)$ is similar — but not exactly equal — to the ERP average. While averaging extracts the ERP component from many recordings where an ERP is known to be present, the objective of the application of the ARX model is to extract an ERP component from a single recording. So, it can be used, for example, to discriminate between stimuli that elicited a P300 and those the did not.

How to use ARX modeling for ERP extraction is shown in Figure 4.3. The analysis works on segments (called *epochs*) of EEG recordings, long enough to cover the expected duration of the ERP with some margin. An initial batch of epochs of EEG recordings where the ERP is present is used to build the reference signal $u(\cdot)$, by averaging. For every new EEG recording the model in Equation (4.1) is identified, with $y(\cdot)$ being the recording to be analyzed and $u(\cdot)$ the previously computed ERP average. Identification of the ARX model is performed by using a least-squares method, which minimizes the figure

$$J = \frac{1}{N} \sum_{t=1}^{N} e(t)^2 , \tag{4.6}$$

where N is the number of samples in an epoch. $e(\cdot)$ is the prediction error of the model:

$$e(t) = y(t) - \hat{y}(t) \tag{4.7}$$

where

$$\hat{y}(t) = \sum_{k=d}^{q+d-1} b_k u(t-k) - \sum_{j=1}^{p} a_j y(t-j) \tag{4.8}$$

After the identification, the useful component, $s(\cdot)$, can be computed by filtering the reference signal $u(\cdot)$ with the filter $B(z)/A(z)$, according to Equation (4.1):

$$s(t) = \sum_{k=d}^{q+d-1} b_k u(t-k) - \sum_{j=1}^{p} a_j s(t-j)\ . \tag{4.9}$$

The identification and filtering steps are performed separately for each candidate EEG epoch. For P300 studies, it means one epoch per stimulus.

The resulting signal $s(\cdot)$ hopefully contains the ERP when it exists, and just noise otherwise. It can be used for a visual inspection by a clinician, or it can be further processed by a computer in order to classify it as containing the ERP in question or not containing it. For a BCI, an automatic classification is obviously required. How this has been done in our case is discussed in Section 5.2.

Before an ARX model can be used at all, the parameters p, q, and d must be chosen. Normally, the choice is made by analyzing a training data set, i.e., a set of EEG recordings for which it is known whether they contain an ERP or not. The choice of the parameters p, q, and d cannot be made by using a least-square criterion as in Equation (4.6), because ever increasing the number of the parameters always makes the model fit better. The magnitude of the contribution to the goodness of the fit of new parameters must be taken into account. Akaike Information Criterion (AIC) [104] does exactly that. It minimizes the figure

$$AIC = 2n - 2\log L\,, \tag{4.10}$$

where n is the number of parameters of a model, and L is the likelihood of the parameters (i.e., the probability of the data given the identified model). In the case of ARX models, it can be written as

$$AIC = 2n + N\log\sigma^2\,, \tag{4.11}$$

where σ^2 is the variance of $e(\cdot)$ given by Equation (4.7), $n = p + q$, and N is still the number of samples. The equivalence of the Equation (4.10) and (4.11) depends on the error $e(\cdot)$ being white, i.e., values at different time instants are independent. This assumption must also be tested when using Equation (4.11). A suitable test for this is the Ljung-Box test [105]. Briefly, the Ljung-Box test computes a statistic on the error $e(\cdot)$, whose distribution is known (a χ^2) if $e(\cdot)$ is indeed white; the test rejects the hypothesis that the error is random if the value of the statistic is above a given percentile threshold.

4.2 Classification

A classifier is a function that assigns labels to objects. A learning algorithm is a procedure that finds a good classifier given a set of labeled examples. In more formal terms, the problem can be stated as following: find a "suitable" classifier function $f : X \rightarrow Y$, where X is a (feature) space (say $X \subseteq \mathbb{R}^n$), Y a set of possible labels (e.g., $Y = \{-1, +1\}$, i.e., a binary classification problem), given a probability distribution $p(\cdot, \cdot)$ defined over $X \times Y$, and a training set of pairs $\langle \boldsymbol{x}_i, y_i \rangle$ where $\boldsymbol{x}_i \in X$, $y_i \in Y$, $i = 1 \ldots N$.

Often, f is constrained to belong to a family $\mathcal{F}$ of classifiers, and it is written as $f(\cdot, \alpha)$, where α is a parameter that identifies a particular classifier in $\mathcal{F}$. A good way to define more precisely what "suitable" means is to look for the f that minimizes the expected number of errors, i.e.:

$$R(\alpha) = \int \frac{1}{2}|y - f(\boldsymbol{x}, \alpha)| p(\boldsymbol{x}, y) d\boldsymbol{x} dy \tag{4.12}$$

for the binary case. $R(\alpha)$ is called *expected risk*. Formula (4.12) depends on α, obviously, but also on the distribution $p(\cdot, \cdot)$. This distribution is unknown in practice, and normally it is approximated with the empirical distribution induced by the training set, and Equation (4.12) becomes

$$R_{\mathrm{emp}}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} |y_i - f(\boldsymbol{x}_i, \alpha)| \, , \tag{4.13}$$

which is called *empirical risk*.

Using (4.13) to find a good classifier may seem a good idea, because $R_{\mathrm{emp}}(\alpha) \rightarrow R(\alpha)$ as N increases (and samples are independents). The problem is that for a finite N the difference between (4.12) and (4.13) may be significant. This problem is called *overfitting*, and it means that the selected classifier $f(\cdot, \alpha)$ performs very well on the training examples but poorly on generic samples $\boldsymbol{x} \in X$. The capacity of *generalization* of the classifier, i.e., the ability of a classifier to perform well on new samples, can be obtained even when using Equation (4.13) for training, as long as there is some limit to the complexity of the family $\mathcal{F}$. How and how much to limit this complexity is still an open question, on which there is much ongoing research.

The following sections present two of the classifiers we used; their understanding is important for the understanding of the next chapters, and for this reasons they are described in some details. The other classifiers are very shortly described where they are mentioned.

4.2.1 Logistic Classifier

A *logistic classifier* [106] approximates the probability $\mathrm{P}(y \mid \boldsymbol{x})$ with a logistic function:

$$\mathrm{P}(y=+1 \mid \boldsymbol{x}) = \frac{1}{1+\exp(w_0 + \sum_{j=1}^{n} w_j x_j)} \tag{4.14}$$

$$\mathrm{P}(y=-1 \mid \boldsymbol{x}) = 1 - \mathrm{P}(y=+1 \mid \boldsymbol{x}) = \frac{\exp(w_0 + \sum_{j=1}^{n} w_j x_j)}{1+\exp(w_0 + \sum_{j=1}^{n} w_j x_j)}\,, \tag{4.15}$$

where x_j are the n components of the vector $\boldsymbol{x}$. The decision of the class to assign to a given sample $\boldsymbol{x}$ is taken by comparing the two probabilities $\mathrm{P}(y=-1 \mid \boldsymbol{x})$ and $\mathrm{P}(y=+1 \mid \boldsymbol{x})$. The parameter vector $\boldsymbol{w}$ can be found by maximizing its log-likelihood

$$L(\boldsymbol{w}) = \sum_{i=1}^{N} \log \mathrm{P}(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) \tag{4.16}$$

by using gradient ascent. In order to improve the generalization ability of the classifier, a penalization term can be added:

$$L^{(\lambda)}(\boldsymbol{w}) = \sum_{i=1}^{N} \log \mathrm{P}(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) - \lambda \|\boldsymbol{w}\|^2\,. \tag{4.17}$$

The additional term penalizes large values of $\boldsymbol{w}$ components. In other words, less importance is given to the directions in the space X that had the most discriminative power with (4.16), and the resulting classifier is more balanced in X. The parameter λ determines how strong the penalty term is.

4.2.2 Support Vector Machines

A *support vector machine* (SVM) is a supervised learning method used for classification and regression. It was developed by Vladimir Vapnik in the late 1970s, while he was addressing the problem of the generalization of a classifier from a theoretical point of view.

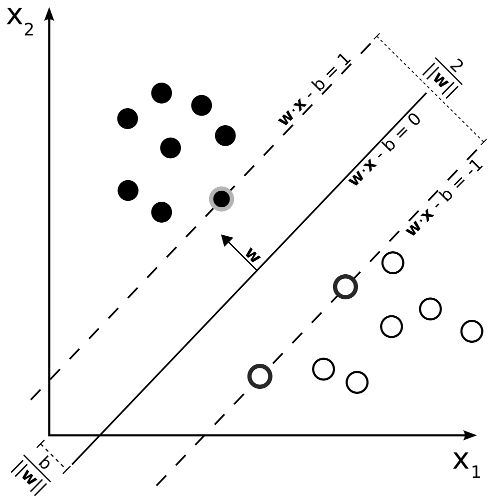


Figure 4.4: SVM: the maximum-margin hyperplane in the separable case. Support vectors are surrounded by a circle.

Vapnik found theoretical bounds on the expected risk given the empirical risk. For a binary problem stated as in Section 4.2, it holds

$$R(\alpha) \leq R_{\text{emp}}(\alpha) + \sqrt{\frac{h(\log \frac{2N}{h} + 1) - \log \frac{\eta}{4}}{N}} \tag{4.18}$$

with probability $1 - \eta$, for every η, $0 < \eta < 1$; h is the Vapnik-Chervonenkis (VC) dimension, a number that measures[1] the complexity of the family $\mathcal{F}$. It is interesting that the second term of the right-hand side of (4.18) does not depend on the distribution $p(\cdot, \cdot)$; also, it grows with h. So Vapnik proposed to keep the expected risk low by minimizing (4.18), an approach that he called *structural risk minimization*. On this principle he developed support vector machines.

In the simplest case, an SVM is a hyperplane in the space X. This hyperplane separates the space in two regions, one for each labels. Samples are assigned labels depending on which side of the hyperplane they lie (see Figure 4.4). In formulas:

$$f(\boldsymbol{x}) = \text{sign}(f^*(\boldsymbol{x})) \tag{4.19}$$

$$f^*(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b\,. \tag{4.20}$$

If the training set is such that there exist a hyperplane that separates exactly the positive and negative samples, the SVM maximizes the

[1] A more in-depth and detailed explanation is beyond the scope of the present work. For more details, see for example [107, 108]

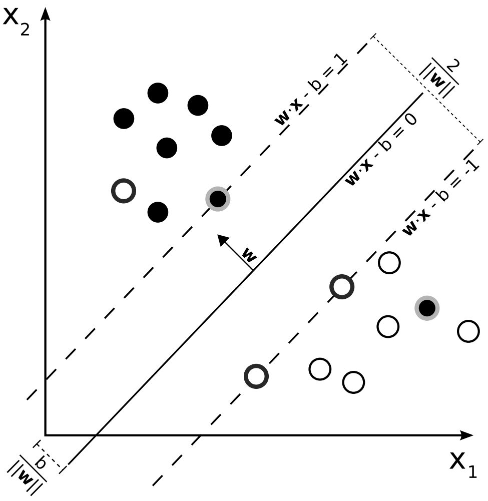


Figure 4.5: SVM: the maximum-margin hyperplane in the non-separable case. Support vectors are surrounded by a circle.

distance of the hyperplane from the nearest samples. Equivalently, the SVM maximizes the distance between positive and negative samples along the direction $\boldsymbol{w}$. Such distance is called *margin* in SVM terminology. Margin maximization is a way to lower the VC-dimension of the SVM, and hence minimize the structural risk [107]. The training samples that are closest to separating hyperplane, i.e., those for which it holds

$$y_i \left(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \right) = 1 \tag{4.21}$$

are called *support vectors*. The reason is that the problem of finding the maximum margin is equivalent to minimize $\|\boldsymbol{w}\|^2$ subject to

$$y_i \left(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \right) \geq 1 \tag{4.22}$$

By introducing *Lagrange multipliers* α_i, $i = 1 \dots N$, it is possible to show [108] that for the optimum $\boldsymbol{w}$ it holds

$$\boldsymbol{w} = \sum_i \alpha_i y_i \boldsymbol{x}_i \tag{4.23}$$

and $\alpha_i \neq 0$ only for the $\boldsymbol{x}_i$ that are support vectors.

If the training set is such that there is no hyperplane that separates positive and negative samples, some samples are necessarily misclassified by any hyperplane. In this case, minimizing $\|\boldsymbol{w}\|^2$ is not enough, but the SVM has to find a trade-off between the maximization of the margin and the minimization of the errors. In mathematical terms it can be written by introducing *slack variables* ξ_i, $i = 1 \ldots N$, and the SVM minimizes

$$\|\boldsymbol{w}\|^2 + C \sum_i \xi_i \tag{4.24}$$

subject to

$$y_i \left(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \right) \geq 1 - \xi_i \tag{4.25}$$

$$\xi_i \geq 0 \tag{4.26}$$

The parameter C can be varied to shift the trade-off between margin and errors. Even in this case, for the optimum $\boldsymbol{w}$ (4.23) holds, but $\alpha_i \neq 0$ is true also for all the misclassified samples, for which $\xi_i \neq 0$ (see Figure 4.5).

Although useful in some cases, a linear classifier proves to be inadequate in the majority of real cases. It is possible to extend the linear SVMs seen so far and make them non-linear in a simple and straightforward way. The trick is to map samples $\boldsymbol{x}$ in a higher-dimensional space $\mathcal{H}$ by means of a non-linear mapping $\boldsymbol{\Phi} : X \to \mathcal{H}$. The separating hyperplane is now to be found in $\mathcal{H}$. A good hyperplane is more likely to exist in $\mathcal{H}$ than in X, because the number of dimensions of $\mathcal{H}$ is greater than that of X, and hence data are more sparse. It is also possible for $\mathcal{H}$ to be infinite dimensional, but there must be a way to compute inner products in such a space. Inner products are needed because they appear in (4.20) and in the formulas used to find the α_i (not shown here). It turns out that is possible to avoid computing inner products with the so-called *kernel trick*.

Putting together (4.20) and (4.23) yields

$$f^*(\boldsymbol{x}) = \langle \sum_i \alpha_i y_i \boldsymbol{x}_i, \boldsymbol{x} \rangle + b = \sum_i \alpha_i y_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + b \,, \tag{4.27}$$

and in the mapped space $\mathcal{H}$

$$f^*(\boldsymbol{x}) = \langle \sum_i \alpha_i y_i \boldsymbol{\Phi}(\boldsymbol{x}_i), \boldsymbol{\Phi}(\boldsymbol{x}) \rangle + b = \sum_i \alpha_i y_i \langle \boldsymbol{\Phi}(\boldsymbol{x}_i), \boldsymbol{\Phi}(\boldsymbol{x}) \rangle + b \,. \tag{4.28}$$

By using a suitable[2] $\boldsymbol{\Phi}$, it is possible to find a *kernel function* K such that $\langle \boldsymbol{\Phi}(\boldsymbol{x}_i), \boldsymbol{\Phi}(\boldsymbol{x}) \rangle = K(\boldsymbol{x}_i, \boldsymbol{x})$, where $K(\cdot, \cdot)$ is much easier to compute than the inner product in $\mathcal{H}$. (4.28) becomes

$$f^*(\boldsymbol{x}) = \sum_i \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b \,. \tag{4.29}$$

Examples of kernels are:

[2] "Suitable" means in fact that $\mathcal{H}$ and $\boldsymbol{\Phi}$ satisfy Mercer's condition [108].

Polynomial $K(\boldsymbol{x},\boldsymbol{y}) = (1 + \langle \boldsymbol{x},\boldsymbol{y} \rangle)^d$, where d is a parameter.

Gaussian $K(\boldsymbol{x},\boldsymbol{y}) = \exp(-\frac{\langle \boldsymbol{x},\boldsymbol{y} \rangle^2}{2\sigma^2})$, where σ is a parameter.

Perceptron $K(\boldsymbol{x},\boldsymbol{y}) = \tanh(b \langle \boldsymbol{x},\boldsymbol{y} \rangle - c)$, where b and c are parameters.

With the introduction of kernels, SVMs become a *family* of classifiers. The choice of an appropriate kernel for a given data set is still an open issue; there are no predefined rules for selecting kernels. Often, one tries many kernels, with different values of C, and chooses the combination that has performed better.

4.3 Genetic Algorithms

Genetic algorithms (GAs) are a class of optimization algorithms that mimic — in some respects — the way natural evolution works. These algorithms work by considering potential solutions to the problem, evaluating them, and combining parts of good solution in order to find better candidate solutions. The range of problems solved with genetic algorithm is very wide; they have been used in scheduling, budgeting, optimization of networks, and many classical problems of operations research.

The father of genetic algorithms can be considered John Holland, who worked on them in the 1970s at the University of Michigan [109], although a group at the Technical University of Berlin (Ingo Rechenberg, Hans-Paul Schwefel, and Peter Bienert) worked on *evolution strategies* [110] at the same time, an approach similar to GAs, but different in some important aspects. The Atlantic Ocean separated the two groups, which worked independently and unaware of each other for a while. Work in the field continued since, and many variants have been developed [111].

Candidate solutions for the problem are encoded in chromosome-like data structures (called *chromosomes*), which often are just binary strings. Genetic algorithms work on a subset of all the possible solutions, which is called *population*. A genetic algorithm begins with an initial population of chromosomes, which are normally chosen randomly (see Figure 4.6). At every iteration of the algorithm, first all the solutions represented by the chromosomes are evaluated with the respect to the optimization problem. This is operation is in fact the *evaluation* of the so-called *fitness function*. The fitness function is a measure of the goodness of the parameters encoded in a given chromosome. Fitness values are used to select individuals from the population. The actual

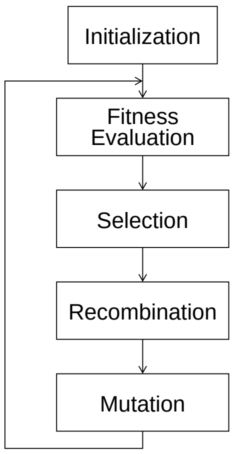


Figure 4.6: General scheme of a genetic algorithm

selection process may be done in different ways, and an individual may be selected more than once, but in any case "fitter" individuals have greater chance to be selected. Examples of selection are: the best n individuals; roulette wheel, where individuals are picked at random with a probability proportional to their fitness; tournament, where many independent tournaments between randomly-chosen individuals are performed, the winner of a tournament being the individual with the best fitness.

Recombination and mutation are then applied to the individuals selected in the previous step. *Recombination* (also *crossover*) is applied to the selected population in pairs: randomly-selected parts of the two chromosomes are exchanged, so as to form new, different individuals. Normally, recombination is applied only with a given probability. *Mutation* is the flipping of bits of the chromosomes (when they are binary strings). Typically, mutation is applied to all the bits of chromosomes with a very low probability (less than 1%). After mutation has been applied, a new population is ready, and the algorithm restarts from the evaluation. In GA terminology, a *generation* is the execution of evaluation, selection, recombination, and mutation.

Generation after generation, the fitness of the population increases, and thus better and better solutions are found. The process is terminated by some criterion. It could be something like "until the optimum is found", but there are two problems: Maybe there is no test for optimality, or maybe the time for finding the optimum is too much, and obtaining a good yet sub-optimal solution is enough. So, normally a genetic algorithm terminates after a predefined number of generations, or after a “good enough” solution has been found, or when no improvement has been seen for some generations.

Genetic algorithms have been defined as a *class* of algorithms, because even after choosing a selection scheme, a termination criterion, and the all various parameters (e.g., mutation probability), the result is still a schema of an algorithm and not an actual algorithm. The encoding of solutions in chromosomes depends on the problem at hand, and a new encoding must be devised for every new problem. The fitness function is at the heart of GAs, and it contains the description of the original optimization problem, rewritten in terms of chromosomes. Thus, there is at least one fitness function for every optimization problem, given the encoding. After defining an encoding and a fitness function, all elements are in place to run an actual genetic algorithm. Sometimes, though, the encoding for a particular problem makes use of structures that are more complicated than a plain string of bits. In such cases, recombination and mutation operators must be rewritten accordingly; the algorithm described in Section 5.4 is an example of this.

5 P300 Detection

To those who enter verily into Eä each in its time shall be met at unawares as something new and unforetold.

J. R. R. Tolkien – *The Silmarillion*

In this chapter some experiments made on detecting the P300 are presented, with details about how the methods presented in Chapter 4 have been used. Numerical results for the classifications are given, along with a consideration arising from the analysis of some aspects of the experiments.

5.1 Data Sets

The experiments with P300 illustrated in this chapter were performed on data from three different sources: recordings at Airlab — our laboratory —, at the S. Camillo Hospital, and from the BCI Competition 2003.

5.1.1 Airlab Speller Data Set

We recorded data with the speller described in Section 3.2. Seven young healthy subjects (2 female and 5 male) participated to one or three sessions recorded in different days, each one consisting in the spelling of some words. Each session was divided in 6–7 runs, each consisting of 4–5 words, for a total of about 150–200 letters per session. Data were recorded in a continuous fashion from four EEG channels, Fz, Cz, Pz, and Oz (see Figure 2.5 at Page 8), and one EOG channel, all sampled at 512 Hz.

5.1.2 S. Camillo Data Set

Other data were kindly provided by Francesco Piccione and Stefano Silvoni, recorded at IRCCS S. Camillo Hospital, Venice. These data are divided in two data sets, and are from experiments like the one described in [61], where subjects had to move a cursor on a computer screen and reach a cross, or from similar experiments where subjects had to move the cursor to one of four possible targets placed at the edge of the screen. In all these experiments, movements are discrete, and four directions are possible: up, down, left, right. Four arrows are present near the margins of the screen, one for each direction, and they are flashed randomly, with each flash representing a stimulus. Flashes last for 150 ms, and are separated by 2.5 s from each other. In this setting, when an arrow flashes which indicates the direction the subject wants to move the cursor, a P300 should be elicited in the subject's brain. EEG is recorded at positions Fz, Cz, Pz, and Oz of the 10-20 system; EOG is recorded too. All signals are band-passed between 0.15 and 30 Hz, and digitized and sampled at 200 Hz. Digitized EEG data are saved on a hard disk as epochs beginning 0.5 s before the stimulus onset, and ending 1 s after it.

In a first data set, data were recorded in different sessions in a period of a few days from 11 subjects, 5 female and 6 male, 8 healthy and 3 affected by ALS, of ages between 21 and 43, except for a subject who was 75. A second data set is composed by recordings made at S. Camillo some months later (with different subjects), and again different sessions were recorded in a period of a few days; 10 subjects affected by ALS and 4 healthy subjects participated to the second study, 3 females and 7 males, with ages between 31 and 73, in the ALS group, and 2 females and 2 males, with ages between 27 and 41, in the control group. All participants underwent neuropsychological evaluation and auditory odd-ball P300 testing, in order to exclude cognitive deficits. All participants had preserved auditory, visual, and cognitive functions, including adequate language comprehension.

5.1.3 BCI Competition Data Set

We used also the data set *IIb* from the BCI Competition 2003 [60, 59], originally provided by the group lead by Jonathan R. Wolpaw at Wadsworth Center, NYS Department of Health. These data come from three recording sessions with a P300 speller, with a grid of 6×6 letters. A P300 speller has been already described in details in Section 3.2; the Wadsworth speller works in the same way, except for the fact that there is no ErrP detection and some parameters are different: matrix flashing lasts for 100 ms, and 75 ms separates two consecutive flashes; moreover, 15 repetitions[1] of the stimulation rounds are done for each letter.

Data were acquired from 64 EEG electrodes, in all the positions of the 10-10 system (see Figure 2.5 at Page 8), with a sample frequency

[1] The description of the data set uses the word "trial" for a round of stimulations. For consistency, the word "repetition" is used in the present work.

of 240 Hz. In the first two recording sessions, 11 words were spelled, for a total of 42 letters; the actual words spelled were available for the competition, so the first two sessions constitute the training set. The third session, 8 words and 31 letters, is the test set. The words spelled in the third session have been made available after the end of the competition, and we used this piece of information for the evaluation of our methods.

5.2 ARX Models

ARX models, described in Section 4.1, have been already used in the literature [99, 100] to extract various ERPs from ongoing EEG recordings. For this reason, we applied the ARX model to the extraction of P300s as the initial step in automatic P300 detection for a BCI. This section explains how the ARX model have been used, the details of the classification, and the results we obtained.

5.2.1 ARX Filtering

The model is described by Equation (4.1):

$$y(t) = \sum_{k=d}^{q+d-1} b_k u(t-k) - \sum_{j=1}^{p} a_j y(t-j) + e(t) \ , \tag{5.1}$$

The orders (p, q, d) of the model are determined with Akaike's criterion (AIC), and the Ljung-Box test (see Section 4.1). Here the procedure is described in details:

- EEG data are segmented in *epochs*. Each epoch begins 0.5 s before the stimulus onset and ends 1 s after it. This interval is much longer than the duration of a P300, because the extra time is needed for a good identification of the AR part of the model.
- Target (P300) epochs are averaged together.
- A maximum value for each of the three ARX parameters (p, q, d) is selected.
- For every possible combination of values for p, q, and d, and for every epoch and every channel, an ARX model is identified. The prediction error $e(\cdot)$ is computed for every identified model.
- From the prediction error, the AIC value and the Ljung-Box statistic are computed for each of the models at the previous step.

- Combinations of p, q, and d for which less than half of the epochs have random residuals are discarded. This is an attempt to strike a balance between avoiding non-random residuals, i.e., underfitted models, and avoiding that a number of epochs become overfitted. The threshold of 50% appeared to be robust, as modifying it did not influence much the final results.

- AIC values from different epochs relative to the same combination of p, q, d, and channel are added together. These step is justified by the fact that models from different epochs are considered independent, and hence their log-likelihoods can be summed. For every combination of p, q, d, and channel, the fraction of epochs that passed the Ljung-Box text — i.e., those whose residuals appear to be random — is computed.

- Finally, for every channel the best combination of p, q, and d is selected, i.e, the one that minimizes the AIC sum and passed the previous step.

This procedure is applied separately to different data sets, and to data coming from different subjects. While results may be different for different channels, only one order triplet (p, q, d) per set/subject is used in the further processing to simplify the implementation of the ARX filtering. Values for different channels are rather similar, and the procedure to select them contains many stochastic steps.

After the choice of the model orders, the ARX method is applied as described in Section 4.1 and in particular in Figure 4.3: For each EEG channel, the target average is used as a reference to identify an ARX model, and the output of the ARMA block is considered to be an approximation of the underlying P300 event. The actual procedure is a little more involved, as it includes some preprocessing of the data and some tricks to help the generalization of the final classifier. Here it is described in full.

Given two data sets of epochs, one for training and the other one for testing:

- Remove any trend from each epoch. This is done by fitting a line to each epoch signal and then subtracting it.

- Split the training data set in two. If data come from different sessions, single sessions are not split, but each session is assigned to either group. The sizes of the two groups must be similar, but not necessarily the same.

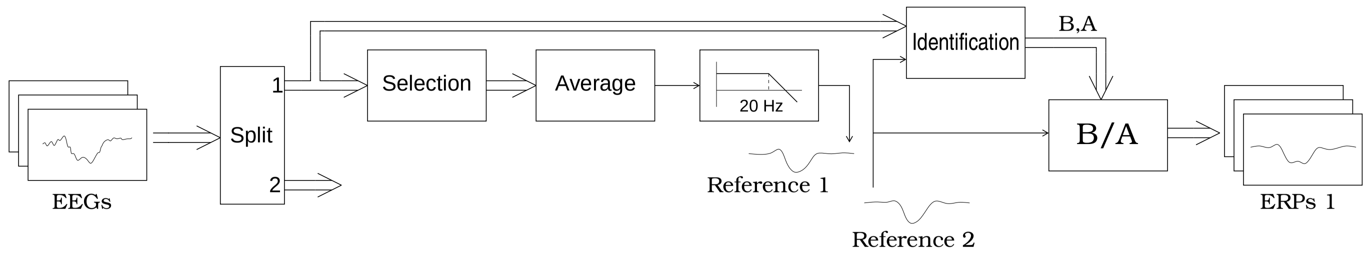


Figure 5.1: Detailed procedure of the processing of the training set with the ARX method. Double lines are used for connections where signals from many epochs travel.

- For each of the two training groups, in parallel (see Figure 5.1):
  - Temporarily discard the most "noisy" target epochs and compute the average of the remaining ones. An epoch is considered noisy if it differs from the full average by at least two standard deviations in at least one time sample. Averages are computed separately for each channel.
  - Filter the target averages with a low-pass filter with a cutoff frequency of 20 Hz. This step should make the averages of the two groups more similar.
  - For every epoch and channel, identify an ARX model with the target average coming from the other group as input, and compute the response of the ARMA block.

- Compute the weighted averages (again, one for each EEG channel) of the target averages of the two groups.

- For every epoch and channel of test set:
  - Identify an ARX model with one of the weighted averages as input, and compute the response of the ARMA block.

Figure 5.1 shows one half of the processing applied to the training set. The other half is symmetrical, and has not been depicted to simplify the figure. This procedure is a refinement of the basic procedure in which the training set is processed in one batch. The rationale for the splitting of the training set is to make the processing of the training set more similar to the processing of the test set. The identification of the ARX models for the test set is done with a reference that necessarily comes from other sessions. By splitting in two the train set, it is possible to have a similar situation also for the training set.

Several adjustments and modifications had been made to the ARX-based procedure for P300 analysis described above, before it took that form. For example, attempts were made with the application of bandpass filtering with different frequencies before ARX identification; we used the difference between the averages of target and nontarget epochs as the reference signal $u(\cdot)$. We also tried to duplicate the ARX filtering scheme of Figure 4.3, with one copy using the average of targets as reference, and the other one using the average of nontargets: Each epoch was processed by the two filters in parallel, and two different classifiers were trained on the extracted features; the answers of the two classifiers were combined by voting to get the final decision. The use of two classifiers instead of one did not improve the results, and in some cases it was even worse. This is probably due to the fact that

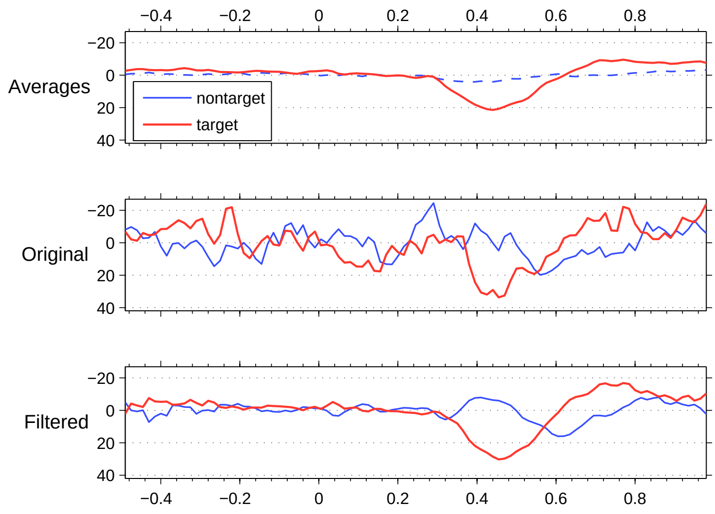


Figure 5.2: Signal filtered with the ARX method

the reference signal for the nontarget model is weaker, and it does not produce a meaningful response.

Figure 5.2 shows an example of the ARX processing on one channel (Pz). The top graph shows the averages for both target and nontarget signals. The nontarget average is not used and has been shown just for reference. In the middle, two particular epochs are shown before ARX processing, one target and one nontarget. ARX filtering has been applied to these signals, and the results are shown in the bottom graph. The responses in bottom graph are obtained by filtering the target average, and in fact they are larger where the target average is larger. In this case, the ARX filtering makes the difference between a target and a nontarget signal more evident.

5.2.2 Feature Extraction And Classification

For classification, a series of features are extracted from the filtered responses or directly from the ARX models. We used three kinds of features: ARX coefficients, impulse response, and characteristics of signals. Given a reference signal, ARX coefficients completely determine the response of the model (see also Equation (4.9)). For this reason, ARX coefficients should contain information about the nature (target/nontarget) of the epoch. Impulse response[2] contains the same information as ARX coefficients, considering that it exists a one-to-one mapping between coefficients and impulse responses[3]. While the impulse response is infinite, only the first samples have been used as features, because responses happen to go toward zero rather quickly; a number of samples equal to $2 \cdot (p + q + d)$ seemed to be sufficient.

The third kind of features are computed from the estimated ERPs, i.e., the ARX responses. They are (the time intervals reported are relative to the stimulus onset):

- Amplitude (with sign) of the maximum peak within the 200–600 ms window. If both a positive and a negative peaks exist in the window, the one with the greater absolute value is taken.
- Latency of the above peak.
- Mean of the amplitude in a 100–600 ms window. This is to assess that the peak of the first feature is not a short-lived phenomenon, but something similar to the typical P300 shape.
- Mean of the amplitude in a 100–600 ms window less the mean of the rest of the signal. Similarly to the previous, this feature is to distinguish signals that have a strong positive component only where a P300 is expected, and not in the whole epoch interval.
- $\mathrm{cov}[\mathrm{s}(\cdot), \mathrm{u}(\cdot)]/\mathrm{var}[\mathrm{u}(\cdot)]$, the ratio between the covariance of the response and the reference signal, and the variance of the reference. Here, the reference is the one used for the identification of the ARX model that has generated the response. This feature is the slope of the regression line when the response is plotted versus the reference.
- Number of peaks within the 100–600 ms window. This is a regularity information.

All the three kinds of features are extracted separately for each EEG channel. For classification of epochs, features from all the channels are concatenated together. Some nontarget epochs are discarded from the training set, so that the number of targets and nontargets is roughly the same, and a classifier is then trained on the features of the training

[2]Impulse response is the output of a system when the input is an impulse, i.e., for discrete-time models, a 1 followed by 0s.

[3]Actually, this is not true in some boundary cases, when a pole and zero of the transfer function coincide. This is a very unlikely situation for a model identified on data.

set. A P300 paradigm requires that nontarget epochs be many more than target ones, and this is a problem when training a classifier, as the classifier may become biased toward the most numerous (nontarget) class; for this reason, the training set has to be balanced.

We applied various classifiers to these features: Bayes networks, logistic, boosting of stumps, SVMs, neural networks (NNs), decision trees. SVMs and logistic classifier are described in Sections 4.2.2 and 4.2.1, respectively. Bayes networks [112] are graphical models representing sets of variables (features) and their probabilistic dependencies. Decision trees [113] are composed by nodes connected by branches, and at each node a branch is followed depending on the comparison of the value of a single feature with a threshold; at the last branching level, a class is assigned. Artificial neural networks [114] are classifiers inspired to biological neural networks; they are arranged in layers, where simple computational units can recognize even complex patterns by working together. Boosting of stumps is the application of AdaBoost [115], an algorithm that builds a classifier by combining many classifiers with lower performances; stumps are decision trees with a single branch, and AdaBoost combines them in a complex classifier.

Some classifiers, in particular SVMs and neural networks, have parameters: the coefficient C in Equation (4.24) and kernel parameters for SVMs, and the number of neurons for neural networks. We chose these parameters through a cross-validated optimization procedure based on the training set. The training set was divided in three or five folders (parts), and they were used each in turn as a test set for classifiers trained on the remaining two folders. Classifiers with different parameter combinations were tried, and the parameters with the best average performance on all the folders were selected.

Decision trees performed very poorly; their results could have been easily obtained by tossing a coin. Probably, there was some parameter to tune for better performance, but we preferred to concentrate on the other classifiers, which performed better.

5.2.3 Results

In a P300 speller application, the result is not just a classification of epochs as target or nontarget, but a letter, which is hopefully the target letter. All the classifiers used in our experiments return an answer with a confidence value attached, and this means that, instead of just a 0/1 answer, a classifier outputs a number between 0 and 1. With this piece of information available, a letter is selected as the most likely intersection between rows and columns. In other words, if $y_1, \ldots, y_n$ are the output

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	11/42 (26%)	24/42 (57%)	24/42 (57%)	24/42 (57%)
Logistic	37/42 (88%)	35/42 (83%)	35/42 (83%)	33/42 (79%)
Stump Boost	16/42 (38%)	22/42 (52%)	31/42 (74%)	30/42 (71%)
SVM	32/42 (76%)	28/42 (67%)	26/42 (62%)	32/42 (76%)
NN	34/42 (81%)	32/42 (76%)	27/42 (64%)	29/42 (69%)

Table 5.1: Results for the BCI Competition 2003 data set IIb: 5-fold cross-validation on the training set. The number of correct letters is shown.

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	8/31 (26%)	17/31 (55%)	19/31 (61%)	16/31 (52%)
Logistic	24/31 (77%)	18/31 (58%)	24/31 (77%)	14/31 (45%)
Stump Boost	8/31 (26%)	14/31 (45%)	25/31 (81%)	27/31 (87%)
SVM	19/31 (61%)	16/31 (52%)	18/31 (58%)	20/31 (65%)
NN	25/31 (81%)	26/31 (84%)	24/31 (77%)	20/31 (65%)

Table 5.2: Results for the BCI Competition 2003 data set IIb: performance on the competition test set. The number of correct letters is shown.

of a classifier for the n rows of the speller, the row $r = \arg\max_{r=1\ldots n} y_r$ is selected. The same happens for columns. If stimulations are repeated for each letter, values of y_r for the same row (or column) from different repetitions are added together before taking the maximum.

For the competition, the 10 channels lying on the midline of the skull are used (Fpz, Afz, Fz, Fcz, Cz, Cpz, Pz, Poz, Oz, Iz), and data is decimated with a factor of 3. Decimation reduces the memory footprint while retaining most information, as the resulting 80 Hz frequency means that all components present in the original signal up to almost 40 Hz are kept.

Tables 5.1 and 5.2 show the results obtained by applying ARX filtering to the BCI competition data. The first table contains the results on just the training test, cross-validated by a 5-fold cross-validation scheme. The second table shows the results obtained on the competition test set according to the competition rules. In both tables, each row represents a different classifier, while columns represent four feature sets: the three presented above (ARX coefficients, impulse response of the ARMA block, characteristics of the filtered signal $s(\cdot)$) plus the concatenation of all these three features, in the last column. The figures are the number of correct letters over the number of all letters (higher is better). Results in the first table are cross-validated, so they can be used as an estimate of the generalization capacity of the various classifier-feature combinations. If we pick the best combination, i.e., logistic classifier and ARX coefficients, the corresponding result for the test set is 77%. While this is not bad, the competition winners reached 100%. Another sign of troubles is the fact that many classifier-feature combinations with good results in the training set (mostly those involving the logistic classifier) have a dramatic drop of performance on the test set. Probably, the logistic classifier is overfitting; after all, the test set comes from a session different from the training set, while the cross-validation on the training set is done with data coming from the same sessions. This explanation is supported by the fact that the most dramatic drops happens when the number of features is larger (i.e., impulse response and the combination of all features). So, there is still much room for improvement, or maybe the ARX model is not so good an idea for this data set.

In the S. Camillo task, a subject may change the designated target direction at any moment, because during the recording the cursor could have been moved after each stimulus. In the data file there is only the information weather a stimulus is a target or not, and for non-targets there is no information about what was the target at that moment. So, there is no way to tell when a change of target happened just by looking at the saved data, and hence, it is not possible to compute something like the "number of correctly predicted arrows" in a fair way; instead, the figures for the performance of the classification system are related to the capacity of discrimination between epochs containing or not a P300. Because it is both simple to compute and simple to understand, *recall* for both targets (left columns) and nontargets (right columns) is shown in Tables 5.3 to 5.13. *Recall* means the fraction correctly classified for a given class, i.e., the number of correctly classified examples of a given class over the total number of examples for the class. Figures for other criteria (mutual information, Kappa statistic, precision) have also been computed, but they lead to the same conclusions as recall, and for this reason, they are not shown. Accuracy is not much useful in this case, as the data set contains many more examples of one class (nontarget). A high accuracy may in fact hide a poor performance on the classification of targets, which are underrepresented.

A few conclusions may be drawn by observing the tables with the results on the S. Camillo data set. In general, features related to characteristics of ARX responses (column *Shape*) are the best. Results in the last column, where all the features are used together, are often slightly lower; the combination of good features with worse ones seems

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	53% 47%	7% 93%	75% 79%	75% 79%
Logistic	79% 81%	76% 78%	83% 84%	74% 75%
Stump Boost	68% 40%	54% 50%	73% 87%	73% 86%
NN	59% 59%	68% 74%	79% 88%	80% 81%
SVM	63% 62%	71% 74%	83% 85%	82% 82%

Table 5.3: Results of ARX filtering for Subject SC1 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 523 (39.2%), nontargets: 812 (60.8%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	18% 86%	19% 85%	68% 69%	67% 68%
Logistic	67% 67%	61% 63%	73% 69%	60% 60%
Stump Boost	54% 55%	54% 51%	68% 73%	69% 70%
NN	47% 73%	29% 88%	67% 73%	68% 67%
SVM	66% 58%	63% 62%	71% 69%	69% 65%

Table 5.4: Results of ARX filtering for Subject SC2 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 312 (25.6%), nontargets: 905 (74.4%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	26% 81%	28% 78%	69% 69%	70% 67%
Logistic	66% 68%	63% 63%	70% 71%	67% 59%
Stump Boost	52% 54%	51% 57%	70% 66%	64% 68%
NN	49% 63%	52% 73%	74% 65%	70% 64%
SVM	55% 61%	64% 66%	72% 71%	70% 69%

Table 5.5: Results of ARX filtering for Subject SC3 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 222 (25.1%), nontargets: 663 (74.9%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	13% 88%	37% 69%	66% 64%	70% 63%
Logistic	72% 75%	61% 69%	74% 71%	64% 63%
Stump Boost	54% 54%	56% 51%	63% 71%	64% 69%
NN	54% 65%	77% 62%	72% 74%	73% 67%
SVM	67% 64%	71% 70%	73% 72%	75% 71%

Table 5.6: Results of ARX filtering for Subject SC4 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 261 (25.2%), nontargets: 774 (74.8%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	11% 88%	12% 89%	49% 54%	39% 66%
Logistic	70% 65%	61% 64%	61% 68%	62% 62%
Stump Boost	62% 49%	53% 50%	54% 70%	61% 52%
NN	36% 72%	41% 76%	62% 63%	66% 58%
SVM	62% 56%	59% 66%	61% 70%	66% 67%

Table 5.7: Results of ARX filtering for Subject SC5 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 298 (27.6%), nontargets: 780 (72.4%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	45% 62%	65% 46%	55% 72%	58% 69%
Logistic	65% 66%	65% 63%	67% 73%	65% 59%
Stump Boost	65% 53%	53% 59%	61% 70%	64% 64%
NN	57% 57%	72% 58%	72% 67%	64% 65%
SVM	62% 61%	67% 68%	67% 70%	68% 67%

Table 5.8: Results of ARX filtering for Subject SC6 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 294 (24.9%), nontargets: 888 (75.1%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	34% 74%	52% 63%	73% 77%	74% 76%
Logistic	74% 73%	66% 70%	77% 72%	61% 67%
Stump Boost	56% 65%	56% 63%	74% 79%	73% 79%
NN	75% 66%	73% 74%	77% 77%	73% 77%
SVM	67% 72%	72% 76%	77% 76%	77% 75%

Table 5.9: Results of ARX filtering for Subject SC7 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 231 (25.7%), nontargets: 669 (74.3%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	24% 73%	65% 50%	60% 74%	61% 75%
Logistic	71% 75%	59% 68%	69% 76%	56% 65%
Stump Boost	59% 59%	57% 58%	58% 79%	63% 72%
NN	38% 79%	61% 75%	67% 76%	69% 73%
SVM	59% 69%	65% 74%	74% 75%	69% 74%

Table 5.10: Results of ARX filtering for Subject SC8 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 229 (27.5%), nontargets: 604 (72.5%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	30% 75%	46% 58%	66% 71%	65% 71%
Logistic	72% 72%	61% 63%	69% 71%	54% 61%
Stump Boost	62% 54%	52% 57%	66% 75%	62% 72%
NN	44% 72%	64% 69%	67% 78%	62% 81%
SVM	59% 67%	65% 72%	76% 72%	72% 73%

Table 5.11: Results of ARX filtering for Subject SC9 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 223 (27.8%), nontargets: 580 (72.2%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	21% 79%	51% 55%	59% 62%	64% 58%
Logistic	69% 63%	62% 61%	68% 66%	63% 59%
Stump Boost	63% 46%	54% 58%	62% 68%	70% 57%
NN	72% 50%	78% 53%	75% 59%	81% 51%
SVM	65% 62%	72% 63%	71% 63%	66% 67%

Table 5.12: Results of ARX filtering for Subject SC10 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 232 (25.4%), nontargets: 682 (74.6%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	42% 62%	54% 47%	62% 72%	62% 64%
Logistic	65% 62%	61% 53%	70% 70%	60% 48%
Stump Boost	59% 55%	50% 52%	71% 65%	70% 59%
NN	69% 50%	73% 58%	74% 60%	77% 59%
SVM	59% 68%	58% 72%	70% 65%	66% 70%

Table 5.13: Results of ARX filtering for Subject SC11 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 204 (43.6%), nontargets: 264 (56.4%).

to confound the classifiers. The best classifiers for this data set seem to be logistic, neural networks, and SVMs. The logistic classifier is the one that suffers more when the three kinds of features are combined. If we discard the worst combinations of classifiers and feature sets, recalls range from 60% to 80%. Because this is a 2-class classification problem, 60% is slightly better than random, and probably insufficient for a real-life BCI, while 80% is definitely useful for building a P300-based BCI, although not enough for single-sweep detection. The last three subjects (SC9, SC10, SC11) are paralyzed; their results are worse than those for healthy people, but the difference is not strong, and there is much variability among subjects.

After having seen Figure 5.2 one might wonder why results do not show a 100% correct classification. A possible explanation may lie in Figure 5.3. Here, two more epochs from the same subject (which happens to be Subject SC1) are shown, one target and one nontarget, and also the corresponding ARX responses. While the unprocessed signals are rather different, and the P300 in the target epoch is evident, with a strong positive peak around 380 ms, the filtered epochs are very similar and almost all the difference has been washed away by

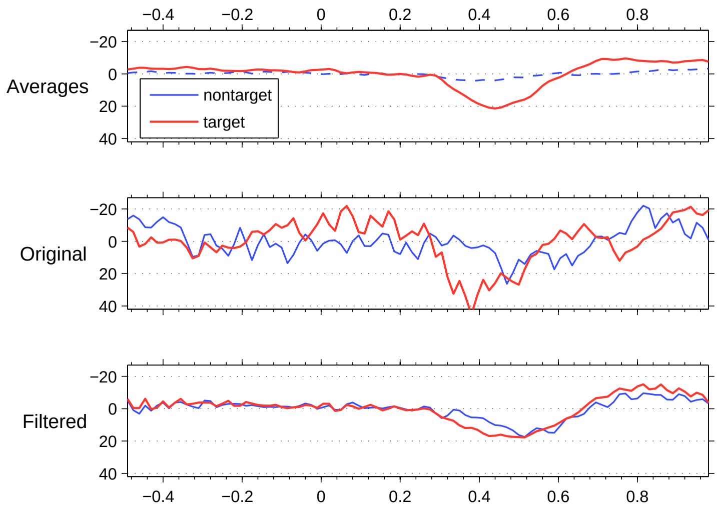


Figure 5.3: Signals filtered with the ARX method

ARX filtering. The apparent cause is the dissimilarity between the reference signal (the target average) and the target epoch in question; the maximum of the reference lies around 440 ms, while in the target epoch is earlier. It almost seems as this epoch has been anticipated by half a tenth of a second.

The fact that a single ERP may differ considerably from the grand average is well known from the literature. And the rationale of ARX filtering bears on it actually, as ARX filtering should be able to adapt the grand average to the changing of the ERP from case to case. Obviously, extreme cases cannot be always handled correctly. By the way, the case shown in Figure 5.3 is really an exception, because our method scored correctly 80% of the same subject's epochs.

5.2.4 EOG Removal

In some data sets also an EOG channel has been recorded. This is not used for P300 detection, but it contains information about interferences coming from the ocular region and could be used for artifact removal. An extension of the ARX model that makes use of the EOG has been developed, and it is shown in Figure 5.4. Here, the EEG signal $y(\cdot)$ is considered the superposition of three components

$$y(t) = s(t) + r(t) + n(t) \tag{5.2}$$

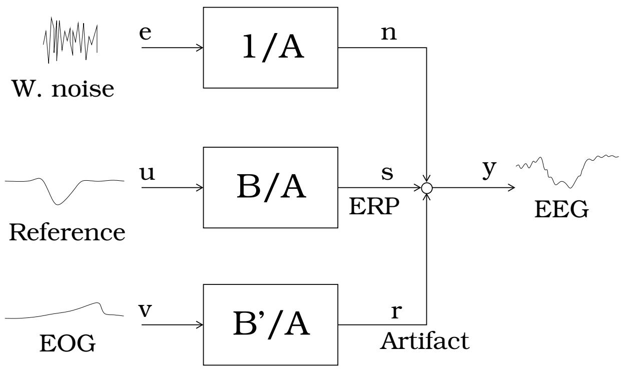


Figure 5.4: Variant

where $r(\cdot)$ represent the contribution of ocular artifacts, $s(\cdot)$ still represents the ERP component, and $n(\cdot)$ the unpredictable noisy component. This contribution is obtained by filtering the recorded EOG $v(\cdot)$ with a deterministic ARMA system:

$$r(t) = \sum_{h=d'}^{q'+d'-1} b'_h v(t-h) - \sum_{j=1}^{p} a_j y(t-j)\,. \tag{5.3}$$

The resulting formula for this extended ARX system is:

$$y(t) = \sum_{k=d}^{q+d-1} b_k u(t-k) + \sum_{h=d'}^{q'+d'-1} b'_h v(t-h) - \sum_{j=1}^{p} a_j y(t-j) + e(t)\,. \tag{5.4}$$

This extended ARX model is used exactly as the base version, with the only difference that in identifying the parameters of the model also the EOG is used. The objective is still to separate the ERP components $s(\cdot)$ from the noise, and $s(\cdot)$ is the result of the application of the $B(z)/A(z)$ filter to the reference signal $u(\cdot)$. An ARX family is characterized by five orders (p, q, q', d, d'), which can still be chosen by minimizing Akaike's score.

Tables 5.14 to 5.24 show the results of the extension of the ARX method that takes into account ocular interference. They are practically identical to the ones where the EOG channel is not used. The reason for this somewhat surprising result may be found in the fact that P300 is generally stronger in the parietal region, where the influence of EOG

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	42% 58%	8% 93%	71% 81%	71% 81%
Logistic	76% 79%	74% 75%	81% 82%	72% 73%
Stump Boost	62% 40%	61% 40%	68% 88%	69% 84%
NN	57% 58%	69% 69%	80% 82%	73% 83%
SVM	60% 62%	70% 73%	79% 85%	80% 81%

Table 5.14: Results of extended ARX filtering for Subject SC1 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 523 (39.2%), nontargets: 812 (60.8%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	4% 95%	4% 95%	66% 70%	66% 70%
Logistic	65% 66%	65% 64%	70% 70%	64% 58%
Stump Boost	48% 60%	58% 53%	65% 76%	66% 70%
NN	66% 52%	58% 71%	67% 73%	65% 74%
SVM	66% 63%	65% 69%	72% 70%	69% 69%

Table 5.15: Results of extended ARX filtering for Subject SC2 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 312 (25.6%), nontargets: 905 (74.4%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	40% 70%	39% 70%	63% 67%	64% 68%
Logistic	67% 65%	56% 61%	68% 71%	60% 62%
Stump Boost	53% 60%	52% 57%	66% 70%	69% 67%
NN	39% 72%	62% 60%	68% 69%	69% 68%
SVM	52% 62%	64% 64%	72% 73%	71% 68%

Table 5.16: Results of extended ARX filtering for Subject SC3 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 222 (25.1%), nontargets: 663 (74.9%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	8% 93%	33% 71%	67% 65%	67% 64%
Logistic	70% 75%	66% 68%	72% 71%	64% 61%
Stump Boost	46% 61%	56% 55%	61% 72%	67% 63%
NN	53% 67%	65% 72%	75% 65%	73% 68%
SVM	68% 66%	70% 71%	72% 74%	76% 70%

Table 5.17: Results of extended ARX filtering for Subject SC4 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 261 (25.2%), nontargets: 774 (74.8%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	35% 65%	0% 100%	26% 83%	61% 48%
Logistic	69% 64%	59% 62%	70% 63%	62% 61%
Stump Boost	54% 53%	55% 55%	56% 66%	65% 57%
NN	33% 74%	71% 46%	63% 70%	64% 66%
SVM	62% 57%	59% 67%	71% 65%	69% 64%

Table 5.18: Results of extended ARX filtering for Subject SC5 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 298 (27.6%), nontargets: 780 (72.4%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	17% 85%	69% 40%	59% 62%	62% 58%
Logistic	64% 66%	62% 62%	65% 72%	62% 61%
Stump Boost	63% 42%	52% 56%	53% 79%	61% 64%
NN	55% 58%	73% 58%	71% 65%	69% 61%
SVM	63% 60%	69% 66%	67% 73%	68% 65%

Table 5.19: Results of extended ARX filtering for Subject SC6 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 294 (24.9%), nontargets: 888 (75.1%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	44% 70%	58% 62%	71% 75%	69% 75%
Logistic	71% 71%	66% 67%	74% 73%	66% 63%
Stump Boost	56% 65%	63% 58%	71% 79%	75% 72%
NN	74% 66%	69% 72%	74% 77%	72% 75%
SVM	65% 74%	72% 73%	76% 77%	76% 76%

Table 5.20: Results of extended ARX filtering for Subject SC7 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 231 (25.7%), nontargets: 669 (74.3%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	20% 78%	72% 29%	58% 75%	57% 76%
Logistic	59% 72%	59% 64%	66% 75%	62% 60%
Stump Boost	48% 60%	57% 56%	64% 74%	60% 76%
NN	61% 55%	54% 75%	62% 75%	64% 73%
SVM	57% 62%	61% 73%	72% 70%	66% 74%

Table 5.21: Results of extended ARX filtering for Subject SC8 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 229 (27.5%), nontargets: 604 (72.5%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	0% 100%	14% 85%	59% 74%	59% 74%
Logistic	74% 69%	63% 56%	71% 71%	57% 63%
Stump Boost	57% 52%	39% 62%	67% 69%	61% 73%
NN	51% 63%	66% 67%	64% 77%	73% 67%
SVM	58% 63%	62% 72%	70% 76%	71% 73%

Table 5.22: Results of extended ARX filtering for Subject SC9 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 223 (27.8%), nontargets: 580 (72.2%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	41% 67%	55% 56%	63% 61%	70% 61%
Logistic	69% 65%	64% 60%	67% 69%	61% 57%
Stump Boost	51% 55%	51% 59%	70% 63%	69% 61%
NN	70% 51%	75% 55%	63% 70%	72% 58%
SVM	61% 63%	63% 67%	68% 70%	63% 71%

Table 5.23: Results of extended ARX filtering for Subject SC10 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 232 (25.4%), nontargets: 682 (74.6%).

Classifier	**Features**			
	ARX Coeff.	**Imp. Resp.**	**Shape**	**All**
Bayes Net	13% 84%	6% 96%	56% 71%	61% 69%
Logistic	66% 64%	59% 61%	65% 73%	63% 61%
Stump Boost	48% 66%	46% 66%	63% 74%	65% 67%
NN	44% 75%	55% 72%	57% 75%	72% 64%
SVM	55% 71%	59% 71%	65% 69%	67% 71%

Table 5.24: Results of extended ARX filtering for Subject SC11 in the S. Camillo data set: 5-fold cross-validated recall for targets and nontargets. Targets: 204 (43.6%), nontargets: 264 (56.4%).

is attenuated by the distance. On the contrary, the Fz channel is much affected by the ocular region, but it seems that either this channel has little importance for classification, or even the “basic” ARX filtering is robust enough to reject ocular artifacts. It could be interesting to confirm either hypothesis with an analysis of the ARX responses, but this has been left for the future, as we have concentrated on different aspects.

5.3 SVMs: A Direct Approach

The results shown for the detection of P300 based on ARX filtering range from “almost acceptable” to “good”, depending on the subjects. In general, they are far better than random, and the system could be used for a live P300 BCI (though we have not tested it online yet). However, the results on the BCI competition data set are much worse than the winning entries to the competition; a fair estimation of our result on the test set is 77% of correct letters, as explained above, which is much worse of the 100% obtained from five different participants to the competition.

One of the advantages of using data coming from a competition is that it is easy to make comparisons with other people's results. In our case, the results told us that our method was not making the best use of the information available in the EEG data, but we needed a way to understand where the problem lied, in the ARX filtering or in the feature extraction and classification.

To this aim, we studied and replicated the method used by the winners of the competition, Kaper and colleagues from the University of Bielefeld, Germany [58]. Their method relies more on the power of the classifier employed, an SVM, than on signal processing, and the feature extraction is done implicitly by the SVM; in other words, this is a blind algorithm, which does not rely on specific knowledge about the P300. The blindness of this method made us confident that its application to the classification to the responses of ARX filtering could provide an answer to our question about the relative weaknesses of the steps of our ARX method.

In Kaper's method, an epoch begins at the time of stimulus, and ends 600 ms after. Data from channels Fz, Cz, Pz, Oz, C3, C4, P3, P4, Po7, and Po8, are used. Epochs are bandpass filtered between 0.5 and 30 Hz, and then normalized to the $[-1, +1]$ interval. The training set is balanced by taking only two nontarget examples from each repetition, which already contains exactly two target examples, and an SVM is trained directly on the balanced training set. In this case, the normalized samples of the EEG signals are used as features for the classification.

In order to find out the unknown letter in the test set, several repetitions are combined together. Starting from the first repetition, a score is assigned to each row and column. Scores from different repetitions are added together, and the row and the column with the maximum total score after the last repetition is selected and the corresponding letter is chosen. This procedure is repeated for each letter in the test set. It is possible to halt the procedure just after a few repetitions, without making use of all the 15 repetitions in the data. The authors correctly classified all the competition test set with only 5 repetitions per letter.

The score mentioned above is the SVM discriminant function $f^*(\cdot)$ in equation (4.29):

$$f^*(\boldsymbol{x}) = \sum_i \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b \qquad (5.5)$$

No. of Repet.	Direct SVM	ARX Response + SVM	ARX Residual + SVM
5	100% (31/31)	9.0% (2.8/31)	49.7% (14.6/31)
15	100% (31/31)	36.8% (11.4/31)	73.2% (25.8/31)

Table 5.25: Results (fraction of correct letters) of SVMs applied to ARX responses for the competition test set

In this case, the kernel function $K(\cdot)$ is Gaussian. The parameter σ of the kernel and the penalization coefficient C in the objective function of the SVM are found with a cross-validation scheme on the training set.

5.3.1 Mixing ARX With SVMs

We tried to see what happened if we trained an SVM, according to the procedure just related, on the responses returned by the ARX filtering. Results were surprising: Table 5.25 shows the number of correct letters after 5 and 15 repetitions with three methods applied on the test set of the BCI Competition 2003 data.

The first method (second column) is the one described in the last section, and predicts 100% correct letters. The next column is for an SVM working on responses extracted by the ARX filtering method introduced before; they are rather bad. But the last column is the more interesting one. It shows the results for an SVM applied to the *residuals* of the ARX filtering, i.e., the outputs of AR parts.

The results in Table 5.25 refer to ARX systems with orders (p, q, d) equal to $(8, 2, 0)$. There are decimals for the numbers of wrong letters as they are the means of results obtained from different (random) choices of nontarget epochs. So, it seems that the SVM works better with what should have been just noise than with what should have been an informative signal. This suggests that the ERP estimated by the ARX filtering contains only a part of the useful information, but not all. These results lead us towards a different approach, as explained in the next section.

5.4 Genetic Algorithm

The approach of Kaper and colleagues illustrates an interesting point: It is possible to classify epochs in an effective way without the usual chain of preprocessing, processing, feature extraction, and classification. This is not a completely new idea, as others have applied a similar concepts to

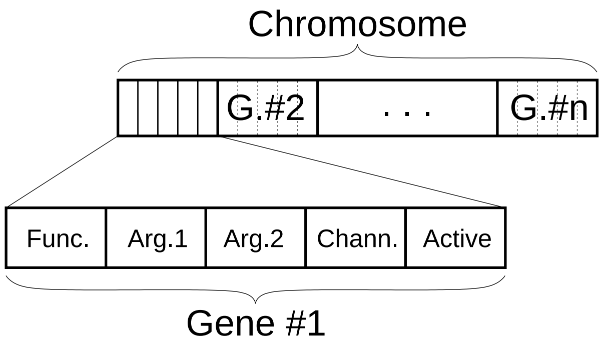


Figure 5.5: Structure of a chromosome

signal classification, even in the BCI field; see for example [116]. Inspired by this, we developed a "shallow" approach which is based on automatic feature extraction. In this approach, a genetic algorithm operates on features to be used for the classification of P300 epochs. Features are encoded in variable-length chromosomes, where each gene encodes one feature. The fitness of an individual is given by the performance of a classifier trained on the encoded features.

GAs have been used already in the BCI field. In [117], the best combination between different features and different classifiers is sought for a motor-imagery task. In [118], a GA selects P300 features extracted by means of a wavelet, and finds the best linear classifier that operates on them.

In our approach, the GA is used only to select the best features to be fed to a classifier, while the classifier itself is trained in the classical way. Each individual in the population has one chromosome, whose logical structure is shown in Figure 5.5. A chromosome contains a variable number of genes, with an identical structure, and each gene is formed by five elements. The first three elements define a feature: The first one is an integer number designating one feature extractor out of a predetermined set, while the two following elements encode two real-valued parameters for such an extractor. Feature extractors are functions with three arguments: a signal from which a feature is extracted, and the two parameters encoded in genes; these parameters are within the range $[0, +1)$, and their actual meaning varies from extractor to extractor. The fourth element of a gene is an integer number, which identifies the EEG channel the feature encoded in the gene is to be extracted from. The last element of a gene is a Boolean flag that determines whether the gene is active or inactive. Inactive genes are not used to compute the fitness of a chromosome. Their role

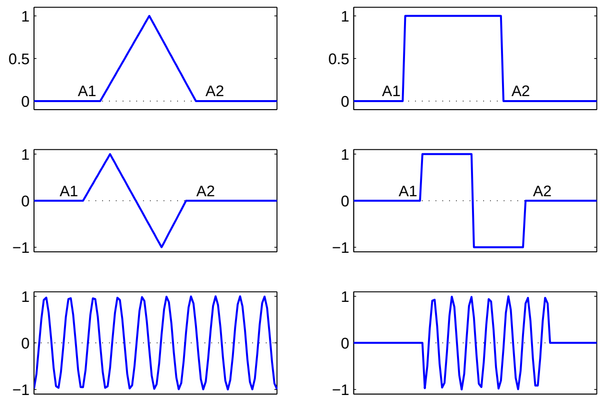


Figure 5.6: The weight functions encoded in genes

is of a genetic reserve, as they can be turned on in a later generation by mutation. The position of a gene within a chromosome is not significant.

We used up to six different feature extractors, which share a very simple scheme: The input signal is multiplied by a weight function, and the result is integrated over time. In other words, feature extractors compute the cross-correlation between the input and a weight function. If we call $s(\cdot)$ the EEG signal from the channel the feature is to be extracted from, a feature extractor constructs a weight function $u(\cdot)$ from the parameters specified by the gene elements and then computes the resulting feature x with the formula:

$$x = \sum_{t=1}^{T} u(t)s(t)\,. \tag{5.6}$$

The six weight functions used by the six feature extractors are shown in Figure 5.6. The feature that uses the weights shown in the top-right box is proportional to the average of the input signal over an interval; the extremes of the interval are determined by the two parameters (A1 and A2 in the figure) encoded in genes. The weights in the top-left box produce a similar effect, but the samples at the center of the interval weight more. The functions in the middle row compute the differences between two adjacent intervals; again the extremes are encoded in genes. The functions in the bottom row compute the cross-correlation with a sine wave; genes encode frequency and phase of the sines. The interval where the bottom-right weight function is not zero is fixed, and it goes from 0 to 600 ms after the stimulus, i.e., it is centered around the P300.

These last two functions permit to do a sort of frequency analysis of the signal.

5.4.1 Fitness and Selection

The fitness of a chromosome is determined by measuring the performance of a logistic classifier on the features it encodes. To have a fair estimate of the performance, a 4-fold cross-validation scheme on the training set is used, and the mean performance on the 4 folds is used as the fitness. The actual criterion used to evaluate the"performance" depends on the kind of data. The fitness function of our genetic algorithm can be easily changed without modifying anything else in the algorithm. This permits to adapt the fitness computation to different BCI tasks.

For data recorded with a P300 speller, the number of correctly predicted letters is used, with a little bonus for letters that can be correctly predicted with less than the maximum number of repetitions, i.e., the number of times the whole grid is flashed for each letter. Let us call l the number of correctly predicted letters out of a total of n, N the number of repetitions in the data set, and r_i, $i = 1 \dots n$, the number of repetitions needed for the prediction of the letter i. The fitness f is then given by

$$f = \frac{1}{n}\left(l + \frac{1}{l}\sum_{i \in I} \frac{N - r_i}{N + 1}\right), \tag{5.7}$$

where I is the set of correctly predicted letters. The second term in the parentheses computes an index, averaged over the l correct letters, that grows with the decreasing of r_i; this index is always strictly less than 1, and therefore it contributes to the fitness less than a single correctly predicted letter. In this way, a higher number of correct letters is always preferred to a lower number of repetitions needed for correct prediction. Repetitions are taken in their occurring order, and r_i is computed in way such that if a letter is correctly predicted by using the first r_i repetitions, then it must be correctly predicted also by using the first $r_i + 1, \dots, N$ repetitions. In other words, if a letter were predicted correctly after 3 repetitions, wrongly after 4, and again correctly when using 5 repetitions or more, then r_i would be 5, and not 3.

For the S. Camillo data set, we chose a different fitness function, as explained above (Section 5.2.3) single epochs should be treated separately due to the characteristics of the protocol. The fitness f of a classifier is obtained by combining precision p_T and recall r_T for targets according to this formula:

$$f = \frac{2}{3}p_{\mathrm{T}} + \frac{1}{3}r_{\mathrm{T}} \tag{5.8}$$

The definitions of precision and recall in terms of *true positives* (TP), *false positives* (FP), and *false negatives* (FN) are:

$$p_{\mathrm{T}} = \frac{TP}{TP + FP} \qquad\qquad r_{\mathrm{T}} = \frac{TP}{TP + FN} \tag{5.9}$$

The rationale behind this formula is that, while it is good to recognize a large portion of P300s (recall), that should not result in too many false positives, which are taken into account in precision. Precision has been weighted more because — at least in the task under consideration — a false positive does more harm than a false negative.

In a genetic algorithm, fitness is used to select the most promising individuals for the next generation. The selection mechanism employed is *tournament selection* with *elitism*, a standard setup in genetic algorithms, with no particular adaptation. Briefly, in tournament selection each individual of the new population is selected by setting up a tournament: A fixed number k of individuals are chosen at random from the old population, and the one with the highest fitness is declared the winner and will get in the next generation. Values for k are usually small, as large values would favorite the fittest individuals too much, causing a loss of diversity in the population; in our work, $k = 4$. Elitism is the practice of keeping the fittest individual or individuals in the new generation, even when selection discarded them (e.g., because they never participated to any tournament), or mutation and selection modified them.

5.4.2 Genetic Operators

After selection, the selected population undergoes crossover and mutation. These two operators have been slightly modified in order to adapt them to the non-standard chromosome structure we employed. Figure 5.7 shows how crossover works. Crossover is applied to *pairs* of chromosomes in the selected population (chosen at random) with a probability of 0.7; both chromosomes are split in two sections at a gene boundary in a random way, and then the four sections are recombined. Because the order of genes in a chromosome is not important, one section from one chromosome can be coupled with either section from the other one, and so there are two different way of doing crossover. Which way to use is randomly chosen each time, and it is important to use

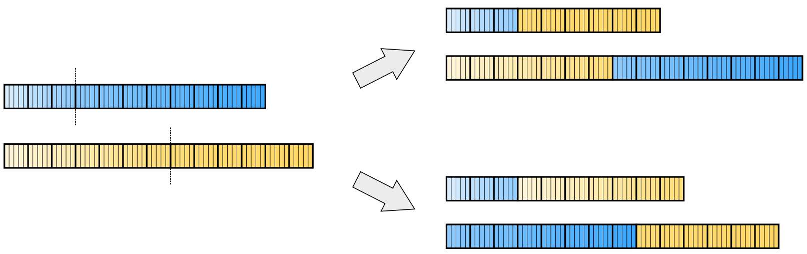


Figure 5.7: Crossover operator. The two possible ways of applying it are shown.

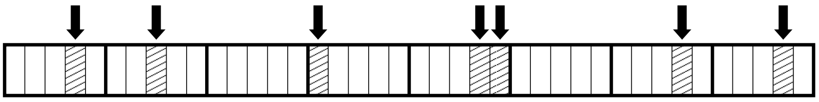


Figure 5.8: Mutation operator. Gene elements selected for mutation are marked.

both ways, as this choice increases the mixing of the genetic material. Crossover may be applied to individuals with a common ancestor, and so they may share some genes. In this case, it is very likely that at least one of the new chromosomes contains duplicated genes, and many duplicates accumulate with time. These duplicates are ignored for fitness evaluation.

Mutation (see Figure 5.8) operates on gene elements; for each element in each gene, a random choice is taken whether to mutate it, independently from each other, but with the same (small) probability, which is 0.005 in this algorithm. Elements are modified differently accordingly to their type. For a discrete element (extractor, channel, and active flag), mutation modifies it by choosing one of the other admissible values for that kind of elements, at random. For a continuous element (the two extractor parameters), a perturbation is added according to a Gaussian distribution; if the result lies outside the admissible interval $[0, 1)$, it is wrapped around, e.g., a value of 0.95 which is perturbed by 0.07 does not result in a new value of 1.02, which is not legal, but it is wrapped to 0.02.

The use of normalized extractor parameters is useful because the way mutation works. When mutation is applied to the gene element that encodes the feature extractor, the parameters are always legal also for the resulting new feature extractor; moreover, in some cases the old and the new weight functions are similar, and this helps the GA.

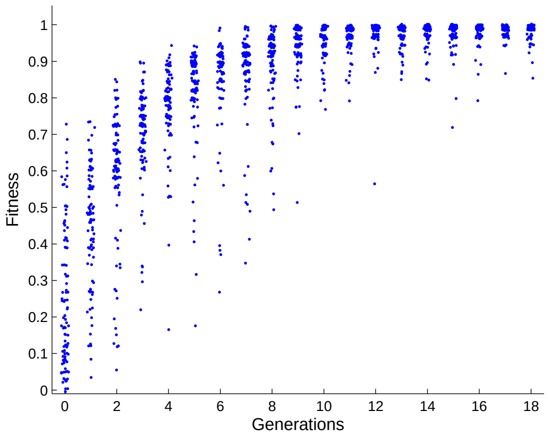


Figure 5.9: Evolution of fitness over time (generations) for the whole population in a GA run. Noise has been added to the point positions so as to make visible also overlapping points.

5.4.3 Population Size and Stop Criterion

The size of the population is constant throughout a GA run; for our experiments we used populations ranging from 70 to 120 individuals. Apart from size, the initial population is completely random; the length, i.e., the number of genes, for each chromosome is extracted from a geometric distribution with mean 20. The actual values for gene elements are taken from uniform distributions over the whole range of legal values for each elements.

The last component to complete the GA description is the stop criterion. We relied only on the number of generations, after some initial experiments where we noticed that in all runs no improvements could be seen in both the fitness of the best individual and the mean fitness of the population after 10–15 generations. Figure 5.9 shows how the fitness of a population evolves in a typical GA run; it is evident that the maximum fitness reaches a plateau after only 7–8 generations, and population fitness tends to stabilize around the 12th generation. This kind of behavior has been observed for all runs, with numbers varying little; for this reason, we decided to stop GA after 15 generations, or a couple of generations before for the most time-consuming runs. In any case, a check on the fitness growth is made after each run, so as to be sure that evolution has actually stopped: If the maximum and

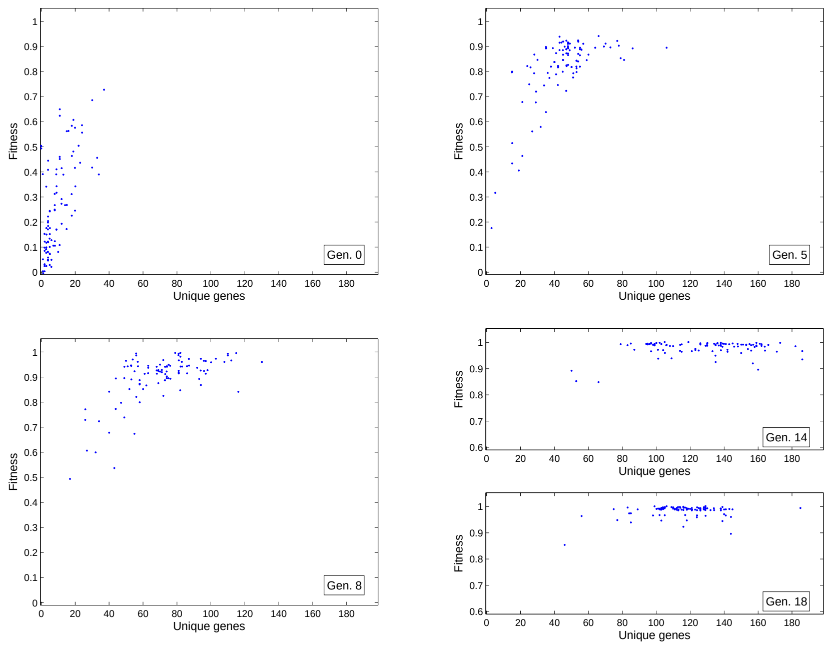


Figure 5.10: Evolution of chromosome length over time. These graphs refer to the same run of Figure 5.9; the generation is shown inside each graph.

mean fitness has been constant for the last 3–4 generations, evolution is considered finished.

5.4.4 Feature Set Validation

After the end of each GA run, the performance of the individuals with a high fitness is validated on a test set, never used before by the GA. This validation is done on the individuals with a fitness at least 99% of the fitness of the best individual in the last generation. Evaluating more than one individual and not just the best one results in a more robust assessment of the effectiveness of the method.

For each individual, the features encoded by its chromosome are extracted from all the training data (i.e., the data used for fitness evaluation), and a logistic classifier is trained on them. The same features are extracted from the test set, and the classifier is evaluated on them. The classifier can also be used online, together with the feature extractors it was trained on.

GA can be very time consuming. The length of chromosomes grows with generations (see Figure 5.10 for an example), and this makes training the logistic classifier more and more expensive. With very large data sets (more than 10000 epochs) the evaluation of a population of about 100 individuals could take even more than an hour on a modern 32-bit processor running at 2 GHz. Yet, the feature extractors and the trained logistic classifier are very fast to apply (especially with the trick shown in Equation (5.20) in the next section), and they can be used online in real-time.

5.4.5 Results

We applied the genetic algorithm first on the BCI Competition 2003 data set IIb, to compare it with other classification methods. In the procedure we used for this data set, data is first decimated by a factor of 3, so as to reduce memory usage, segmented in epochs, and detrended. We used epochs 1.5 s long initially, the same used for the ARX method; in later experiments we shortened them to 1 s, from 200 ms before the stimulus to 800 ms after it, to speed up the computation. Of the 64 channels, only 10 have been used, in three different combinations: those lying on the midline of the skull (Fpz, Afz, Fz, Fcz, Cz, Cpz, Pz, Poz, Oz, Iz), those used by Kaper and colleagues [58] (Fz, Cz, Pz, Oz, C3, C4, P3, P4, Po7, Po8), and other 10 channels lying on or near the midline (Fz, Fc3, Fc4, Cz, Cp3, Cp4, Pz, Po3, Po4, Oz). We also normalized EEG data in some classification experiments.

Table 5.26 shows the results of some of the most significant of our classification experiments; other experiments we made with different parameters do not provide further insight. The performance on the test set is given, as the ratio of the number of correct letters over the total number of letters. A single GA run usually returns more than one set of features, because many different chromosomes reaches the top fitness. All these chromosomes are closely related, due to the mixing of the genetic material. For this reason, it is not possible to summarize their performance in one number, as by averaging, for example. Therefore, Table 5.26 shows the entire range of values obtained for all chromosomes with a fitness above 99% of the top fitness.

From the table, it is apparent that the single most important parameter is the channel selection. When using the channels from Kaper, the GA consistently returns only chromosomes that score 100% correctly on the test data. The other parameters affect little the test performance, although they may influence the speed of convergence. When the number of feature extractors grows, the GA takes more

Norm.	Channels	Epoch	Feat. Funcs	Runs	Test perf.
-	z	long	f3	1	29–30/31
1	z	long	f3	1	28–30/31
4	z	long	f3	1	30–31/31
3	z	long	f3	1	28–30/31
3	z	long	f5	2	28–30/31
3	k	long	f5	1	31/31
3	k	long	f3	1	31/31
3	k	short	f2	4	31/31
-	k	short	f2	7	31/31
-	v	short	f2	2	30–31/31
3	v	short	f2	2	26–31/31

Table 5.26: Results of the genetic algorithm on the competition data set.

Legend

Norm. shows the type of normalization: '-' = no normalization; '1' = the same factor is used for all channels of all epochs; '3' = each channel of each epoch is normalized independently of the others; '4' = within each epoch, the same factor is used for all channels, but epochs are normalized independently of each other.

Channels shows which channel combination has been used: 'z' = Fpz, Afz, Fz, Fcz, Cz, Cpz, Pz, Poz, Oz, Iz; 'k' = Fz, Cz, Pz, Oz, C3, C4, P3, P4, Po7, Po8; 'v' = Fz, Fc3, Fc4, Cz, Cp3, Cp4, Pz, Po3, Po4, Oz.

Epoch is 'long' for the $(-0.5, +1.0)$ interval, and 'short' for $(-0.2, +0.8)$.

Feat. Funcs indicates the subset of the functions of Figure 5.6 used: 'f2' = triangle and square wave; 'f3' = triangle, square, and sine wave; 'f5' = all the functions except for the short sine wave.

Runs indicates the number of times the GA has run.

Test perf. is the number of correct letters in the test set.

generations before converging. The reason is that the more the feature extractors, the bigger the search space is. Using a shorter epoch length speeds up the computation of the fitness function and reduces the memory usage.

A typical problem of GAs is overfitting, as they are optimization algorithms. However, we have not observed overfitting in any of our classification experiments, probably because the cross-validation used in the fitness evaluation combined with the noisiness of the EEG data help in finding features with a good generalization capacity.

The number of the features found by the GA ranges from 100 to 150, more or less, as can be deduced from Figure 5.10, where only the number of unique and active genes is shown. In general, the chromosomes contain a number of genes about three times as bigger, because some genes are inactive, and duplicates crop up because of recombination.

At this point, one may expect to find a list of the features found by the GA, or a graph where they are nicely shown in a 2-dimensional map highlighting the most important time interval for every channel. Well, it turns out that we can do better than that. The GA deals not only with features, it optimizes the features for the classifier used in the fitness function as well. So, features are strongly connected with the classifier, and with some algebra it is possible to explicit this connection and understand the real meaning of what the GA finds.

Let us call $s(\cdot)$ an EEG signal from a single channel of an epoch, and consider only the features extracted by the GA for this channel. For feature extractors like the one described above, a feature is obtained by computing the cross-correlation of the signal with a weight function $u_j(\cdot)$:

$$x_j = \sum_{t=1}^{T} u_j(t)s(t)\,, \tag{5.10}$$

where j means that x_j is the feature encoded by the j-th gene, and T is the number of time samples per epoch. Please notice that $u_j(\cdot)$ is not a generic function, but the precise function encoded by all the parameters contained in a gene. A trained logistic classifier estimates the probability for the signal $s(\cdot)$ to belong to a class according to Equation (4.14):

$$\mathrm{P}\big(y = +1 \,\big|\, \boldsymbol{x}\big) = \frac{1}{1 + \exp(w_0 + \sum_{j=1}^{n} w_j x_j)}\,, \tag{5.11}$$

where x_j are the features given by Equation (5.10). Equation (5.11) gives the probability for the *target* class, which is enough because the probability for the other class can be computed by difference: $\mathrm{P}\big(y = -1 \,\big|\, \boldsymbol{x}\big) = 1 - \mathrm{P}\big(y = +1 \,\big|\, \boldsymbol{x}\big)$.

It is possible to rewrite the argument of the exponential in Equation (5.11) by substituting the (5.10) in it:

$$\begin{aligned} w_0 + \sum_{j=1}^{n} w_j x_j &= w_0 + \sum_{j=1}^{n} w_j \left(\sum_{t=1}^{T} u_j(t) s(t) \right) \\ &= w_0 + \sum_{t=1}^{T} \left(\sum_{j=1}^{n} w_j u_j(t) \right) s(t) \,. \end{aligned} \tag{5.12}$$

The term

$$v(t) = \sum_{j=1}^{n} w_j u_j(t) \,, \tag{5.13}$$

with $t = 1 \dots T$, depends only on the feature set (through $\boldsymbol{u}(\cdot)$) and on the classifier (through $\boldsymbol{w}$), and therefore it is the same for all epochs. Equation (5.11) becomes

$$\mathrm{P}\big(y = +1 \,\big|\, s(\cdot)\big) = \frac{1}{1 + \exp(w_0 + \sum_{t=1}^{T} v(t) s(t))} \,, \tag{5.14}$$

or, by considering $v(\cdot)$ and $s(\cdot)$ as vectors, whose components are the time sample, and using the dot product notation:

$$\mathrm{P}\big(y = +1 \,\big|\, \boldsymbol{s}\big) = \frac{1}{1 + \exp(w_0 + \langle \boldsymbol{v}, \boldsymbol{s} \rangle)} \,. \tag{5.15}$$

Let us return to consider all the channels. Let $s_c(\cdot)$ denote the signal for channel c, with $c = 1 \dots C$, and C is total number of channels, and let also $c(j)$ be the channel the j-th feature is to be extracted from. Equation (5.10) becomes

$$x_j = \sum_{t=1}^{T} u_j(t) s_{c(j)}(t) \,, \tag{5.16}$$

and from Equation (5.12) we have

$$\begin{aligned} w_0 + \sum_{j=1}^{n} w_j x_j &= w_0 + \sum_{j=1}^{n} w_j \left(\sum_{t=1}^{T} u_j(t) s_{c(j)}(t) \right) \\ &= w_0 + \sum_{t=1}^{T} \left(\sum_{j=1}^{n} w_j u_j(t) s_{c(j)}(t) \right) \,. \end{aligned} \tag{5.17}$$

If we split the inner summation by grouping features relative to the same channel, we get

$$\begin{aligned} & w_0 + \sum_{t=1}^{T} \left(\sum_{j=1}^{n} w_j u_j(t) s_{c(j)}(t) \right) \\ &= w_0 + \sum_{t=1}^{T} \left(\sum_{c=1}^{C} \sum_{j:c(j)=c} w_j u_j(t) s_c(t) \right) \\ &= w_0 + \sum_{c=1}^{C} \sum_{t=1}^{T} \left(\sum_{j:c(j)=c} w_j u_j(t) \right) s_c(t)\,. \end{aligned} \tag{5.18}$$

Again, there is a term that is the same for all epochs, but now it depends on the channel:

$$v_c(t) = \sum_{j:c(j)=c} w_j u_j(t)\,, \tag{5.19}$$

with $t = 1 \dots T$ and $c = 1 \dots C$.

Putting everything together, in vector notation:

$$\mathrm{P}\big(y = +1 \,\big|\, \boldsymbol{s}_1, \dots, \boldsymbol{s}_C\big) = \frac{1}{1 + \exp(w_0 + \sum_{c=1}^{C} \langle \boldsymbol{v}_c, \boldsymbol{s}_c \rangle)}\,. \tag{5.20}$$

This formula estimates the probability for the target class directly from the epoch signals $\boldsymbol{s}_1, \dots, \boldsymbol{s}_C$. The dot product $\langle \boldsymbol{v}_c, \boldsymbol{s}_c \rangle$ is actually a correlation, as elements of these vectors are time samples, and we can see Equation (5.20) as a classification based on the similarity of epoch signals with some templates ($\boldsymbol{v}_c$). These templates can be considered as the real output of a GA run.

This explains the fact that after some experiments with or without using some of the feature extractors, we found that the first two functions in Figure 5.6 (the triangle and the rectangle) were sufficient. The reason is now clear: If templates are the real output of GA, what the individual features are is not important, as long as they allow the building of complex enough templates.

Figures 5.11 to 5.14 show the ten templates (green solid line) for the ten channels used for the BCI competition data set. Together, the averages for target and nontarget epochs are also shown, for reference. Templates have been multiplied by a scaling factor so that they can be shown in the same graph as the averages; the same factor has been used for all templates in all figures, so no distortion has been introduced. The figures refer to two different runs of the GA (two template sets, i.e., chromosomes, for each run are shown) on the same data set and with the same preprocessing: same channel selection, decimation by a factor of 3, and detrending.

All the template sets shown in the four figures achieved 100% correct letters, yet they are very different. There are some common characteristics, like the negative deflection around 300 ms in Pz, and the positive peak followed by a negative one between 300 and 500 ms in Po7. The negative peak of Pz is present in all the templates that achieved the maximum fitness we have looked at, and is much in accord with the most important feature of the P300 found in the literature: a positive peak about 300 ms after the stimulus, stronger in the parietal region. The sign of the template is negative because the logistic function in (5.20) reaches the maximum for negative values of the exponent.

There also some peculiarities in the templates. For example, there is a strong difference between the two averages in channels Fz, Cz, and C4, but it is ignored in all the examples shown. Yet, the template for C4 in Figure 5.11 shows a strong peak at 550 ms after the stimulus, where the averages almost coincide. This is probably caused by the presence in the training set of epochs for which the difference between target and nontarget signals is significant at 500 ms in C4, and also the features that are more discriminant for other epochs are less effective. Verifying such an explanation is not easy, though, and also difficult is to asses the real impact of a template on the classification, as neither the templates nor the EEG channels can be considered independent. Some hints may come from the comparison of the templates from many good-performing individuals; for example, the fact that the template signals for either channel Po7 or Po8 are rather strong suggests that the activity in the visual cortex is important for the correct classification.

The last consideration suggests that the interpretation of features and weights as templates could permit to select the most important channels for classification. An objective measure of the relative contribution of the various channel could be extracted from Equation (5.20), but our first attempts failed. More work is needed to obtained useful results.

Table 5.27 shows the results obtained on data from the P300 speller experiment run in our laboratory. The classifiers were trained and tested independently for each subject. The test set consisted in the data recorded in the last session for the subjects that recorded a total of three sessions (as shown in table), and the data from the last run for the subjects that recorded just one session; the training set was composed by the remaining data. Data were decimated with a factor of 4, resulting in a frequency of 128 Hz, segmented in epochs from 200 ms before to 800 ms after stimuli, and detrended; the EOG channel was not used. As before, the performance on the test set is given as the

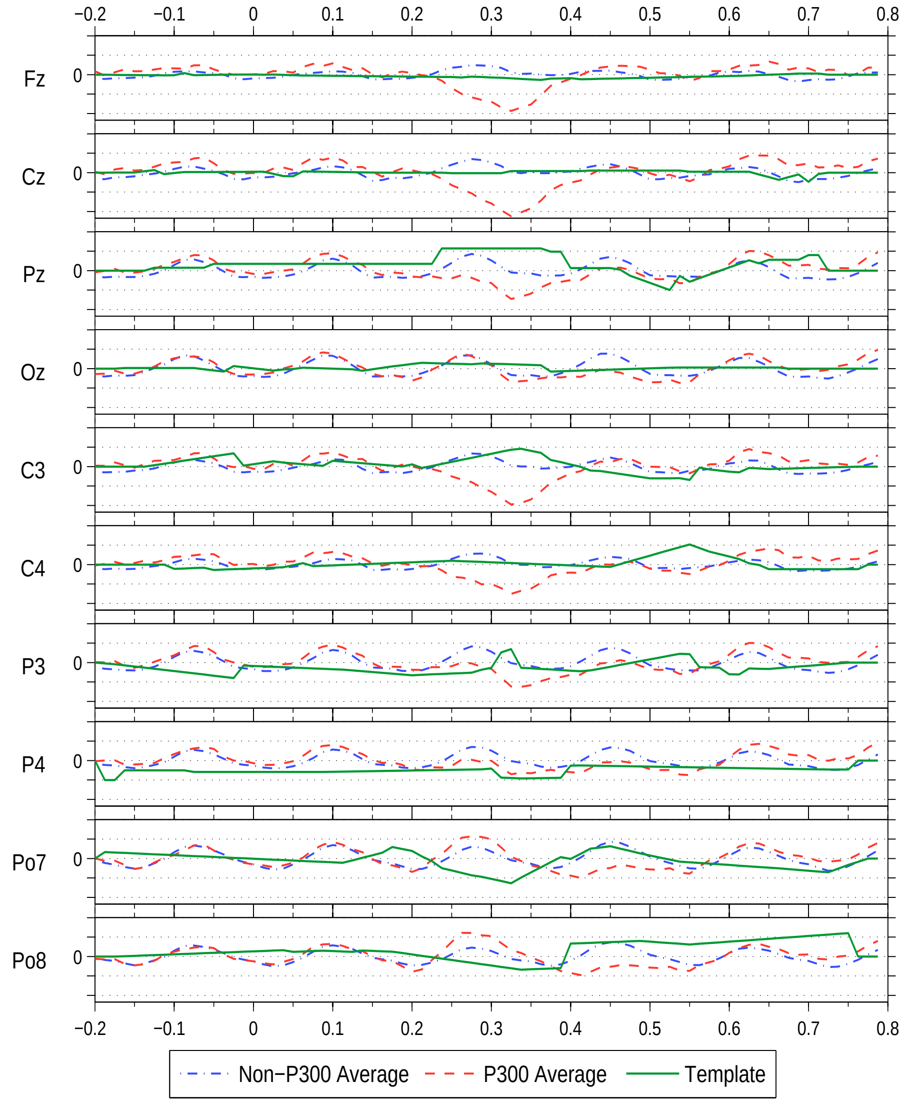


Figure 5.11: Weight templates for a run of GA on the competition data set (first of a set of four). Positive axes point down. Time in seconds.

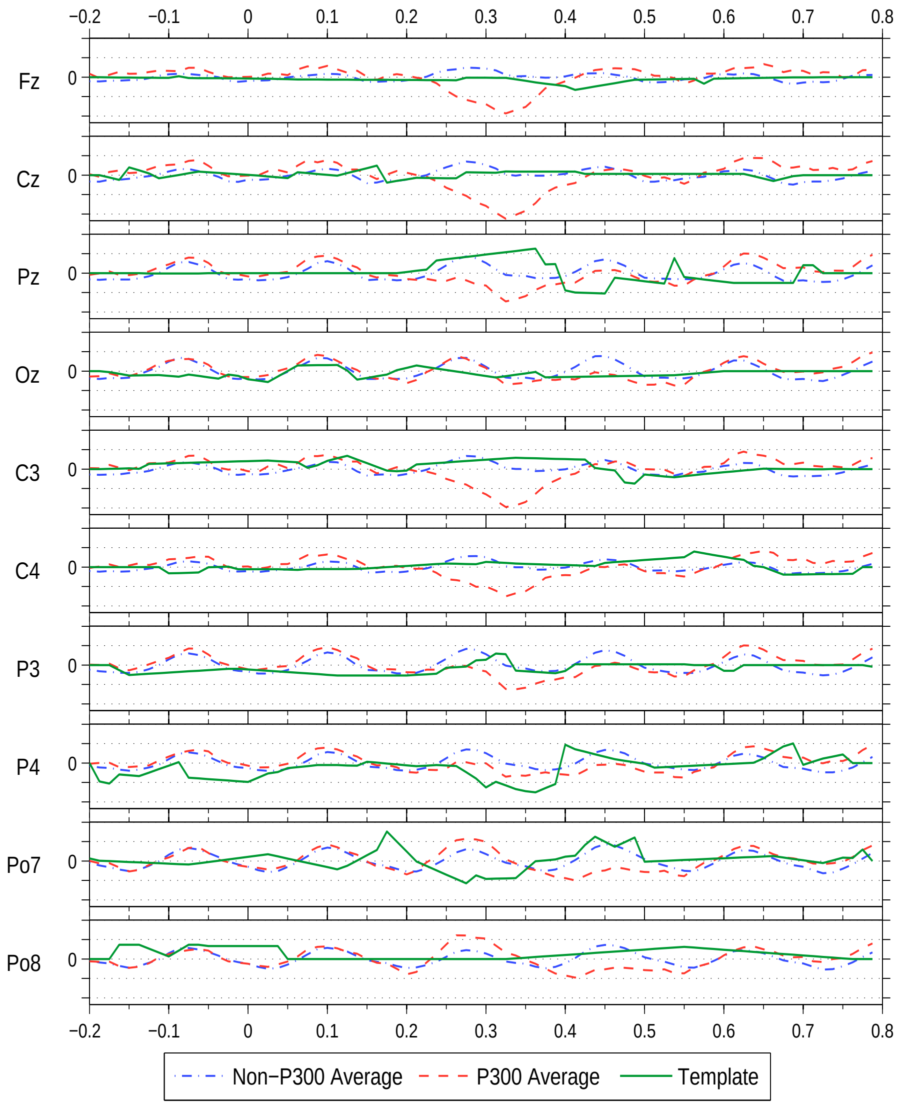


Figure 5.12: Weight templates for a run of GA on the competition data set (second of a set of four). Positive axes point down. Time in seconds.

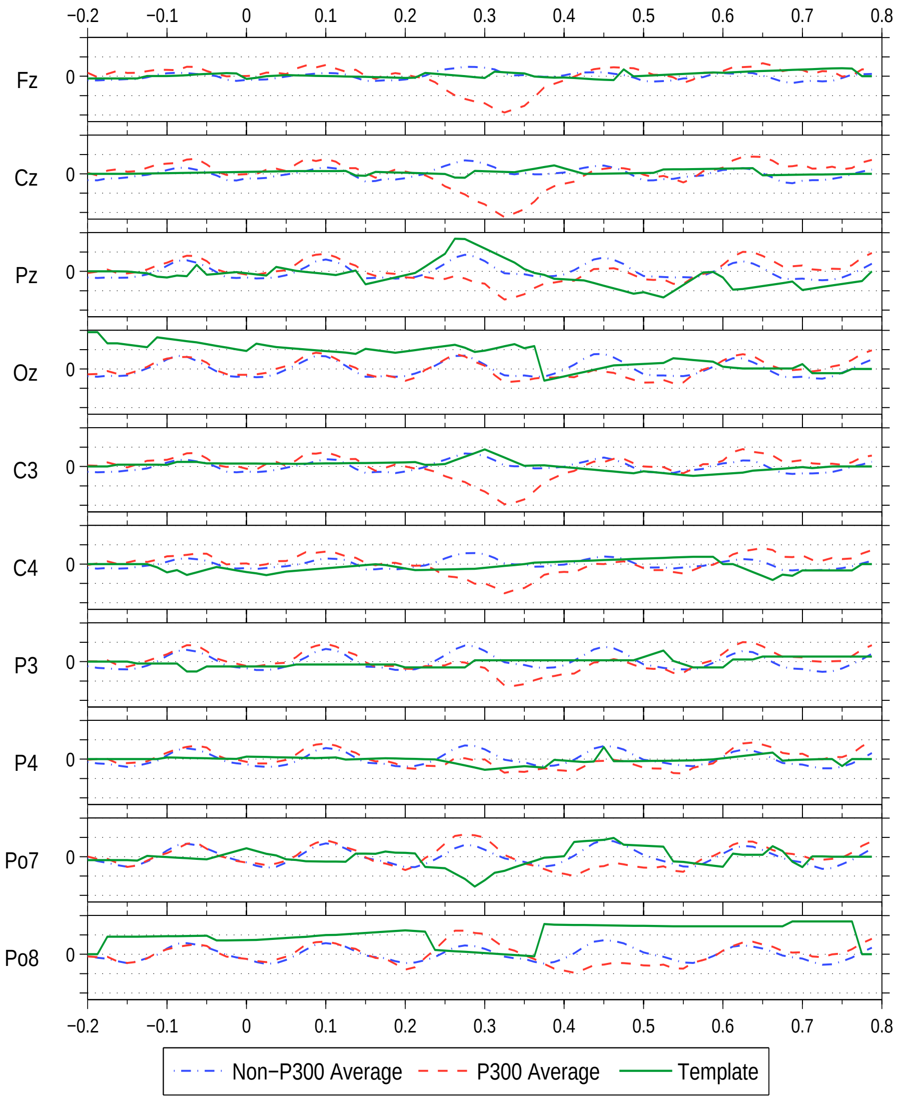


Figure 5.13: Weight templates for a run of GA on the competition data set (third of a set of four). Positive axes point down. Time in seconds.

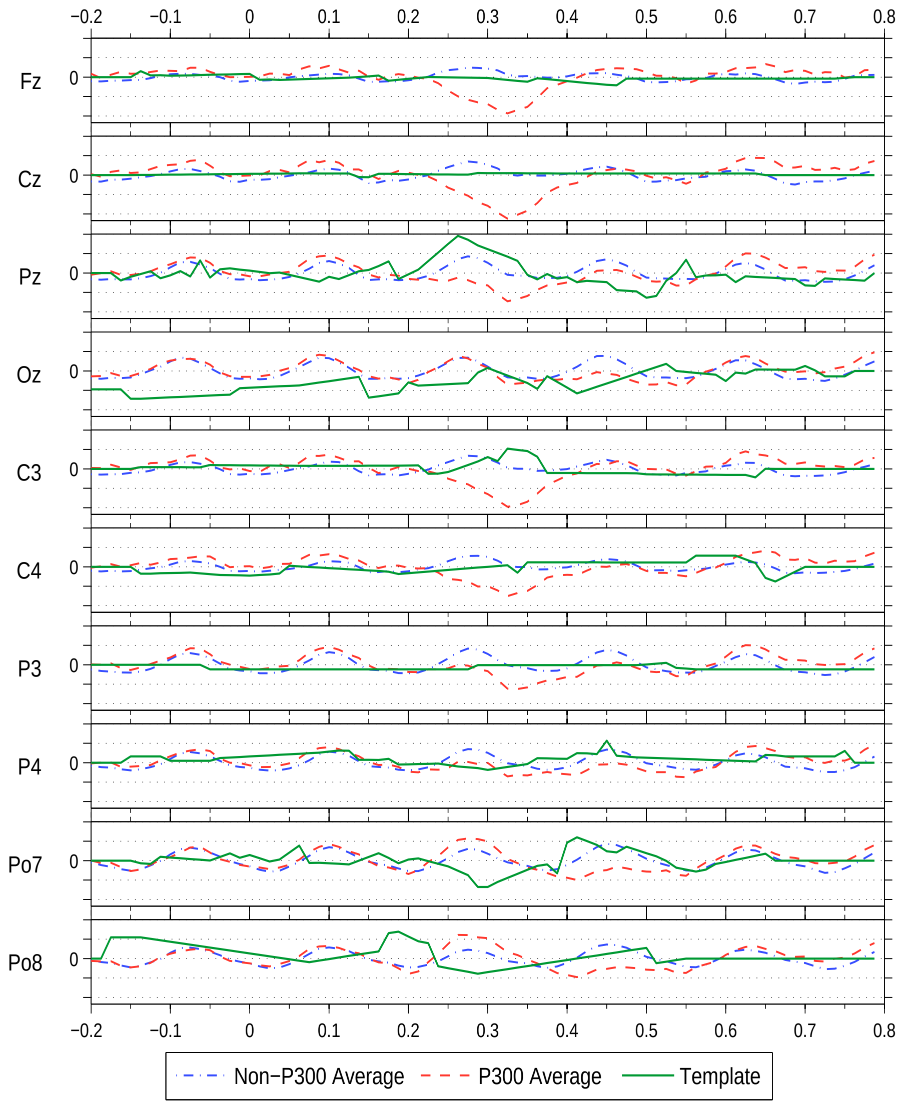


Figure 5.14: Weight templates for a run of GA on the competition data set (fourth of a set of four). Positive axes point down. Time in seconds.

Subject	**No. of sessions**	**Training set size**	**GA Test performance**	**SVM Test performance**
Airlab S1	3	429	121–126/143 (85–88%)	119/143 (83%)
Airlab S2	1	165	24–27/30 (80–90%)	23/30 (77%)
Airlab S3	3	409	89–95/165 (54–58%)	89/165 (54%)
Airlab S4	3	372	91–94/144 (63–65%)	64/144 (44%)
Airlab S5	3	344	80–85/199 (40–43%)	92/199 (46%)
Airlab S6	1	175	14–16/23 (61–70%)	17/23 (74%)
Airlab S7	3	396	46-52/135 (34–39%)	51/135 (38%)

Table 5.27: Results of the GA on the Airlab data set; the performance of SVMs are shown in the last column. Training set size is the number of letters spelled in the training set. Performance is given as the number of correctly predicted letters over the total numbers of letters in the test set.

range of the values obtained for the GA individuals with at least 99% of the maximum fitness; the results come from two different runs of the GA. Two subjects achieved very good results, with more than 80% of correct letters; another subject achieved correct classification rates above 60%, which is still a good result, and the other four subjects obtained less impressive figures, although still much above the level of a random classifier, which is about 3%. Given that we used only 4 channels and 5 repetitions, the overall performance of the GA can be considered pretty good.

Table 5.27 shows also the results achieved by the SVM method illustrated in Section 5.3. By comparing the GA with SVM, we can see that the GA results are always better or comparable to those of SVM; only for Subject S6 the SVM performs significantly better. For Subjects S2 and S4 the GA is significantly better, with a difference of about 20% in correct letters for Subject S4 in favor of GA.

We applied the GA also to the second S. Camillo data set (the second data set was deemed more interesting as it contained more ALS subjects than the first). EEG data were decimated to half of the original frequency; epochs were trimmed to the interval from $-.2\,\mathrm{s}$ to $+.8\,\mathrm{s}$ (i.e., half a second was thrown away), and the linear trend was removed. No normalization of data was performed; we tried to normalize EEG data before running the GA, but it did not change the test performance significantly, so we chose the way that required less computation. The GA was trained on the first three quarters of the available data for each subject, and the features encoded by the best chromosome and the corresponding classifier trained on the training set were applied to the

Subject	**Training**		**Test**		**Recall**	
	Targ.	**Non-T.**	**Targ.**	**Non-T.**	**Targ.**	**Non-T.**
SC12	204	703	99	284	61%±3%	66%±2%
SC15	121	354	58	166	96%±1%	95%±1%
SC16	98	277	57	206	63%±5%	77%±3%
SC17	175	492	61	178	86%±4%	83%±3%
SC18	114	325	32	94	75%±7%	78%±3%
SC19	124	356	52	148	87%±3%	85%±5%
SC20	144	433	50	138	82%±4%	76%±3%
SC21	112	340	39	112	82%±7%	72%±5%
SC23	188	550	91	278	59%±7%	55%±3%
SC25	185	529	50	154	94%±2%	86%±2%
SC26	219	690	47	142	97%±2%	82%±2%
SC13	86	228	30	84	75%±4%	81%±2%
SC14	116	327	34	95	85%±5%	90%±3%
SC22	218	617	84	240	77%±3%	77%±2%
SC24	165	489	61	170	55%±7%	73%±4%

Table 5.28: Results over 14 runs of GA for the second S. Camillo data set: means and standard deviations of recall for targets and non-targets.

remaining quarter of the data. This procedure was repeated with 14 independent runs of the GA on each subjects.

The results are shown in Table 5.28: For each subject, the mean value and the standard deviation of the recall of targets and non-targets over all the 14 runs are presented. Subjects SC13, SC14, S22, S24 are healthy, while all the others are affected by ALS. For most subjects, the performance of the GA is good, achieving consistently more than 70% of recall in 8 subjects (i.e., the mean is at least two standard deviations over 70%). In 6 subjects, the classifiers achieved often more than 80% of correct answers.

Figures 5.15 and 5.16 show what the GA has found for Subject SC15, in two different runs of the GA that obtained 90% of correct epochs in the test set. As before, the continuous green lines represent the weight assigned to individual EEG samples in the final logistic classifier, while the averages of targets and non-targets epochs are given as references. Units are µV and s; the templates have been scaled to fit in the graphs. The averages of targets and non-targets are not very different, yet the algorithm found good classifiers concentrating more on the Fz and Cz channels, between 400 and 800 ms after the stimulus, and basically ignoring Pz, where a P300 should be more prominent.

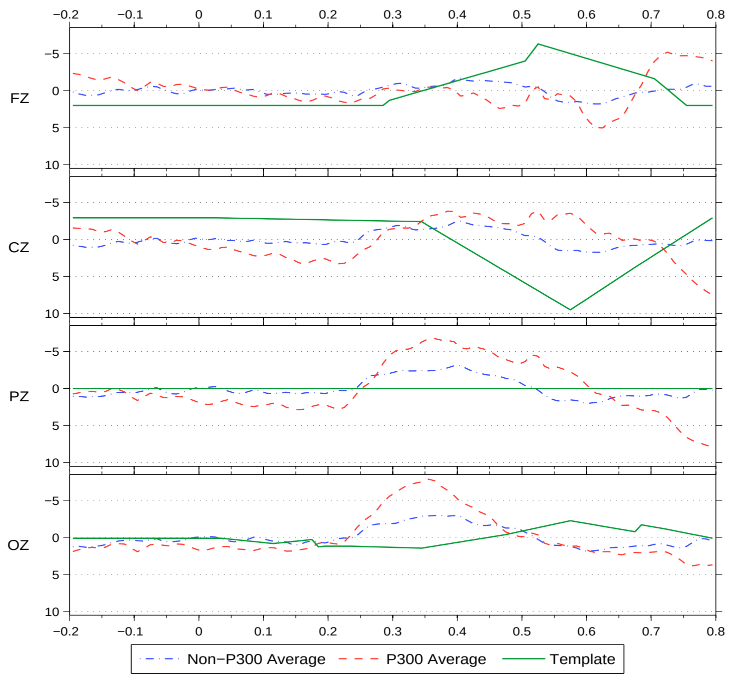


Figure 5.15: Examples of templates obtained in a GA run for Subject SC15 (first in a set of two). Units are µV and s.

A possible explanation of this preference for the Fz and Cz channels can be found in the physiology, by recalling that the P300 complex includes other subcomponents besides P3b (which has its maximal amplitude in parietal regions), as P3a and the slow wave post P3b, which have their maximal amplitude over fronto-central regions. P3a occurs when the changes in physical properties of the novel stimulus are task relevant and attention is switched to the stimulus source, and the slow wave post P3b occurs in the latency range from 500 to 1400 ms, and its amplitude increases as the task becomes more demanding and difficult [119, 120]. This suggests that the characteristics of the task and the attention requirement determine a strong response in the front-central region, which is exploited by the classifiers found by the GA.

5.4.6 Averages As Templates

The templates obtained by the GA show some similarities with the P300 averages in some time intervals, so we wondered if the P300 averages could be used as templates in Equation (5.20). Averages cannot be put

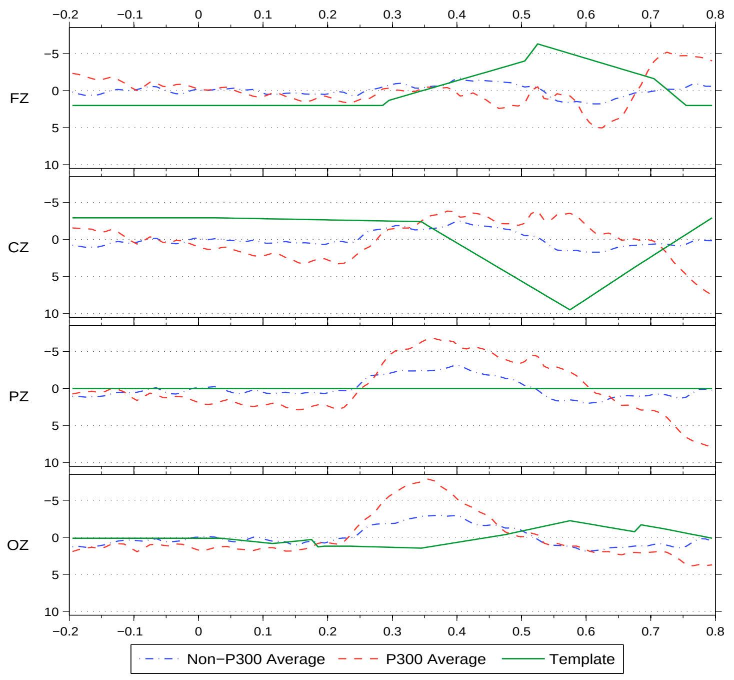


Figure 5.16: Examples of templates obtained in a GA run for Subject SC15 (second in a set of two). Units are μV and s.

in the formula directly, because templates are on a different scale, and therefore we considered the cross-correlation of epoch signals with P300 averages as features to be used in a logistic classifier. The results for such a system are shown in Table 5.29, in the column *Template/Target*. They have been obtained by using the very same training and test sets as for the results shown in the previous pages, and so a direct comparison with the GA results is possible (shown in the column *GA*). Using P300 averages as templates performs worse than the GA, and the use of the differences between the averages of P300 epochs and the averages of non-P300 epochs as templates (shown in column *Template/Difference*) does not improve the results.

5.4.7 GA And Training Time

The actual time needed to run the GA depends mostly on the size of the training set. In our experiments, it took about 15 minutes for the small data sets (e.g., S. Camillo), and up to 10–12 hours for the biggest data sets (Airlab P300 speller). When the training time is so long, one could argue that it could be difficult to adapt the classifier to the

Subject /	**Template**		
Data Set	**Target**	**Difference**	**GA**
Comp. 2	31/31 (100%)	30/31 (97%)	100%
Airlab S1	109/143 (76%)	105/143 (73%)	85–88%
Airlab S2	17/30 (57%)	21/30 (70%)	80–90%
Airlab S3	72/165 (44%)	63/165 (38%)	54–58%
Airlab S4	33/144 (23%)	47/144 (33%)	63–65%
Airlab S5	47/199 (24%)	54/199 (27%)	40–43%
Airlab S6	5/23 (22%)	5/23 (22%)	61–70%
Airlab S7	20/135 (15%)	19/135 (14%)	34–39%

Table 5.29: Results of using grand averages as templates

natural drifting of the ERPs that happens with time or with changing conditions of the user. Yet, our system has not been optimized for speed; it relies on a mix of Matlab and Java and there are some known inefficiencies. A port of the algorithm to a more efficient setup (e.g., pure C language) should dramatically improve the speed. Moreover, even with a training time of a few hours it is possible to run the GA overnight, so that every morning a user can have a new classifier, updated with the recording of the day before. When the GA is used to adapt the classifier to ERP drifts, there are further possibilities to improve speed that can be explored; for example, the GA could reuse the results of old runs instead of starting from scratch.

5.5 Online Results

Here we present some preliminary results of the application of the GA in a functioning (i.e., online) P300 speller, as described in Section 3.2, and in driving the wheelchair described in Section 3.4.

5.5.1 P300 Speller

Three subjects participated in a first set of online experiments. The data were acquired as in our experiments before, i.e., at locations Fz, Cz, Pz, and Oz, and at a frequency of 512 Hz. The P300 speller used 5 repetitions of each stimulation per letter, except for one user, who was tested also with a lower number of repetitions. In these experiments, the users had to select letters indicated to them by the BCI before each trial, so as to simplify the evaluation of the performance. The results are shown in Table 5.30: They are similar to those obtained

Subject	**Training set size**	**No. of repetitions**	**Online performance**
B1	196	5	41/44 (93%)
		4	19/21 (90%)
		3	40/52 (77%)
B2	67	5	57/233 (24%)
B3	108	5	108/181 (60%)

Table 5.30: Results of the GA online. Training set size is the number of letters spelled in the training set. Performance is given as the number of correctly predicted letters over the total numbers of letters in the online usage.

Subject	**Training set size**	**No. of repetitions**	**Online performance**
B1	196	4	74/109 (68%)
B3	108	5	137/202 (68%)

Table 5.31: Results of the GA online in free mode. Training set size is the number of letters spelled in the training set. Performance is given as the number of correctly predicted letters over the total numbers of letters in the online usage.

offline (see Table 5.27), and confirm the validity of the method. The low performance of Subject B2 seems not to be due to the small training set, but to low concentration; the subject reported problems in focusing on the task, perhaps because of a failure of the brightness regulation of the computer screen that affected the recordings.

Subjects B1 and B3 also tried to use the BCI to spell words in the so-called *free mode*, where they spelled words of their own choosing and had to correct errors by selecting *backspace*. The results are shown in Table 5.31 and confirm that the classifier found by the GA can be used to really drive a BCI application. Subject B2 could have tried to use the speller by increasing the number of repetitions, but as the data was recorded also to evaluate error potentials, this would have made the recording sessions much longer.

5.5.2 Wheelchair Driving

A second set of online experiments was made with two subjects driving a wheelchair through a P300 BCI, as described in Section 3.4.2. EEG channels and sample frequency are the same as for the speller: channels Fz, Cz, Pz, and Oz, sampled at 512 Hz. The number of repetitions was always equal to 10. We expected to have a lower performance than

Subject	**Training set size**	**No. of repetitions**	**Online performance**
B2	70	10	93/112 (83%)
B4	99	10	87/94 (93%)

Table 5.32: Results of the GA online for the wheelchair driving task. Training set size is the number of destinations selected in the training set. Performance is given as the number of correctly predicted destinations over the total numbers of destinations in the online use.

in the speller, due to movement artifacts, distraction, and a higher ratio between targets and nontargets, therefore we used a number of repetitions higher than in the case of the speller. Results are shown in Table 5.32. Subject B2 recorded training data on a day and test-drove the wheelchair on two other days; Subject B4 did part of the tests on the same day of training, and part on a different day. The performance obtained is good for both subjects, especially if considered that the wheelchair didn't move very smoothly, and the experiments were made in an environment where there were other people talking and making noise. Moreover, during some experiments an electrode cap was used instead of stand-alone cup electrodes; this is a remarkable fact because electrodes had high impedance (10–30 kΩ), and channel O1 was used instead of Oz without retraining the classifier[4]. The P300 classifier worked equally well with both montages.

[4]The use of this setup was not planned in advance, but was due to a coincidence. In the same days of the wheelchair experiments we had to test a new conductive gel with the electrode cap; so we tried to use the electrode cap with the GA classifier and the wheelchair BCI just to see if we got meaningful signals. After unexpectedly achieving 100% correct targets in the first run, we decided to use the cap in more experiments on both subjects, as a test of the robustness of the classifier.

6 ErrP Detection

‘As I said, the Big Bosses, ay’, his voice sank almost to a whisper, ‘ay, even the Biggest, can make mistakes’.

J. R. R. Tolkien – *The Lord Of The Rings*

Real BCIs sometimes misclassify user intent, and much research is devoted to improving BCI classification ability in order to improve their performance. An alternative way to improve BCI performance is the early identification of errors and their automatic correction. We know from the literature that BCI errors elicit error potentials (ErrP), and the detection of ErrP could be a viable way to improve BCI performance. In this chapter some experiments with error potentials, arising from mistakes both of subjects and of the machine they are interacting with, are presented. The chapter is closed by some considerations about the impact that ErrP detection may have on the performance of a BCI; in particular, the case of a P300 speller is studied.

6.1 User Errors

Our interest in error potentials lies in using them for BCI, as a mean to improve the performance of a BCI, by automatically aborting the commands that the BCI has got wrong. Nonetheless, we concentrated first on ErrPs induced by of human mistakes, on which the knowledge appears to be more consolidated. We designed two experiments to this aim, taking inspiration from the literature.

Eriksen Task There are different tasks that have been used to study ErrPs in the literature; the first task we concentrated on is one of the many variants of the task originally presented by Eriksen and Eriksen in [121], which appears to be widely used to study attention and error potentials. Subjects are presented with stimuli on a computer screen in rapid succession, each consisting of one target arrow pointing either left or right, and two other “noise” arrows pointing either in the same or the opposite direction of the target arrow (see Figure 6.1); subjects are supposed to press one of two buttons, either left or right, indicated by the

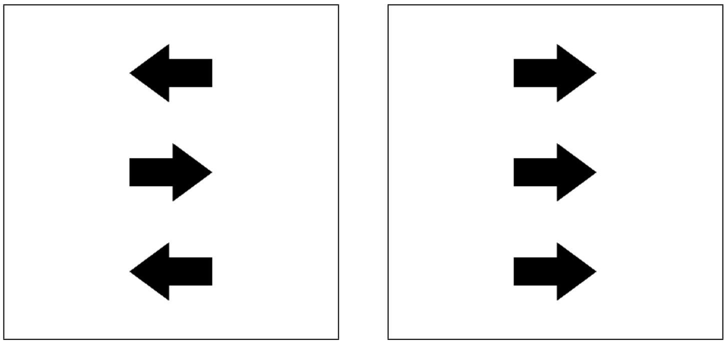


Figure 6.1: Graphical interface used for Eriksen tasks. In both cases the subject is supposed to press “right”; the top and bottom arrows are used as a disturbance.

target arrow. Subjects are asked to respond as quickly as possible, and in fact they have a short time limit to press the button ; the exact length of this limit is defined during a preliminary session, where subjects also learn to respond fast. The limit is chosen long enough that a subject is able to respond in time almost always, but short enough to induce him to make mistakes, i.e., to press the wrong button; we used limits ranging from 300 ms to 400 ms, depending on the subject. The direction of the target and noise arrows are randomized and balanced, so that the four possible combinations are equally represented. In order to avoid visual ERPs interfering with ErrP, arrows are left on the screen for 500 ms after the button pressing.

Each subject participated in several sessions, each one consisting of 100 trials (stimuli) separated by 1.5 s; EEG at positions Fz, Cz, Pz, Oz, and also EOG were recorded at 512 Hz, as explained in Section 3.1. During recording sessions, if a subject failed to respond within the given time, a sound signaled the fact and the trial was discarded from further analysis.

In this kind of task the subject is not told of an error by the system, but he becomes soon aware of it, as the error is due only to time pressure. Following the literature (e.g., [63]), we measured time beginning from the stimulus for our analysis. This is reasonable, because it is the event of pressing the button that generates the error potentials[1].

Five young healthy subjects performed Eriksen tasks in one or two sessions, divided in runs of 100 trials each. Figure 6.2 shows the

[1]Out of curiosity, we compared the averages of epochs synchronized on the stimulus and on the key pressing. Indeed, the difference between epochs with and without errors is stronger when the synchronization instant is the key pressing.

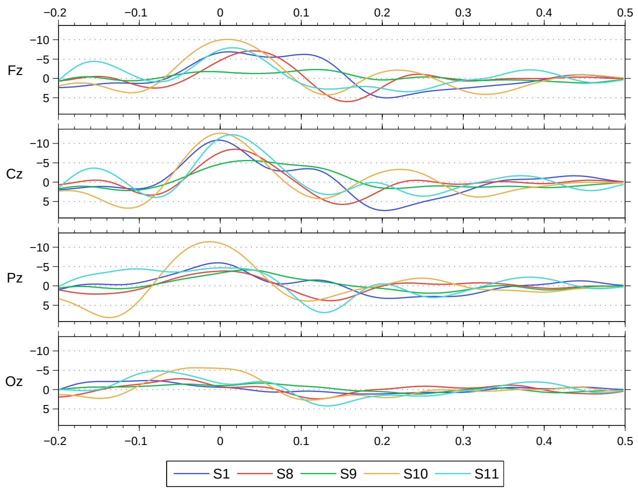


Figure 6.2: Error-minus-correct means for five subjects in Eriksen tasks. Units are seconds and microvolts. Time 0 is button pressing time.

difference between the averages of ErrP epochs and the averages of non-ErrP epochs (error-minus-correct means) for all subjects, band-passed in the range 1–10 Hz. There is an evident Ne component in all subjects, stronger in Cz, followed by a Pe component except for Subject S9; the negativity peak coincides with the time of button pressing, which is consistent with the theory that subjects realize of their error even before moving their fingers, but too late to suppress the movements.

Stopwatch Task In the "stopwatch" task, a subject is asked to mentally estimate a fixed amount of time, which is very similar to what was presented in [67]. This task is more similar to the way a BCI could exploit error potentials, as subjects are not aware of an error until the system they are interacting with gives them a feedback.

More precisely, in each trial the screen is initially blank; a black circle appears on the screen, and this signals the subject to start mentally keeping track of the time passing. When the subject thinks that a predetermined amount of time has passed (5 s in our first experiments, then we reduced it to 3 to speed up recordings), he presses a button, and this makes the circle disappear. After a short pause, 500 ms, a feedback appears as a black circle containing either the word "OK" or "NO", depending on whether the time estimated by the subject has been close enough to the requested time. The feedback remains on the screen for a

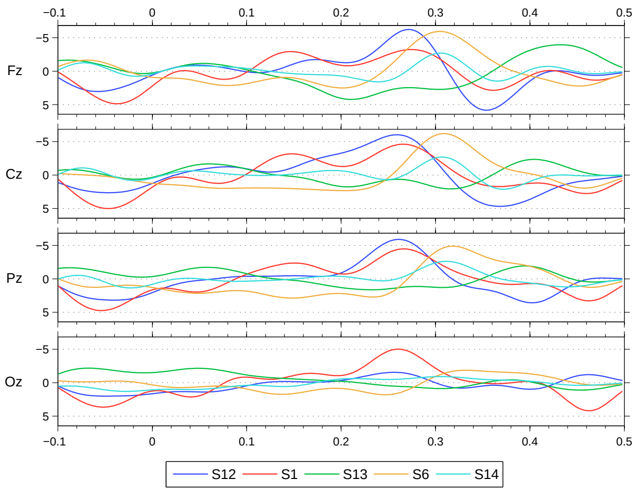


Figure 6.3: Error-minus-correct means for five subjects in stopwatch tasks. Units are seconds and microvolts. Time 0 is feedback time.

second, and then it disappears, leaving the screen blank until the next trial, which begins 3 s after.

Some subjects seem to learn to estimate time better during each session, but their performance drops again when a new session begins. If the tolerance for the estimation error were fixed, such subjects would make many mistakes at the beginning of a session, and virtually none near the end. In order to keep the error rate uniform within a session, we used a dynamic tolerance: After every successful trial (i.e., when the subject estimates the time within the current tolerance), the tolerance is reduced by a few hundredths of seconds; after two consecutive errors, the tolerance is increased by a few tenths. The initial value of the tolerance has been determined in preliminary experiments and set to ±400 ms for all subjects; during experiments, it went from as low as 50 ms to more than 1 s.

Five young healthy subjects performed stopwatch tasks in one or two sessions, divided in runs of 20–30 trials each; EEG and EOG were recorded, with the same montage and frequency of the Eriksen task. Figure 6.3 shows the error-minus-correct means, again band-passed in the range 1–10 Hz. For this task, ErrPs are studied by using the feedback appearance as time 0. There is an evident Ne component in all subjects, although weaker for two of them, with latencies ranging from 260 ms to more than 400 in one case; a Pe is clearly discernible only for three subjects. Subject S1 presents a visible oscillation at about 8 Hz, which

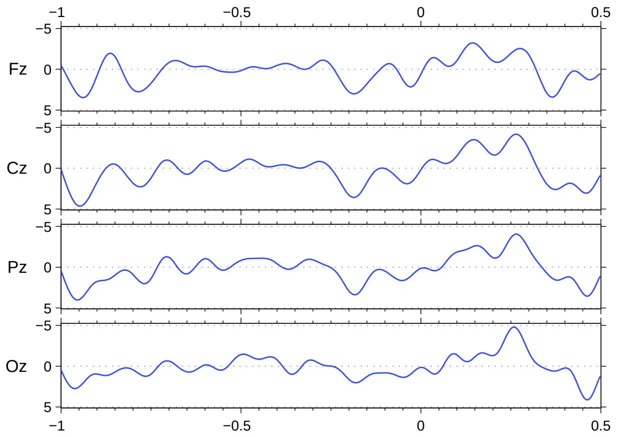


Figure 6.4: Error-minus-correct means for Subject S1 in stopwatch tasks. Units are seconds and microvolts. Time 0 is feedback time.

is not probably related to ErrP, as it is present even a second before the feedback. The fact that this oscillation is weaker ca. 500 ms before feedback, i.e., the time of button pressing, and it is more evident in channel Cz (see Figure 6.4) suggests that maybe it is a kind of μ rhythm.

6.1.1 Automatic Detection

ErrPs can be useful in a BCI only if they can be detected automatically; here, the method we used is described.

First, data are segmented in epochs ranging from 100 ms before the synchronization instant to 500 ms after it. The synchronization instant is the key pressing for the Eriksen task and the feedback onset for the stopwatch task. Epochs containing strong EOG activity are discarded before further analysis; in particular, if the EOG exceeds at any time the threshold of 50 μV, the epoch is discarded.

The implication of discarding epochs in a BCI is that in those cases there is no way to correct possibly wrong responses. Obviously, robustness with respect to EOG contamination is a desirable property for a detection algorithm, but we preferred to concentrate first on producing something working, and leave the improving of its robustness for the future. The results of a first work where a very simple idea is used to evaluate and address this problem are described in Section 6.3.

After discarding noising epochs, remaining data are then filtered in the band 1–10 Hz; this filtering enhances the signal-to-noise ratio by eliminating frequency components extraneous to ErrP. Filtered signals show, on average, a difference between ErrP epochs and non-ErrP ones, but only in some time intervals; these intervals are those in Figures 6.2 and 6.3 where the means are not zero. We decided to concentrate only on the these intervals for ErrP detection, and developed a way to automatically determine significant intervals.

Student's t-test is used to find time points where the ErrP and non-ErrP epochs differ; for each channel and time point, the distribution of signal samples are divided in two groups, depending on whether they came from ErrP or non-ErrP epochs, and their mean is compared. In formulas, this means considering signals $s_{c,1}(t)$ from channel c of ErrP epochs and $s_{c,0}(t)$ from the same channel of non-ErrP epochs as random variables, where t ranges on the sampling times from -0.1 to $0.5\,\mathrm{s}$. The t-test is used to see if, for any given t and c, the mean of $s_{c,1}(t)$ differs significantly from the mean of $s_{c,0}(t)$; the significance level has been chosen to be 0.01, but much smaller p-values have been often found in analyzing data.

The t-test requires that the distributions of the samples under test be normal and have the same variance. This has been verified by means of normal probability plots drawn for a subset of time instants. In a normal probability plot, samples are plotted in a graph as empirical probability against values. Although in some cases the tails of the observed distributions are longer than the Gaussian ones, the departure from normality is never dramatic. It must also be noted that the t-test is used only to get a rough idea of where the two averages differ significantly, and it is not important if the p-values are not very precise. Equality of variance has been also tested in a more formal way by applying the F-test to data from some subjects, and it is verified at significance level around 0.01. This check has been done only in a preliminary analysis to verify our assumption; the automated procedure skips this step.

The t-test is applied at every sampling time, and it finds interesting time points. These points tend to lie in groups, because the filtered signals have a strong autocorrelation for short lag; but intervals may be many and of different sizes, with "holes" in between (see the top part of Figure 6.5 for an example), while we are interested only in finding one contiguous time interval containing all the interesting features of signals. For this reason, a clustering algorithm is run on the time points found by the t-test to fill holes and discard isolated points or small intervals. We chose DBSCAN [122], a clustering algorithm based on density, because it clusters together nearby points and ignores outliers, which is exactly what we need. More precisely, DBSCAN has two parameters, k and ε, and divides points in three categories: *core points*, whose neighborhood

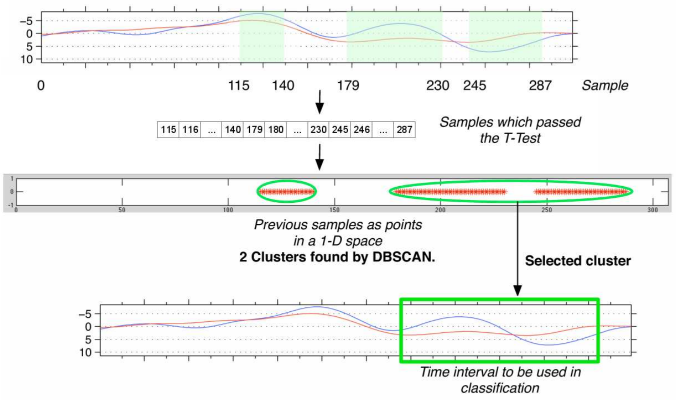

Sample
Samples which passed the T-Test
115 116 ... 140 179 180 ... 230 245 246 ... 287
Previous samples as points in a 1-D space
2 Clusters found by DBSCAN.
Selected cluster
Time interval to be used in classification

Figure 6.5: Procedure for the identification of significant intervals. Top: areas in green contain the samples that passed the t-test with a p-value of 0.01 or less. Middle: clustering of samples. Bottom: the interval used for classification.

of radius ε (ε-neighborhood) contains at least k points, *border points*, which are contained in the ε-neighborhood of a core point and are not core points themselves, and *noise points*, which are all the other points. Core points that are less than ε apart are iteratively fused in one cluster, and border points are assigned to the same cluster of the core point in whose ε-neighborhood they are contained. Noise points do not belong to any cluster.

For our application, a cluster is transformed in an interval by taking the smallest interval containing the cluster; in other words, any gap is filled (see Figure 6.5). For the DBSCAN parameters, we have chosen $k = 10$ and $\varepsilon = 60\,\text{ms}$; k is also the minimum number of points in a cluster (a cluster contains at least a core point and the points in its neighborhood), so the smallest interval found by DBSCAN must be at least k samples long, which, in our case, means $10/(512\,\text{Hz}) \approx 20\,\text{ms}$. ε is also the minimum distance between two core points in different clusters, hence sequences of contiguous points from the t-test less than 60 ms are fused together; when Ne and Pe components are present, they generate (more or less) contiguous sequences of points passing the t-test, and these sequences are fused by clustering as they are closer than 60 ms.

If more than one interval (cluster) per channel is found by DBSCAN, only the largest interval will be considered in the further processing steps. More often than not, the channels with the most significant interval found by the t-test have been Cz and Fz, in accord with the literature. For this reason, we use only these two channels for epoch classification. As a further simplification, the intervals found by DBSCAN for the two channels are fused together (the minimum interval encompassing both is taken), and used for both channels.

The significant intervals are used to extract two different kinds of features: raw sample values and coefficients of polynomial approximation. Features of the first kind are just the values of the EEG samples falling within the time interval found with the above method for the channels Fz and Cz of each epoch. The second kind of features is computed by fitting (in the least square sum sense) two third-grade polynomials to the EEG signal from the Fz and Cz channels of each epoch; the 8 (4+4) coefficients of both polynomials represent the extracted features.

Features are then fed to classifiers; we used a total of four different classifiers: LDA, k-NN, a Bayesian classifier, and a SVM. A *linear discriminant analysis* (LDA) classifier was used for both kinds of features. If we call $\boldsymbol{x}$ the vector of features, then LDA finds a vector of weights $\boldsymbol{w}$ and a scalar c such that $\langle \boldsymbol{w}, \boldsymbol{x} \rangle - c = 0$ is the linear decision boundary between two regions of the feature space corresponding to the assignments of either class in a binary classification problem (see [123] for more details).

Another classifier used on raw samples is *k nearest neighbor* (k-NN) [124, 125], which is probably the simplest classifier one can think of and still achieve respectable performance. When k-NN classifies a given example, it looks at the k nearest examples in the training set and chooses the majority class. We used 5 as the value of k, and the Euclidean distance in the feature space as the metric for distances between examples. In presence of noise, the fact that there are more examples of one class than another biases the classifier in favor of the more represented class; in order to reduce this effect, randomly chosen non-ErrP epochs are discarded from the training set so as to equalize it.

We used also two methods from the literature, which were already been applied to detect other event-related potentials: the Bayesian method described in [116], and the SVM-based approach [58] already used for P300s, as seen in Section 5.3. The SVM-based method was applied to all the four channels of whole epochs, as a sort of a check that the selection of channels and time interval would not lower the classification performance.

The Bayesian approach cited above considers signals $s_{c,1}(t)$ and $s_{c,0}(t)$ at time t from channel c of respectively ErrP and non-ErrP epochs as Gaussian random variables, and it also assumes that samples from different channels or taken at different times are independent and have

Subject		Size	LDA	Bayes	k-NN	P. LDA	SVM
S1	ErrP	309	79%	73%	74%	85%	85%
	N-ErrP	1390	84%	74%	83%	81%	87%
S8	ErrP	166	58%	61%	62%	66%	67%
	N-ErrP	831	73%	65%	68%	69%	72%
S9	ErrP	189	57%	65%	47%	68%	70%
	N-ErrP	366	69%	66%	76%	70%	75%
S10	ErrP	35	60%	74%	63%	69%	69%
	N-ErrP	189	80%	75%	86%	81%	72%
S11	ErrP	83	67%	58%	67%	81%	81%
	N-ErrP	760	83%	68%	71%	81%	81%

Table 6.1: 10-fold cross-validated results of ErrP detection in Eriksen tasks

the same variance. This results in a very simple discriminant function

$$\sum_c \sum_t (s_c^{(i)}(t) - \mu_{c,0}(t))^2 < \sum_c \sum_t (s_c^{(i)}(t) - \mu_{c,1}(t))^2 \tag{6.1}$$

where $s_c^{(i)}(\cdot)$ is the signal from the channel c for the epoch i, and $\mu_{c,0}(t)$ and $\mu_{c,1}(t)$ are the mean for the channel c at time t of, respectively, all the ErrP and non-ErrP epochs.

While the independence assumption is clearly false, the model would be too complex to be used without it, and the final discriminant function (6.1) seems to work nonetheless. It is interesting to note that Equation (6.1) states that an example is classified depending on the distance from two points in the feature space, which are the means taken over the training examples for each class.

6.1.2 Classification Results

Results of the methods just described in detecting ErrPs are shown in Tables 6.1 and 6.2 for the Eriksen and stopwatch tasks respectively. Percentages represent the fraction of ErrP or non-ErrP epochs correctly classified (recall). To have a robust estimation of the performance, we applied 10-fold cross-validation to the classification step; the step that finds the interesting time interval was not cross-validated, because it has been seen to vary very little when discarding a small number of examples. The *size* shown in the tables is the number of epochs, either ErrP or non-ErrP, that remained after discarding the most noisy ones.

In general, the best classifiers seem to have been SVM and LDA applied to polynomial coefficients. The additional information made available to SVMs does not appear to have given any advantage to

Subject		**Size**	**LDA**	**Bayes**	***k*-NN**	**P. LDA**	**SVM**
S12	ErrP	39	54%	67%	74%	59%	67%
	N-ErrP	99	68%	70%	74%	75%	74%
S1	ErrP	89	56%	66%	69%	60%	61%
	N-ErrP	287	63%	70%	73%	67%	70%
S13	ErrP	38	58%	66%	64%	58%	68%
	N-ErrP	33	64%	61%	64%	64%	18%
S6	ErrP	60	47%	73%	60%	67%	68%
	N-ErrP	117	64%	74%	58%	71%	74%
S14	ErrP	75	53%	51%	55%	52%	56%
	N-ErrP	121	63%	60%	60%	65%	64%

Table 6.2: 10-fold cross-validated results of ErrP detection in Stopwatch tasks

Subject		**Size**		**Performance**
		Train.	**Test**	
S1	ErrP	181	139	86–88%
	N-ErrP	935	513	84–86%
S8	ErrP	108	133	61–70%
	N-ErrP	580	586	62–70%
S9	ErrP	122	126	80–86%
	N-ErrP	328	173	51–59%

Table 6.3: Results of ErrP detection with GA in Eriksen tasks

them; thus, concentrating on shorter time intervals seems a good idea, as it permits to achieve the same performance but requires less resources than using the whole epoch. The Bayesian classifier and k-NN performed a little worse in general, although in a couple of cases outperformed the two best classifiers by a small margin. The Eriksen task seems to elicit ErrPs that are more easy to classify than those from the stopwatch task. Subject S13 has recorded only few epochs (71 remained after discarding EOG-contaminated epochs), and this small number may explain the poor performance, not mentioning the fact that the ErrP seen in Figure 6.3 is not very strong. Subject S14 has the weakest response and indeed classification is slightly better than random.

We tried also the genetic algorithm (GA) described in Section 5.4 on some data recorded in Eriksen tasks. Table 6.3 shows the results, expressed as percentage of correct classification for the two classes. In this case, epochs from 200 ms before the key pressing to 600 ms after were used, with a decimation factor of 4 (128 Hz was the resulting frequency) and detrending; also, removal of epochs with noisy EOG was performed. Data was divided in two sets, one for training and one for testing, and

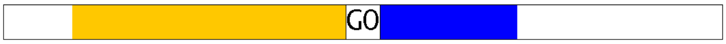


Figure 6.6: Graphical interface for the Ferrez-Millán task

the GA was run twice for each subject. Fitness was a combination of precision (i.e., the number of correctly predicted examples of a class over the number of examples predicted as belonging to that class) and recall for ErrP, with a weight of $^2/_3$ for precision, and $^1/_3$ for recall. The performance for the first two subjects is good, in line with the results of the best classifiers in Table 6.1, while the GA appears to have some difficulties in recognizing trials without ErrP of Subject S9. These results suggest the GA could be a good method to classify also ErrPs, and not only P300s, but a more extensive work is needed to verify its effectiveness.

6.2 Machine Errors

In the previous section we have studied errors made by human subjects, but in the BCI field, ErrPs due to machine errors are more relevant. For this reason, we have set up two different experiments to study this kind of ErrP.

Ferrez-Millán Task In the first experiment we replicated the settings described by Ferrez and Millán in [72]. The interface is shown in Figure 6.6; the user is supposed to fill either the left or the right bar by pressing one of two buttons. Each bar is composed by ten parts, and every time the user presses a button, the next part gets filled, according to the button pressed; and when either bar is full, the trial ends. The user is free to choose either left or right, but the interface is programmed such that with a probability of 20% the wrong bar grows. To avoid unwanted overlapping of different parts of a trial, the user is not supposed to press any button until the word "GO" appears at the center, which happens after a pause of at least 1.5 s from the previous trial; feedback, i.e., bar filling, is given 500 ms after a button pressing, when all motor activity should be over.

Subjects are fully aware of how the system works; nonetheless, epochs relative to "errors" are significantly different from epochs with no errors,

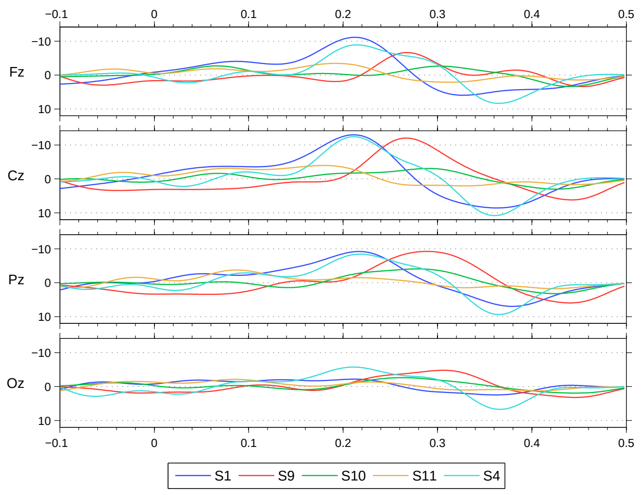


Figure 6.7: Error-minus-correct means for five subjects in Ferrez-Millán tasks. Units are seconds and microvolts. Time 0 is feedback time.

as shown in Figure 6.7. In three subjects there is a very strong Ne component (more than 10 μV), followed by a weaker Pe, while the other two have weaker responses but still visible (the vertical scale of Figure 6.7 is almost four times that of Figure 6.3). Results in Table 6.4 reflect the intensity of the responses: the classification work best with the subjects with the strongest responses. For Subject S10, results are practically random, except for the Bayesian classifier, which achieved good results also for the other subjects. LDA applied to polynomial interpolation and SVM appear in general to be a little worse than the Bayesian classifier, but better than the other k-NN and LDA applied to raw samples. All in all, it seems that we have been able to discriminate ErrP even in this case, more related to BCI.

P300-Speller Task In the last experiment we made, a BCI was simulated by following the P300-speller protocol described in Section 3.2. The interaction with the BCI was simulated in the sense that subjects were told that the BCI would select letters basing on subjects' P300s, while in reality it selected the right letter with a probability of 80%, and did not consider the EEG recordings at all. The letter to spell was told to the subjects before the beginning of each trial, and when a wrong letter was spelled, the subject had to choose the *backspace* symbol in the next trial.

Subject		Size	LDA	Bayes	*k*-NN	P. LDA	SVM
S1	ErrP	165	72%	78%	68%	73%	80%
	N-ErrP	544	83%	82%	82%	78%	86%
S9	ErrP	38	68%	76%	66%	74%	71%
	N-ErrP	104	83%	85%	84%	84%	83%
S10	ErrP	36	53%	69%	42%	47%	50%
	N-ErrP	141	59%	65%	47%	64%	62%
S11	ErrP	74	59%	70%	65%	65%	59%
	N-ErrP	251	72%	72%	72%	69%	73%
S4	ErrP	49	61%	76%	76%	80%	80%
	N-ErrP	221	86%	80%	82%	83%	82%

Table 6.4: 10-fold cross-validated results of ErrP detection in Ferrez-Millán tasks

We chose an 80% probability because it is reasonable for such a BCI, and it is low enough to have a considerable number of error epochs without frustrating users too much and inducing them in believing that the BCI does not work.

Five subjects took part in this experiment, each in three separate sessions. Figure 6.8 shows the usual error-minus-correct means, filtered in the 1–10 Hz band. Four out of five subjects have a strong Ne with a peak at about 300 ms, followed by a proportionate Pe 100 ms after. The response of Subject S3 is rather weak, with peaks of less than 3 μV in absolute value.

Compared to the most similar task, the Ferrez-Millán, latencies appear to be longer, and this is true also for Subjects S1 and S4, which took part to both experiments: The negative peak is about 100 ms later for the P300-speller task than for the Ferrez-Millán task. This might suggest that the more complex the task, the bigger the latency is, but the number of subjects employed is not enough to support any such correlation (see also Figure 6.9 and its discussion at Page 117).

Table 6.5 shows the classification results, but with a difference with the respect to the previous tables: This time, the three sessions of each subject were used as the folds of a 3-fold cross-validation. Except for Subject S3, the best classifiers (SVM and LDA with polynomial coefficients) reach about 80% of recall, which is good. The performance for Subject S3, whose responses in Figure 6.8 are the weakest, is just slightly better than random.

We tested the GA described in Section 5.4 also on the ErrP data recorded in the P300 speller task; the results are shown in Table 6.6. The preprocessing of data is very similar to the one used for the method described through this chapter. Epochs running from 200 ms before the

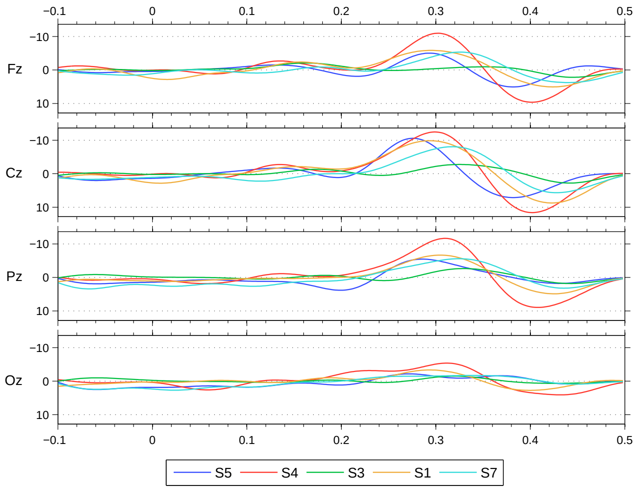


Figure 6.8: Error-minus-correct means for five subjects in P300-speller tasks. Units are seconds and microvolts. Time 0 is feedback time.

Subject		**Size**	**LDA**	**Bayes**	***k*-NN**	**P. LDA**	**SVM**
S1	ErrP	89	69%	65%	65%	69%	79%
	N-ErrP	352	86%	75%	87%	83%	84%
S3	ErrP	56	54%	46%	50%	54%	52%
	N-ErrP	284	66%	55%	62%	68%	56%
S4	ErrP	83	59%	72%	64%	72%	70%
	N-ErrP	384	81%	81%	82%	84%	84%
S5	ErrP	88	74%	75%	68%	77%	84%
	N-ErrP	434	86%	76%	86%	88%	82%
S7	ErrP	41	54%	66%	63%	73%	71%
	N-ErrP	245	81%	72%	80%	87%	80%

Table 6.5: 3-fold cross-validated results of ErrP detection in P300-speller tasks

Subject	Training		Test		Recall	
	ErrP	N-ErrP	ErrP	N-ErrP	ErrP	N-ErrP
S1	66	268	23	94	69%±6%	90%±3%
S3	58	223	14	111	55%±11%	61%±4%
S4	65	296	23	112	75%±5%	81%±2%
S5	50	283	38	156	67%±6%	82%±6%
S7	38	202	10	61	64%±10%	76%±6%

Table 6.6: Results over 10 runs of GA for ErrPs in the P300-speller task

stimulus to 600 ms after, were decimated by a factor 4 (128 Hz was the resulting frequency), and detrended; epochs with strong EOG activity were removed. The training set for each subject is composed by the data recorded in the first two sessions, and the test set by the data of the third session. The fitness function is the combination of precision and recall of ErrPs described by Equation 5.8 (Page 81). Ten independent runs of the GA were executed for each subject, and the performance on the test set of the chromosome with the highest fitness was measured. Table 6.6 shows the mean value and the standard deviation of the recall of targets and non-targets over all the runs for each subject.

If we compare Tables 6.5 and 6.6, we can see that the results are rather similar. If we compare the LDA classifier applied to polynomial coefficients (column *P. LDA*), which was consistently one of the best classifiers, with the GA, the major difference can be seen for Subjects S5 and S7, which happens to be also the two subjects with the smallest number of ErrPs in the training set. It seems that the GA is more sensitive to the size of the training set; this may warrant to reserve the GA for the recognition of P300s, for which it is easier to collect more epochs. As a side note, the number of epochs between the two tables differ slightly; this is due to the fact that the rejection of epochs contaminated by EOG was applied after the decimation for the data used by the GA. The difference in processing is the results of the way the software running the genetic algorithm, originally developed for P300s, has been adapted to deal with ErrPs.

We can conclude that even in this case of the P300-speller task, which is the nearest thing to ErrPs elicited by a real BCI, automatic detection of ErrPs has been possible with good results (except for one subject). This is very promising for the improving BCI performance by means of ErrPs.

Comparison Of Classification Intervals Figure 6.9 shows the intervals found by the DBSCAN algorithm and used for the classification of ErrPs

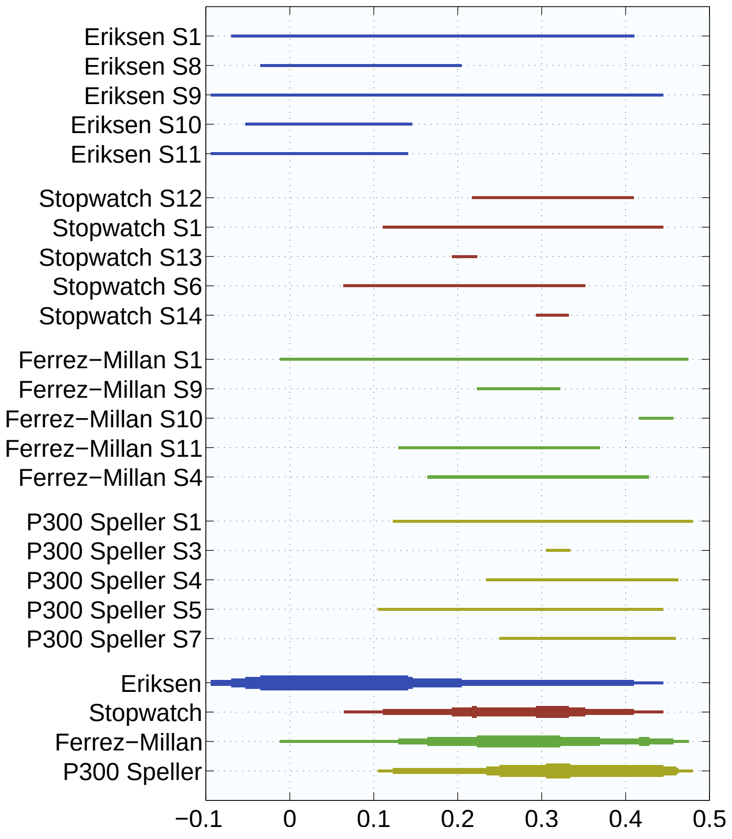


Figure 6.9: Intervals used for ErrP classification for different subjects and tasks. The four variable-width lines in the lower part represent the number of times each time interval has been used for the given task. Time on the horizontal axis is in seconds.

for the various subjects and tasks. Also, the variable-width lines in the lower part show how often a particular interval has been used for the given task (the thicker the line, the more often the corresponding time interval has been used). In the figure some intervals are prominent for being very short: Subjects S13 and S14 in the stopwatch task, Subject S10 in the Ferrez-Millán task, and Subject S3 in the P300 speller task; the difference between the error and the correct responses seems not significant, and, accordingly, the classification performances are very low, even for the SVM classifier, which uses the whole epoch. The summary lines are consistent with the observations about the ErrP latencies made before: The significant intervals for the Eriksen task begin much earlier than those for the other tasks; moreover, it seems that ErrPs from the P300 speller are slightly later than those from the stopwatch and Ferrez-Millán tasks.

6.3 EOG Artifacts Handling

EOG artifacts may corrupt EEG data, and users may unconsciously blink or move their eyes in different ways depending on the stimulus and thus drive the BCI using their eyes instead of only their brains, especially healthy users. The problem of EOG-corrupted epochs is particularly important when the number of epochs is not very high or for the ErrPs in the spelling tasks, where the ErrP stimulation occurs after a period of intense visual stimulations: We had a user that initially tended to systematically blink after the end of the P300 stimulations in the spelling tasks, i.e., very close to the to ErrP stimulus. For this reason, we preferred to remove epochs with strong EOG activity altogether. The problem with this approach is that it cannot work online: When running online, a classifier must choose whether the given epoch contains an ErrP or it does not; the system may be programmed to take no action when the EOG channel goes above a given threshold, but this is equivalent to classify the corresponding epoch as not containing an ErrP.

In preparation for the online experiments, we studied the behavior of the classifiers described above when dealing with epochs contaminated by EOG activity. More precisely, we removed contaminated epochs from the training set, but left them in the test set. Table 6.7 shows the cross-validated results obtained when no epoch is removed from the test sets; epochs with noisy EOG were removed from the training sets. Apart from the handling of contaminated epochs, the training and validation procedure is the same used for the compilation of Table 6.5, so it is possible to make comparisons. It is easy to see that the performance on the full test set is very similar to the performance on the “clean” epochs; the only exception is Subject S7, which exhibited a poor performance on EOG-contaminated epochs.

There is another way to handle noisy epochs: They can be considered as not containing an ErrP by default. Table 6.8 shows the results of Table 6.5 corrected to take into account discarded epochs, which are treated as if they were classified as non-ErrP. By comparing these new number with those in Table 6.7, we can see that this approach does not seem to be a good idea; the only possible exception, again, is Subject S7, which exhibits a decrease in the recall of ErrPs, but also an increase in the recall of non-ErrPs. Whether these contrasting variations in

Subject		**Size**	**LDA**	**Bayes**	***k*-NN**	**P. LDA**
S1	ErrP	92	67%	65%	65%	67%
	N-ErrP	450	86%	74%	85%	84%
S3	ErrP	97	51%	48%	49%	54%
	N-ErrP	477	66%	59%	60%	68%
S4	ErrP	90	58%	68%	63%	70%
	N-ErrP	426	80%	80%	81%	83%
S5	ErrP	91	75%	76%	68%	78%
	N-ErrP	452	87%	76%	86%	88%
S7	ErrP	88	47%	65%	65%	60%
	N-ErrP	443	82%	65%	69%	82%

Table 6.7: 3-fold cross-validated results of ErrP detection in P300-speller tasks on the whole test set

Subject		**Size**	**LDA**	**Bayes**	***k*-NN**	**P. LDA**
S1	ErrP	92	66%	63%	63%	66%
	N-ErrP	450	89%	81%	90%	87%
S3	ErrP	97	31%	27%	29%	31%
	N-ErrP	477	80%	73%	77%	81%
S4	ErrP	90	54%	67%	59%	67%
	N-ErrP	426	83%	83%	84%	86%
S5	ErrP	91	71%	73%	66%	75%
	N-ErrP	452	87%	77%	86%	88%
S7	ErrP	88	25%	31%	30%	34%
	N-ErrP	443	90%	85%	89%	93%

Table 6.8: 3-fold cross-validated results of ErrP detection in P300-speller tasks with correction for removed epochs

Subject		**Train. Size**	**Online Performance**
B1	ErrP	84	23/35 (66%)
	N-ErrP	290	51/74 (69%)
B3	ErrP	65	38/65 (58%)
	N-ErrP	193	91/137 (66%)

Table 6.9: Results of the online ErrP classification. Training and test set size is the number of selections of each class. Performance is the fraction of correct classification.

Subject S7 compensate or make a difference is not possible to tell without a framework for judging the performance of the ErrP classifier and its impact on the underlying BCI, which is the topic of the next chapter.

6.4 Online Results

The experiments with the P300 speller reported in Section 5.5.1 made use also of ErrPs. We used one of the classifiers presented in Section 6.1.1: LDA applied to raw EEG samples (after band-passing and decimation). Classification was applied both offline and online during such experiments; of particular significance are the experiments made with the speller in *free mode*, i.e., when the ErrP classification influences the spelling and the user has a real-time feedback about the classifier decisions and performance. Such experiments were performed only by subject B1 and B3, as already explained in Section 5.5.1.

Results of the online experiments are shown in Table 6.9. The classifiers were tested in sessions different from those used for training, so they are really indicative of a possible online use. For both users the classification performance is well above chance level, but this is not enough to say whether ErrP detection has been useful for such users. Again, answering this question (and the more general question whether ErrP detection can improve the performance of BCI) requires a more in-depth analysis of the performance measuring in a BCI. The next chapter contains such analysis and also the answer to the question of whether ErrPs can improve the performance of a BCI and how much.

7 Measuring The Performance Of A BCI

Sam paid it out slowly, measuring it with his arms:
'Five, ten, twenty, thirty ells, more or less,' he said.
'Who'd have thought it!' Frodo exclaimed.
'Ah! Who would?' said Sam.
J. R. R. Tolkien – *The Lord Of The Rings*

Incorporating an automatic correction system based on ErrPs in a BCI may be a way to improve its performance. Yet, if the classifier that recognizes the ErrPs makes too many mistakes, it may make things worse instead. So, the work on ErrP detection in the previous chapter brought up the question when ErrP detection can improve the performance of a BCI, and this in turn brought up the question how to measure the performance of a BCI. This chapter details the problems we found with methods for performance measuring found in the literature, and how we tried to address them using an approach based on *utility*. This approach is then applied to the results of the previous chapters so as to answer the question when ErrP detection can improve the performance of a BCI, and how much.

7.1 Performance Measures

If ErrPs can be used to improve a BCI performance, a way is needed to measure such performance in order to assess any improvement. The performance of a BCI can be expressed in many different ways; classification accuracy, information transfer rate, letters or words per minute, kappa statistic, and others are used in the literature.

The problem is that there are many different BCI protocols, and their differences must be taken into account when comparing results for different experiments; it is not fair to compare the accuracy of a binary protocol, e.g., a BCI based on motor imagery of left/right hand, with the accuracy of a more complex protocol, like a motor-imagery BCI discriminating four movements or a P300 speller with a 36-symbol matrix. It is not only a problem of the number of possible choices in a selection, but also the time needed to operate different BCIs is different.

Classification accuracy, i.e., the fraction of examples correctly classified, is the simplest performance measure used in the BCI literature. Sometimes, the error rate is used instead, but it is equivalent to accuracy, as the error rate is the fraction of examples wrongly classified. They are both easy to compute and to understand, but have many shortcomings. They do not make any distinction between different kinds of errors, while different kinds of errors have different impact (cost) in general. Moreover, classes that appear less frequently in the data are weighted less in the accuracy computation, and this is likely to lead to biased classifiers and biased evaluations.

Another way to measure BCI performance is Cohen's Kappa coefficient [126]; the Kappa coefficient is a way to express the agreement between two classifications. This measure takes into account the different frequencies for classes and also how errors are distributed among classes, although its meaning is not so explicit as for accuracy.

Information transfer rate (ITR — sometimes simply called *bit rate*) seems to be the performance measure most used in the literature. It has many advantages: It does not depend on any particular protocol, it takes into account both the number of choices and time, it is strongly theoretically grounded, and it could be applied also to continuous ranges of choices [127].

There is a problem with information transfer rate, though: In many articles in the literature, no *real* transfer is measured, and instead a theoretical figure from information theory is used. In [128], the authors derive a formula to compute the (mean) number of bits transferred per trial:

$$B = \log_2 N + p \log_2 p + (1-p) \log_2 \frac{1-p}{N-1}\,, \qquad (7.1)$$

where N is the number of possible choices per trial, and p is the accuracy of the BCI, i.e., the probability that the BCI selects what the user intends. Equation (7.1) divided by the trial duration gives the mean number of bits transferred per time unit. This formula, as the authors say, is derived from Shannon's theory [129], and it represents the *mutual information* between the user's choice and the BCI selection, under the assumption that all choices convey the same amount of information (i.e., they are chosen by the user with equal probability), p is the same for all the possible choices, and that all the wrong choices have the same probability in case of error. In this view, a subject communicates by using a BCI that is a noisy channel, where the noise is introduced by the BCI every time it selects the wrong choice.

According to Shannon's noisy channel coding theorem [129], it is possible to achieve an arbitrarily small error probability in a communication on a noisy channel as long as the information transfer rate does not go beyond a certain limit: the *channel capacity*. The channel capacity is given by the mutual information[1], and this seems to justify the use of mutual information in Equation (7.1). The only problem is that Shannon proved his famous theorem by transferring information embedded in ever increasing blocks of bits, and in practice only very complex error correction schemes have permitted to get near Shannon's limit. For a BCI, where a human subject sits at one end of the noisy channel, it is not possible to implement such complex error correction schemes, and thus the limit given by the mutual information, as in Equation (7.1), represents a theoretical figure, unreachable by any real BCI whose error rate is significantly different from zero.

To better understand how far from practice can be the performance measure of Equation (7.1), let us focus on a simple example with a standard P300 speller, like the one already described in Section 3.2 but without ErrP detection, where there are 36 symbols. Let us suppose that the speller speed is 4 letters per minute, and that a user achieves a performance of 45% in accuracy, which is low, but still far better than random-level accuracy (2.7%). By substituting $p = 45\%$ and $N = 36$ in Equation (7.1), we get $B = 1.36\,\text{bits}$. The theoretical bit rate for this user would be $4 \cdot B = 5.4\,\text{bits/min}$; this is not very fast, but, still, communication should be possible.

Now let us look at the practical use of such a BCI: The most natural way would be to move on to the next letter when the speller gets one right, and to "hit" backspace every time the speller is wrong. What is the actual transfer rate for this way of using the speller? Since every letter is more likely to be wrong than not, and this happens to backspace as well, wrong letters pile up and the expected time to spell a letter correctly is infinite, and thus the answer is, on average, exactly $0\,\text{bits/min}$ (see also the derivation of Equation (7.14)). While it could be still possible to raise p above 0.5 by increasing the number of stimulations per letter and render the P300 speller of the example usable, the channel capacity measured by Equation (7.1) promises a performance far beyond what is attainable once the details of the BCI are taken into account.

There exists a generalization of formula (7.1) that makes use of the confusion matrix and allows each letter to have a different probability of occurrence and a different accuracy [130], but such formula has the same shortcoming of (7.1), i.e., it treats the BCI as a communication channel

[1] Actually, the channel capacity is the maximum mutual information over all probability distributions for the input symbols. The input symbol probability cannot always be controlled, especially in the BCI context, where it is not possible to use arbitrary coding systems.

with no reference to the way the channel is actually used; moreover, it has a huge number of parameters. In order to keep the exposition simpler, we limit the discussion to the simplified formula, but all our considerations can be easily extended to use the general formula.

7.2 Utility: An Alternative Performance Measure

Given the problems of the performance measures described above, we developed a different measure, more user-centered, based on the concept of *utility*. If the BCI is supposed to serve the user as a tool, the user should get a benefit when the BCI works correctly. We can formalize this utility concept by defining utility as the mean benefit over time for the user:

$$U = \mathrm{E}\left[\lim_{T\to\infty} \frac{\int_0^T b(t)dt}{T}\right] \tag{7.2}$$

where $b(t)$ is a *benefit function*, which assumes positive (or negative) values depending on whether the choice at time t conforms to (or contradicts) the user intention.

Let us consider the case of a discrete BCI, i.e., a BCI system with a finite number of outputs. In this situation, the benefit function will be discrete, and defined only in the time-instant t_k when an output is generated. In fact, a quantifiable benefit can be defined only when an output is selected (e.g., a letter printed by the speller, or a target reached on the screen). In mathematical terms we may write

$$b(t) = \sum_{k=1}^{K} b_k \delta(t - t_k)\,, \tag{7.3}$$

where K is the number of outputs in the interval $[0, T]$. Putting Equation (7.3) into Equation (7.2) and observing that $T = \sum_{k=1}^{K} \Delta t_k$, where $\Delta t_k = t_k - t_{k-1}$ for $k > 1$ and $\Delta t_1 = t_1$, we may write

$$\begin{aligned} U = \mathrm{E}\left[\lim_{T\to\infty} \frac{\sum_1^K \int_0^T b_k \delta(t - t_k)dt}{T}\right] \\ = \mathrm{E}\left[\lim_{K\to\infty} \frac{\sum_1^K b_k}{\sum_{k=1}^K \Delta t_k}\right]\,. \end{aligned} \tag{7.4}$$

For a stochastic variable n generated by an ergodic process the following relation holds

$$\lim_{K\to\infty} \frac{\sum_{k=1}^K n_k}{K} = \mathrm{E}[n]\,, \tag{7.5}$$

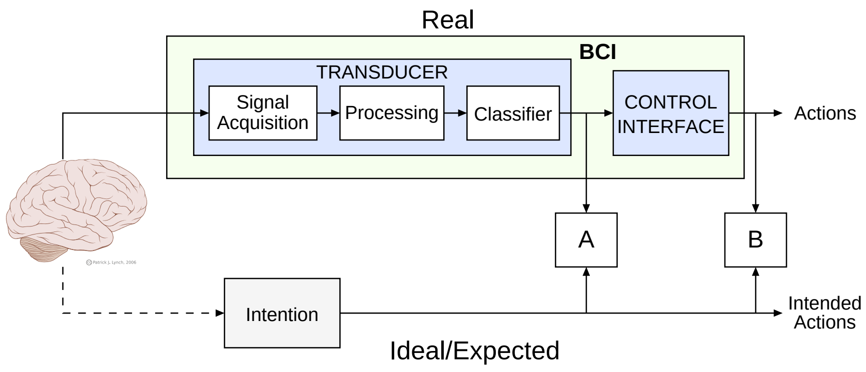


Figure 7.1: Different metrics measure performance at different points

which can be used to derive a formulation of utility valid for a discrete-BCI system:

$$U = \mathrm{E}\left[\frac{\mathrm{E}[b_k]}{\mathrm{E}[\Delta t_k]}\right] = \frac{\mathrm{E}[b_k]}{\mathrm{E}[\Delta t_k]}\,. \tag{7.6}$$

This formula is easily readable as the ratio between the average benefit (among all the possible discrete choices) and the average time needed to get it. If a BCI system is able to reach the desired target (maximum benefit) in a shorter interval of time, utility will be larger.

Figure 7.1 helps in pointing out a major difference between the utility concept and other measurements used in the literature. Our utility measures the performance of a BCI in terms of the satisfaction of the user intention, by comparing the action performed by the BCI (e.g., words spelled, commands issued to an external device) with the action intended by the user; this comparison is performed at the level of block 'B' in Figure 7.1. Conversely, the other performance measurements (such as the information transfer rate or the Kappa coefficient) compare the output of the classifier with the desired output (the comparison is performed at level of block 'A'), without taking into account the way this classification is used by the BCI system. As utility is application related, it is an elective measurement to compare different implementations of BCI applications (i.e., with or without a automatic correction system).

7.3 Utility And Bit Rate In A P300 Speller

Let us apply the concept of utility to the P300 speller. A benefit is given to the user whenever the speller writes a correct letter. A correct letter can be spelled in just one trial if the BCI classifier predicts the column and raw properly, or more than one if the classifier is mistaken, and the user has to try to erase the wrong letter first and then respell the desired letter. So, evaluating the utility as expressed in (7.6) for a P300 speller requires only to compute the expected time between two correct letters.

In order to keep the exposition simpler, we use the same assumptions of Equation (7.1); moreover, we assume that the accuracy of the speller p is constant and the system has no memory, i.e., each trial is not influenced by the result of the previous one. These assumptions permit also to compare the utility with the ITR easily.

If we call T_L the expected time needed to correctly spell a letter, and c is a constant expressing the duration of a single trial, i.e., the time to spell a letter either correctly or wrongly, then:

$$T_\mathrm{L} = c + (1-p)(T_\mathrm{B}^{(1)} + T_\mathrm{L}^{(1)})\,, \tag{7.7}$$

where $T_\mathrm{B}^{(1)}$ is the cost of a backspace, and $T_\mathrm{L}^{(1)}$ is the cost of respelling the letter after the backspace; in other words, if the letter is wrong, which happens with probability $1-p$, one has to take into account the time of a backspace followed by another letter, with the correction of any other error that may ensue. Similarly,

$$T_\mathrm{B}^{(1)} = c + (1-p)(T_\mathrm{B}^{(2)} + T_\mathrm{B}^{(3)}) \tag{7.8}$$
$$T_\mathrm{L}^{(1)} = c + (1-p)(T_\mathrm{B}^{(4)} + T_\mathrm{L}^{(2)})\,, \tag{7.9}$$

where superscripts are used to distinguish the costs of different occurrences of letters or backspaces; by substituting the (7.8) and (7.9) into the (7.7),

$$T_\mathrm{L} = c + 2(1-p)c + (1-p)^2(T_\mathrm{B}^{(2)} + T_\mathrm{B}^{(3)} + T_\mathrm{B}^{(4)} + T_\mathrm{L}^{(2)})\,. \tag{7.10}$$

By iterating the above steps, one gets

$$\begin{aligned} T_\mathrm{L} &= c + 2(1-p)c + 4(1-p)^2c + 8(1-p)^3c + \dots \\ &= c\sum_{i=0}^{\infty}(2-2p)^i\,. \end{aligned} \tag{7.11}$$

This is a series, and, if $2-2p<1$, it converges to its limit

$$T_\mathrm{L} = \frac{c}{2p-1}\,. \tag{7.12}$$

But when $p \leq 0.5$ the series does not converge, and the expected time to spell a letter correctly is indeed infinite, i.e., the information transferred is 0.

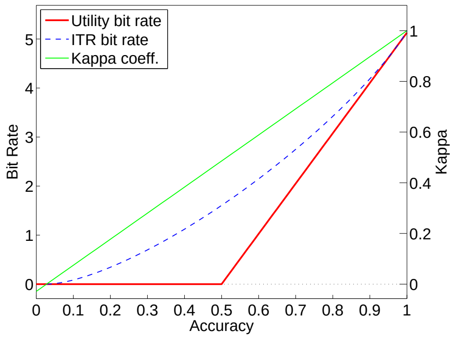


Figure 7.2: Comparison between utility, theoretical bit rate, and Kappa coefficient for a P300 speller with 36 symbols.

The utility of the P300 speller is simply the ratio between the average benefit b_{L} carried by any *correctly spelled* letter divided by the expected time T_{L} required to spell it:

$$U = \frac{b_{\mathrm{L}}}{T_{\mathrm{L}}}. \tag{7.13}$$

We could take 1 as a measure of the benefit of a letter, or we can use the information conveyed by a letter. Only $N-1$ symbols can appear in real words (the backspace cannot), and we are measuring the information contained in the spelled text; together with the assumption of equal probability, this permit to assert that the information conveyed by a spelled letter is $b_{\mathrm{L}} = log_2(N-1)$ bits. While we disregard the fact that in reality different letters appear with different frequencies, this is useful to compare Equation (7.13) with the theoretical bit rate (the ITR) based on Equation (7.1).

By substituting the values in Equation (7.13), we get

$$U = \frac{(2p-1)\log_2(N-1)}{c}, \tag{7.14}$$

while the theoretical bit rate is given by Equation (7.1) divided by c, the time of a trial:

$$I_{\mathrm{ITR}} = \frac{B}{c} = \frac{\log_2 N + p\log_2 p + (1-p)\log_2 \frac{1-p}{N-1}}{c}. \tag{7.15}$$

Figure 7.2 compares the two measures and shows how different they can be. The reason is that (7.15) measures the *capacity of a channel*, i.e., the maximum performance obtainable by a noisy channel, while (7.14) measures the expected performance of the same channel when information is conveyed in a specific way; in our case, this is the natural way of using a P300 speller. As expected, the latter curve lies always below the theoretical limit, and it is equal to zero when the accuracy is too low. For high accuracy values, the two curves almost coincide, although there is a small gap due to the presence of a *backspace* symbol, which is obviously never used if no error is committed. The bit-rate curves in Figure 7.2 are consistent with the plots in [131, Chapter 3], where a somewhat similar approach to measuring a BCI performance is developed.

It is worth noting that the graph shows regions where the channel is useless for a P300 speller (when $p \leq 0.5$ in our case) and also areas where the speed of the speller is very far from the theoretical limit. The reported comparison should also warn us against applying Equation (7.15) blindly, because it may provide very unrealistic scores when accuracy is not very high.

Figure 7.2 shows also the value of the Kappa coefficient versus accuracy. It is apparent that also Kappa is not a good predictor of the usefulness of a BCI.

The problems pointed out above are not meant as a critique against the use of mutual information *per se*, because mutual information *is* useful as a mean to compare different BCIs, as already explained. But this comparison gives only a rough indication on relative performances, and for the evaluation of the gain of using ErrP detection we need something more precise. Therefore, we developed a different approach to performance measuring based on the actual task.

7.4 Impact Of ErrP Detection In A P300-Speller

Using the same techniques employed in the last section, we can compute the utility of a P300 speller that makes use of ErrPs for automatic error correction, and come closer to answer the question about the improvement brought by ErrPs. As explained in Section 3.2, such a speller selects a letter by means of P300 detection, displays the letter on the screen, and if it detects an ErrP, it cancels the last selected letter.

While the performance of a P300 speller can be expressed as a function of one parameter p that conveys the goodness of the classification, the performance of an ErrP classifier needs two parameters, because the accuracies for error and correct trials could be different.

As it will be apparent in the following, using recalls of the two classes (error and correct) simplifies the formulas. So, let us define r_{E} as the recall for errors, i.e., the fraction of times that an actual error is recognized by the ErrP classifier, and r_{C} as the recall for correct trials, i.e., the fraction of times that a correctly spelled letter is recognized by the ErrP classifier. Along the line of the previous assumptions, we assume that r_{E} and r_{C} are constant and do not depend on the actual letter.

The utility of the speller is still given by Equation (7.13), but T_{L} is to be computed according to the functioning of the new speller. There are four possible cases:

1. The P300 speller selects the correct letter, and the ErrP classifier correctly recognizes it. This happens with probability $p_1 = p\, r_{\mathrm{C}}$.

2. The P300 speller selects a wrong letter, and the ErrP classifier does not recognize the error. This happens with probability $p_2 = (1-p)\,(1-r_{\mathrm{E}})$, and the user has to "spell" a backspace and then the letter again.

3. The P300 speller selects the correct letter, and the ErrP classifier wrongly detects an error. This happens with probability $p_3 = p\,(1-r_{\mathrm{C}})$, and the user has to respell the letter (which has been canceled by the error detection system).

4. The P300 speller selects a wrong letter, and the ErrP classifier recognizes the error. This happens with probability $p_4 = (1-p)\, r_{\mathrm{E}}$, and the user has to respell the letter; the wrong letter is canceled by the system.

As before, let us call T_{L} the expected time to spell a letter correctly, and T_{B} the expected time to effectively issue a backspace command; both times take into account the correction of any error, i.e., the removal of possible misspelled letters. The expected time to spell a letter correctly can be computed by weighting the time for each of the four possible cases above with their probabilities.

$$\begin{aligned} T_{\mathrm{L}} = p_1 \cdot c + p_2 \cdot (c + T_{\mathrm{B}} + T_{\mathrm{L}}) + p_3 \cdot (c + T_{\mathrm{L}}) + p_4 \cdot (c + T_{\mathrm{L}}) \\ = p\, r_{\mathrm{C}} \cdot c + (1-p)\,(1-r_{\mathrm{E}}) \cdot (c + T_{\mathrm{B}} + T_{\mathrm{L}}) \\ + p\,(1-r_{\mathrm{C}}) \cdot (c + T_{\mathrm{L}}) + (1-p)\, r_{\mathrm{E}} \cdot (c + T_{\mathrm{L}})\,, \end{aligned} \tag{7.16}$$

where c is the constant duration of a trial, as before. In this case c also includes the time for the feedback used by the ErrP classifier, but, given the small latency of ErrP and the fact that any P300 speller must give a feedback so as the user can decide the next letter to spell, the effect on c would be negligible, if any.

A formula for T_{B} can be derived in the same way, by considering the four cases:

1. The P300 speller selects the backspace, and the ErrP classifier correctly recognizes it. This happens with probability $p_1 = p\, r_\mathrm{C}$.

2. The P300 speller selects a letter, and the ErrP classifier does not recognize the error. This happens with probability $p_2 = (1-p)\,(1-r_\mathrm{E})$, and the user has to issue the backspace command twice.

3. The P300 speller selects the backspace, and the ErrP classifier wrongly detects an error. This happens with probability $p_3 = p\,(1 - r_\mathrm{C})$, and the user has to reissue the backspace command (which has been canceled by the error detection).

4. The P300 speller selects a letter, and the ErrP classifier recognizes the error. This happens with probability $p_4 = (1-p)\, r_\mathrm{E}$, and the user has to reissue the command; the last wrong letter is canceled by the system.

Again, by weighting time with probability:

$$\begin{aligned} T_\mathrm{B} = p\, r_\mathrm{C} \cdot c + (1-p)\,(1-r_\mathrm{E}) \cdot (c + 2T_\mathrm{B}) \\ + p\,(1-r_\mathrm{C}) \cdot (c + T_\mathrm{B}) + (1-p)\, r_\mathrm{E} \cdot (c + T_\mathrm{B})\,. \end{aligned} \tag{7.17}$$

By subtracting Equations (7.16) and (7.17) from each other (only one term on the right-hand sides differs), it is possible to derive the equality

$$T_\mathrm{B} = T_\mathrm{L}\,. \tag{7.18}$$

Solving either equation leads to

$$T_\mathrm{L} = T_\mathrm{B} = \frac{c}{p\, r_\mathrm{C} + (1-p)\, r_\mathrm{E} + p - 1}\,. \tag{7.19}$$

This formula represents a limit of a series, the sum of ever increasing sequence of letters and backspace with ever smaller probability; the limit exists only if the denominator in Equation (7.19) is positive, i.e., when

$$r_\mathrm{C} > \frac{1-p}{p}\,(1-r_\mathrm{E})\,. \tag{7.20}$$

Figure 7.3 shows the boundaries defined by Inequality (7.20) for different values of p; the inequality is satisfied for the points lying above the lines. The time for spelling a letter is finite, i.e., the P300 speller can be useful, only for the values of r_E and r_C satisfying Inequality (7.20). It can be noticed that the constraint becomes tighter as p diminishes, with recall of ErrPs becoming ever more important.

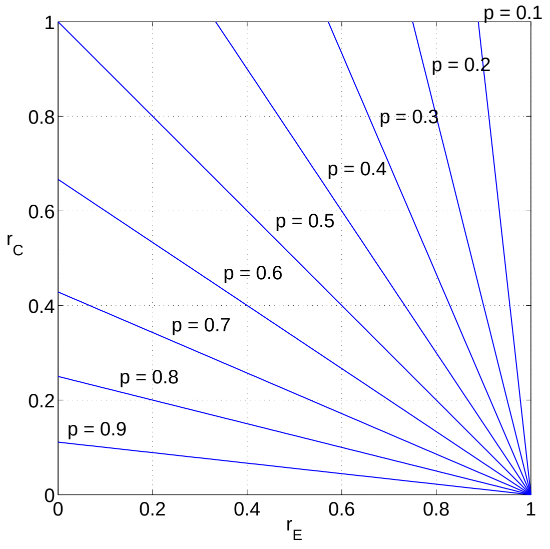


Figure 7.3: Condition for usability of a P300 speller with ErrP detection

Inequality (7.20) can be written also as

$$p\,r_{\mathrm{C}} > (1-p)\,(1-r_{\mathrm{E}})\,, \tag{7.21}$$

where the two sides are the probabilities p_1 and p_2 (defined at Page 131); in other words, the speller can be used as long as the number of correct selections surpasses the number of wrong letters. The number of letters canceled by the error detection system affects the speed of the speller, but it does not affect the fact that the right letter is spelled eventually.

We can now compute the utility for the P300 speller with automatic error correction. As before, we use the information conveyed by a spelled letter, $b_{\mathrm{L}} = log_2(N-1)$ bits, for the benefit; by substituting b_{L} and the value for T_{L} given by Equation (7.19) in Equation (7.13), we get:

$$U = \frac{(p\,r_{\mathrm{C}} + (1-p)\,r_{\mathrm{E}} + p - 1)\,\log_2(N-1)}{c}\,. \tag{7.22}$$

Now we can answer to the question: When does ErrP detection give any improvement to the P300 speller? The answer can be found by comparing Equations (7.14) and (7.22), or — equivalently — Equations (7.12) and (7.19).

A first observation is that, as expected, Equation (7.12) can be seen as a particular case of Equation (7.19) with $r_{\mathrm{C}} = 1$ and $r_{\mathrm{E}} = 0$, i.e., no error is ever corrected.

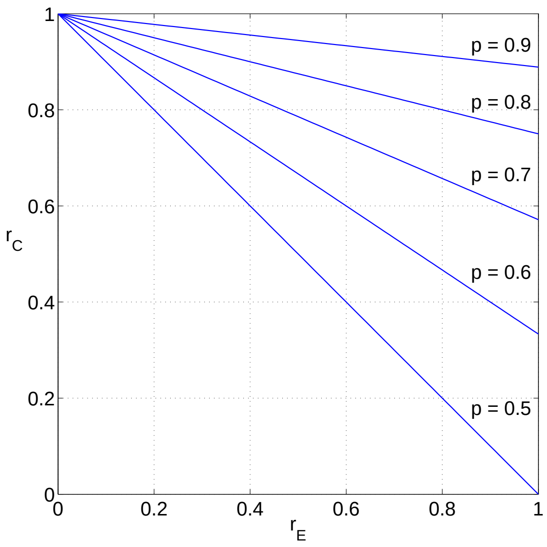


Figure 7.4: Comparison between two P300 spellers with and without ErrP detection

For $p \leq 0.5$ Equation (7.12) has no sense, but, as shown in Figure 7.3, it is possible to operate a P300 speller even with such a high error rate as long as the error detection is sufficiently accurate. Actually, it could be argued that this is an unlikely scenario, as P300 detection can be done after many repetitions, while ErrP must be detected in a single sweep; yet, this is part of the whole picture.

When $p > 0.5$, Equations (7.12) and (7.19) must be compared directly. In order to have any improvement,

$$p\, r_{\mathrm{C}} + (1-p)\, r_{\mathrm{E}} + p - 1 > 2p - 1\,, \tag{7.23}$$

i.e.,

$$r_{\mathrm{C}} > \frac{p-1}{p}\, r_{\mathrm{E}} + 1\,. \tag{7.24}$$

Figure 7.4 shows the boundaries defined by Inequality (7.24) for different values of p (for $p < 0.5$ the comparison has no sense); points above the lines represent values of r_{C} and r_{E} for which ErrP detection is advantageous. In this case, as p grows the area defined by Inequality (7.24) shrinks; this happens, because as p grows the performance of the "vanilla" P300 speller gets better and better, and it becomes harder and harder for the ErrP classifier to improve the speller performance.

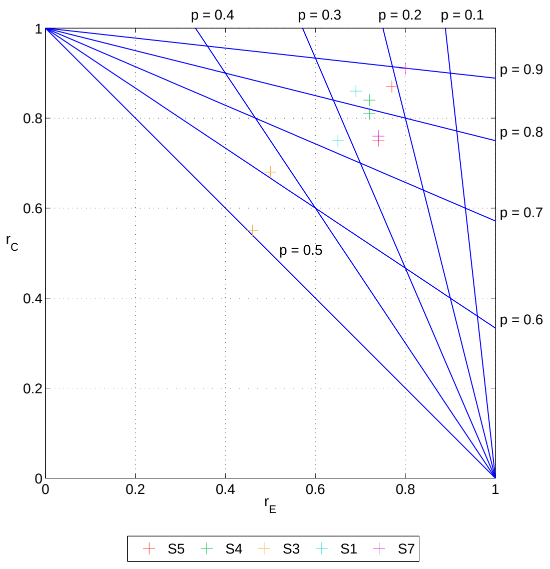


Figure 7.5: When ErrP detection improves the performance of a P300 speller

An alternative form for Inequality (7.24) is

$$p\, r_{\mathrm{C}} + (1 - p)\, r_{\mathrm{E}} > p\,, \tag{7.25}$$

where the left hand side is equal to $p_1 + p_4$ (defined at Page 131); this means that ErrP detection is advantageous when the overall accuracy of the P300 and ErrP system is better than the accuracy of the sole P300 system.

Figure 7.5 summarizes both Figures 7.3 and 7.4, and shows the values of r_{C} and r_{E} for which ErrP detection is advantageous for the whole range of p. As before, the part of the plane above the lines is the useful part; values below the lines are either useless or counterproductive. Figure 7.5 can be used as a guide to decide to bias the ErrP classifier either toward correct or erroneous epochs, depending on the value of p.

Figure 7.5 shows also where the results of the ErrP classifiers in Table 6.5 lie; for each subject, only the best and the worst results (in term of performance improvement) are shown. Subject S3 can hardly get any improvement, but the other four subjects can take advantage of ErrP detection even if their accuracy for the P300 speller is as high as 70–80%. If we consider the performance of the genetic algorithm for P300 detection shown in Table 5.27 (Page 95) as indicative, Subject S1 is not helped by ErrP detection, because of the high performance of the P300 classifier (about 85%). Subjects S3, S4, S5, and S7, which scored

Subj.	P300		LDA		Bayes		k-NN		P. LDA	
	Acc.	Ut.	Ut.	Ratio	Ut.	Ratio	Ut.	Ratio	Ut.	Ratio
S1	87%	15	14	0.95	12	0.81	14	0.94	14	0.93
	80%	12	13	1.04	11	0.87	13	1.02	12	1.01
S3	56%	2.5	3.2	1.28	2.1	0.85	2.3	0.93	3.7	1.49
	80%	12	8.8	0.72	7.6	0.61	7.8	0.63	9.3	0.75
S4	64%	5.7	7.4	1.29	8.1	1.42	7.9	1.38	8.7	1.51
	80%	12	11	0.93	12	0.96	12	0.96	12	1.01
S5	42%	0	4.5	∞	3.7	∞	3.6	∞	5.0	∞
	80%	12	13	1.08	11	0.93	13	1.04	14	1.10
S7	37%	0	0	—	0.41	∞	0.71	∞	1.1	∞
	80%	12	11	0.92	9.2	0.75	9.9	0.80	12	0.96

Table 7.1: Performance for an integrated P300+ErrP speller when noisy epochs are classified

about 56%, 64%, 42%, and 37% with the P300 speller respectively, can be helped (tough the margin for Subject S7 is tiny). It is interesting to notice that Subjects S5 and S7 fall in the "unlikely" scenario mentioned above, where the P300 speller accuracy is below 50%.

The gain of introducing ErrPs in a P300 speller can be computed as the ratio of the utilities given by Equations (7.22) and (7.14):

$$g = \frac{p\,r_{\mathrm{C}} + (1-p)\,r_{\mathrm{E}} + p - 1}{(2p-1)}\,, \tag{7.26}$$

subject to the constraints that both the numerator and the denominator are positive, i.e., Inequality (7.20) and $p > 0.5$. If only the denominator is negative, it means that the P300 speller cannot work without ErrPs, and hence g should be considered infinite. If both the numerator and the denominator are negative, it means that the P300 speller cannot work, with or without ErrPs, and hence g is indefinite. If only the numerator is negative, it means that introducing ErrPs renders the speller unusable, and hence $g = 0$.

7.4.1 Performance In The P300 Speller Experiments

Now we can compute the utility of the P300 speller with automatic error correction and the gain with respect to a plain P300 speller, a question we left open in the previous chapter. In order to do this, we take the recall figures for errors of Table 6.7, where no epoch is discarded from the test sets, and we use them together with the accuracy figures of Table 5.27 obtained with the GA for the P300 speller. By substituting those numbers in Equations (7.14), (7.22), and (7.26), we

Subj.	**P300 Acc.**	**P300 Ut.**	**LDA Ut.**	**LDA Ratio**	**Bayes Ut.**	**Bayes Ratio**	***k*-NN Ut.**	***k*-NN Ratio**	**P. LDA Ut.**	**P. LDA Ratio**
S1	87%	15	15	0.99	13	0.89	15	0.99	15	0.96
	80%	12	13	1.07	12	0.96	13	1.08	13	1.05
S3	56%	2.5	3.0	1.20	1.8	0.73	2.4	0.99	3.1	1.25
	80%	12	10	0.84	9.0	0.73	9.7	0.79	10	0.85
S4	64%	5.7	7.5	1.31	8.5	1.47	8.0	1.39	8.9	1.54
	80%	12	12	0.95	12	1.00	12	0.98	13	1.04
S5	42%	0	4.0	∞	3.4	∞	3.4	∞	4.6	∞
	80%	12	13	1.06	12	0.94	13	1.03	13	1.09
S7	37%	0	0	—	0	—	0	—	0	—
	80%	12	12	0.95	11	0.90	12	0.95	13	1.02

Table 7.2: Performance for an integrated P300+ErrP speller when noisy epochs are considered not to contain an ErrP

obtain Table 7.1; we used $N = 36$ and $c = .25\,\text{min}$ to have the utility measured in bits per minute. The table shows, for every subject, the accuracy of the P300 speller obtained by the GA, the utility of the P300 speller without automatic error correction, and then, for each of the classifiers trained on ErrP data, the utility of the speller with error correction and the ratio between the utilities. Ratios greater than 1 indicate that error correction improves the performance. The second line for each subject is obtained with an accuracy of the "basic" P300 speller is 80%, the value used during the recording sessions to generate some errors. The table confirms what we said before commenting Figure 7.5. Subject S1 is not helped at all by error correction; Subjects S3 and S4 can get an improvement of about 50% with the LDA classifier applied to polynomial coefficients; Subjects S5 and S7, who suffered from low accuracy of the P300 speller, could use the speller with error correction. The numbers in the table tell us more, tough: For example, the utility for the last subject in Table 7.2 is so low (it is equivalent to about 1 letter every 5 minutes) that we can say that the subject have no control of the BCI even with automatic error correction; and also the performance of Subject S3 is very low, less than a letter per minute.

Table 7.2 is similar to Table 7.1, but performance is computed by using the recall values from Table 6.8, i.e., by considering epochs contaminated by EOG noise as not containing an ErrP. The performance values are not very different from those in the previous table, with two notable exceptions: a reduced gain for Subject S3 when using the LDA classifier applied to polynomial coefficients, and the impossibility to use the P300 speller even with error correction for Subject S7. In most cases it seems that both ways of handling epochs contaminated by EOG noise are equally good in most of the cases, but the idea of trying to classify noisy epochs anyway seems to work better overall. This means that thee classifiers are able to extract some meaningful information even from epochs affected by artifacts.

As a final comment on Tables 7.1 and 7.2, we can say that for ErrP detection is very important to have a high accuracy, more important than for P300 detection, as ErrP stimulations are not repeated. There are different ways to improve the accuracy: Work can be done to devise better signal processing algorithm and classifiers, or new presentation interfaces and protocol could be introduced that better capture the subject attention and hence enhance the signal quality. Either way, the utility metric can be used to assess the performance improvement and thus to guide the development of algorithms and interface.

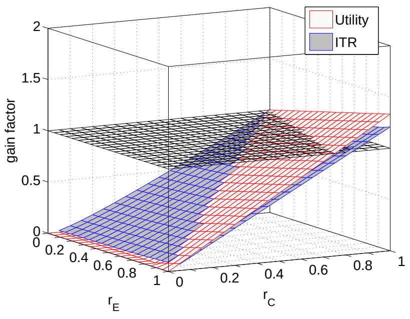
7.4.2 Comparison With The Theoretical Bit Rate

It is interesting to compare the gain in adding ErrP detection computed with the approach based on utility and with an approach based on the theoretical bit rate. The information theoretical gain of introducing ErrPs can be computed as in [72], where the authors compare the information per trial given by (7.1) with the following formula, which they derived:

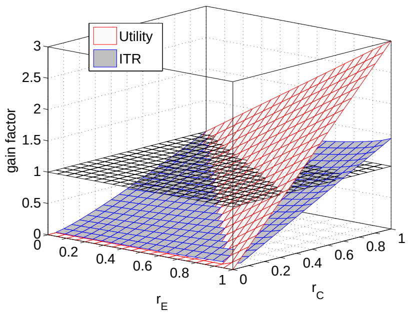
$$B = p_{\mathrm{t}} \cdot \left(\log_2 N + p' \log_2 p' + (1 - p') \log_2 \frac{1 - p'}{N - 1} \right) , \tag{7.27}$$

where $p_{\mathrm{t}} = p\, r_{\mathrm{C}} + (1 - p)\,(1 - r_{\mathrm{E}})$ and $p' = p\, r_{\mathrm{E}} / p_{\mathrm{t}}$. In other words, they derive the new accuracy p' for the system after discarding outcomes rejected by the ErrP detection, and use it in (7.1); the factor p_{t} takes into account the fact that discarded outcomes do not contribute to any information transfer.

Figure 7.6 shows the performance gain factor obtained by applying the theoretical bit rate approach (darker surface with blue grid) and our utility-based approach (lighter surface with red grid). The two surfaces show the ratios between the bit rate or the utility of a P300 speller with and without ErrP detection as a function of the recalls r_{C} and r_{E}, for the cases where the accuracy of the P300-based classifier is $p = 0.6$ or $p = 0.8$, and there are $N = 36$ symbols in the matrix. The regions of the two surfaces that represent gains greater than 1, i.e., the regions in the r_{C},r_{E} plane corresponding to points for which ErrP detection is advantageous, and the magnitude of the gain can be very different. The difference between the two approach is small for high values of p, and it grows as the value of p gets smaller. The approach based on the



(a) $p = 80\%$



(b) $p = 60\%$

Figure 7.6: Comparison of performance improvement ratio in a P300 speller ($N = 36$ symbols) computed according to utility and theoretical bit rate, for different accuracies of the P300 speller p

theoretical bit rate seems to underestimate the contribution of ErrPs; this is due to its tendency to underestimate the cost of errors, as shown in Figure 7.2.

If we consider a slightly different scenario involving a binary BCI, things are different, though. Let us consider the case where the two commands of the binary BCI cancel each other, i.e., if the BCI recognizes the wrong command, the user can undo it by selecting the other command; e.g, the BCI drives a cursor in one dimension. We can measure the utility of such a BCI by considering the time needed to reach a given result, e.g., a position for a cursor. The time needed for a cursor to reach one of two targets that lie on a line n step from the starting position, and where the cursor moves with probability p in the direction intended by the user, can be computed by modeling the task as a 1-D random walk where p is the probability of moving in a given direction (let us say right). For n and p sufficiently high the walk ends on the right-hand target with probability 1, i.e., it ends on the left-hand target with probability 0. So, we can simplify the model by considering a walk that is unbounded on the left side.

It is interesting to notice that also the P300 speller can be modeled as 1-D random walk bounded only in one direction; if consider the number of letters to spell plus the number of letter to cancel with backspace as the distance from the target, writing n letters with a P300 speller exhibits the same behavior (in terms of time and probability distribution) as a 1-D random walk to a target n steps afar: Every correct letter that is spelled is a step toward the goal; every wrong

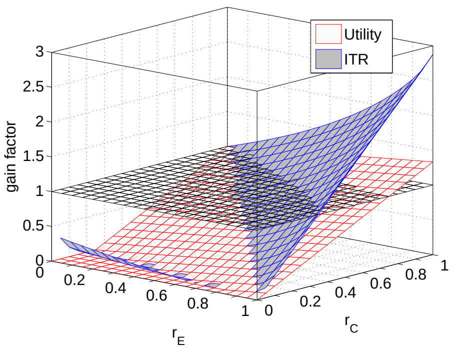


Figure 7.7: Comparison of performance improvement in a binary BCI (p=80%) due to ErrPs obtained according to different formulas

letter is a step back, which must be undone by subsequently selecting backspace. For this reason, we can use the equations found for the P300 speller to compute the expected time needed to move the cursor by one step in the right direction: Equation (7.12) when no error correction is used, and Equation (7.19) when error correction is present. In order to reach a position n steps farther, the time must be multiplied by n; the exact value of n is not important, as we are interested only in ratio of the two time lengths, not in their values. So the time to reach the target is

$$T = n\frac{c}{2p-1} \tag{7.28}$$

when no automatic error correction is used, and

$$T^{\star} = n\frac{c}{p\,r_{\mathrm{C}} + (1-p)\,r_{\mathrm{E}} + p - 1} \tag{7.29}$$

with error correction.

The ratio between Equations 7.29 and 7.28 can be used to predict the performance improvement when ErrPs are used, in terms of expected time to complete the task. While we can use the ratio between Equations 7.27 and 7.1 to compute the gain according to the theoretical bit rate measure. Figure 7.7 shows the ratios between the performances with and without ErrP detection for a binary BCI controlling a 1-D cursor, where the accuracy of the base classifier is $p = 0.8$ or $p = 0.6$, computed both with the channel-capacity approach (darker surface with blue grid) and our utility-based approach (lighter surface with red grid). The situation is the opposite of Figure 7.6: Here the channel-capacity approach overestimates the impact of automatic error detection.

7.5 Application Of Utility To More Complex BCIs

Sometimes a BCI entails a more complex interaction, where users reach their final goal after more than one step, e.g., hierarchical menus. Computing the utility in such cases is more complicated than the examples shown so far, because the influence on the utility of right and wrong selections at different steps is different. Yet, the example in this section shows that it is possible and rather easy to compute utility even in such cases.

There are two ways to do it: simulation and modeling. Simulation means taking a Monte Carlo approach (where a random number generator is used to simulate the repetition of many experiments, according to given probability distributions for different events, and averages are computed), while modeling means computing the relevant expected result from a mathematical model, as we did in the previous sections. In this section we will use a Markov model, which is general enough to be applied to virtually all interesting cases of discrete BCIs.

The interface chosen as an example is based on the P300 and presents the user with four choices (we label them with the first four letters 'A', 'B', 'C', and 'D'), which are highlighted one by one in random order. An epoch is associated with each highlighting event, and each epoch is classified as target or non-target independently of the others. When one of the choices has been highlighted and recognized as eliciting a P300 for three times, the choice is considered confirmed, and the corresponding action is carried on. The execution of an action concludes a single run, and then a new run begins, with all counters reset.

For this application it makes sense to assign a negative utility to wrong selections and a positive utility to right selections; the values may differ among different selections, according to the importance of selections, or the adverse effects of selecting a particular item at the wrong time. We can expand the utility given in Equation (7.6):

$$U = \frac{\sum_{x,y \in S} \mathrm{P}\left(x,y\right) c_{xy}}{\sum_{x,y \in S} \mathrm{P}\left(x,y\right) t_{xy}}\,, \tag{7.30}$$

where S is the set of all possible choices, $\mathrm{P}\left(x,y\right)$ is the probability of the interface selecting the choice y when the user intended to select x, c_{xy} is the benefit of the interface selecting the choice y when the user intended to select x, and t_{xy} is the expected time the interface takes to select the choice y when the user intended to select x.

To simplify the exposition, we assume that any correct choice has a utility of +1, while a wrong choice has a utility of −1, i.e.,

$$c_{xy} = \begin{cases} -1 & \text{if } x \neq y \\ +1 & \text{if } x = y \end{cases} \tag{7.31}$$

We assume also that all the possible choices are selected by the user with equal probability, i.e., $\mathrm{P}(x) = \frac{1}{4}$ for every x. Under these assumptions the interface is symmetric with respect to the choice desired by the user, so we can use the model to find the expected utility for one choice (e.g., x='A') and substitute the result for all the other choices. So, the (7.30) becomes:

$$\begin{aligned} U = & \frac{\sum_{x\in S} \mathrm{P}(x) \sum_{y\in S} \mathrm{P}(y|x)\, c_{xy}}{\sum_{x\in S} \mathrm{P}(x) \sum_{y\in S} \mathrm{P}(y|x)\, t_{xy}} \\ & = \frac{4 \times \frac{1}{4} \sum_{y\in S} \mathrm{P}(y|x=\mathrm{A})\, c_{\mathrm{A}y}}{4 \times \frac{1}{4} \sum_{y\in S} \mathrm{P}(y|x=\mathrm{A})\, t_{\mathrm{A}y}} \\ & = \frac{\sum_{y\in S} \mathrm{P}(y|x=\mathrm{A})\, c_{\mathrm{A}y}}{\sum_{y\in S} \mathrm{P}(y|x=\mathrm{A})\, t_{\mathrm{A}y}} = \frac{\sum_{y\in S} \mathrm{P}(y|x=\mathrm{A})\, c_{\mathrm{A}y}}{t_\mathrm{A}}, \end{aligned} \tag{7.32}$$

where we define $t_\mathrm{A} = \sum_{y\in S} \mathrm{P}(y|x=\mathrm{A})\, t_{\mathrm{A}y}$.

In order to evaluate Equation (7.32), we need to compute $\mathrm{P}(y|x=\mathrm{A})$ and t_A. We can do this by modeling the application as a Markov chain [132]. The state of the chain represents the number of times each choice has been selected. Every possible state can be identified by a 4-tuple of numbers of selections, a number for each possible choice: $(n_\mathrm{A}, n_\mathrm{B}, n_\mathrm{C}, n_\mathrm{D})$, $0 \leq n_\mathrm{X} < 3$. The states where a choice has been selected three times, i.e, the states corresponding to an actual selection of a choice, are modeled as four absorbing states, because reaching any of these states terminates a run; for absorbing states, the number of selections for all the choices is not important, the only thing that counts is which choice has reached 3 selections. All other states are transient. The initial state is the state $(0,0,0,0)$, where all selections are zero.

We can express transition probabilities as a function of the recall for targets, r_T, and the recall for non-targets, r_N. If the intended selection is 'A', the probability of transitioning from state $(n_\mathrm{A}, n_\mathrm{B}, n_\mathrm{C}, n_\mathrm{D})$ to state $(n_\mathrm{A}+1, n_\mathrm{B}, n_\mathrm{C}, n_\mathrm{D})$ is equal to $\frac{1}{4}r_\mathrm{T}$; the probability of transitioning to states $(n_\mathrm{A}, n_\mathrm{B}+1, n_\mathrm{C}, n_\mathrm{D})$, $(n_\mathrm{A}, n_\mathrm{B}, n_\mathrm{C}+1, n_\mathrm{D})$, and $(n_\mathrm{A}, n_\mathrm{B}, n_\mathrm{C}, n_\mathrm{D}+1)$ is $\frac{1}{4}(1-r_\mathrm{N})$; finally, the probability of remaining in the same state is $\frac{1}{4}(1-r_\mathrm{T}) + \frac{3}{4}r_\mathrm{N}$.

In our model, $\mathrm{P}(y|A)$ and $t_{\mathrm{A}y}$ represent the probability of ending in the absorbing states y and the expected time before that happens, respectively. There are formulas to compute them for an absorbing Markov chain [133]. If P is the transition matrix, when there are absorbing states, the matrix can be partitioned in this fashion:

$$P = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix} , \tag{7.33}$$

where Q is the transition matrix for transient states, and R is the transition matrix from transient to absorbing states. The average time spent in the transient state j when starting from the state i before reaching any absorbing state is given by the element i,j of the matrix $T = (I - Q)^{-1}$. The probability of ending in the absorbing state j when starting from the state i is given by the element i,j of the matrix $G = (I - Q)^{-1}R$.

We can now compute all the elements in Equation (7.32). t_A is the expected time before reaching any absorbing state; this can be computed as the sum of the expected time spent in all transient states when starting from the initial state. Let us say that the initial state is the number 1 in the transition matrix, then

$$t_\mathrm{A} = \sum_j T_{1j} . \tag{7.34}$$

$\mathrm{P}\left(y|x = \mathrm{A}\right)$ is simply an element in the matrix G:

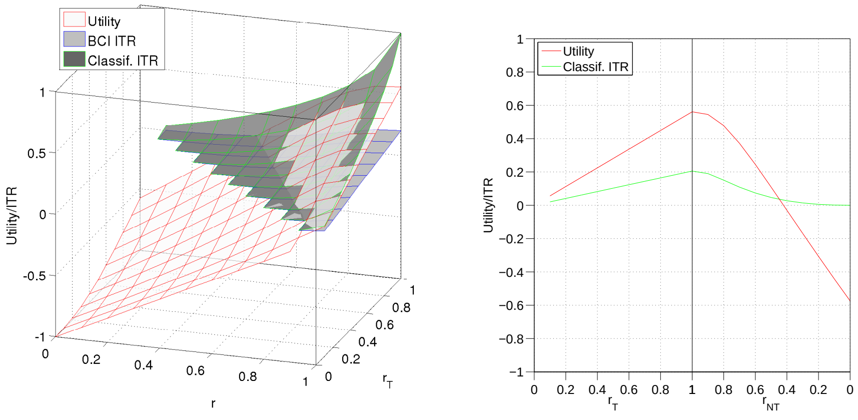
$$\mathrm{P}\left(y|x = \mathrm{A}\right) = G_{1y} . \tag{7.35}$$

Given the symmetry assumptions above, G_{1y} can only assume two different values, depending on whether y is the target (intended) choice or not. Let us call those values $g_\mathrm{T} = G_{1\mathrm{A}}$ when y is target, and $g_\mathrm{NT} = G_{1\mathrm{B}} = G_{1\mathrm{C}} = G_{1\mathrm{D}}$ when it is not.

Putting everything together, we get

$$U = \frac{\sum_{y \in S} \mathrm{P}\left(y|x = \mathrm{A}\right) c_{\mathrm{A}y}}{t_\mathrm{A}} \\ = \frac{\sum_{y \in S} G_{1y} c_{xy}}{\sum_j T_{1j}} = \frac{g_\mathrm{T} - 3g_\mathrm{NT}}{\sum_j T_{1j}} . \tag{7.36}$$

The Markov chain has 85 states, 4 absorbing states plus $3^4 = 81$ transient states (where each of the four choices may have been selected 0, 1, or 2 times); hence, its transition matrix has $85^2 = 7225$ elements, and, although it is sparse, doing symbolic computation on such a big matrix is cumbersome. Yet, it is quite easy to numerically compute



(a) (b)

Figure 7.8: Comparison between utility and ITR to measure performance in a BCI

the elements of P, and hence T, given the values of r_T and r_N, and we can study how the utility varies as r_T and r_N vary by evaluating the utility for a few values of r_T and r_N, and interpolating for intermediate values. This can be used for example to decide to bias the classifier toward targets or non-targets so as to improve the expected utility.

We can compare the utility with the theoretical bit rate (the channel capacity) of the P300 classifier. The theoretical bit rate for the classifier can be computed by applying the definition of mutual information to the confusion matrix; for a classifier with recalls r_T and r_N, as defined above, and probability of target presentation equal to $1/N$, the confusion matrix is

$$\begin{pmatrix} \frac{r_\mathrm{T}}{N} & \frac{1-r_\mathrm{T}}{N} \\ \frac{(1-r_\mathrm{N})(N-1)}{N} & \frac{r_\mathrm{N}(N-1)}{N} \end{pmatrix}. \tag{7.37}$$

Figure 7.8.a shows how utility and theoretical bit rate vary with r_T and r_N (theoretical bit rate has not been plotted when the classifier is worse than random). Although — not surprisingly — they both grow as r_T and r_N grow, the two measures show a different bias toward non-target recall. This is more clear in Figure 7.8.b, which shows the two measures when $r_\mathrm{T} = 1$ or $r_\mathrm{N} = 1$ for different values of the other recall. Low values of r_N penalize utility much more than the channel capacity, because utility takes into account the asymmetry between targets and non-targets: A non-target classified wrongly may cause damage, while misclassified target only slows the interface. So, utility gives better information about which is the best classifier when more than one is available.

Monte Carlo simulations, where 1 million runs have been simulated for different values of r_T and r_N, agree with the results of the Markov model to the third or fourth significant digit. The results of the simulations are not reported, as they have been used only to confirm the predictions of the Markov model.

7.6 Utility and Comparisons

In this chapter the utility of different BCIs has been derived. In all the cases, the formulas and the computations have been done by putting the performance of a classifier (for P300s, ErrPs, or motor imagery) in a model. This is a very powerful characteristic of the utility idea: It is possible to evaluate different classifiers on the same interface by simply plugging the classifier accuracy figure in a suitable model of the interface. An advantage of this approach is that it is possible to give a realistic estimation of the performance of a BCI, and also predict the performance of many different BCIs when they use a given classifier. In this way it is possible to give a fairer evaluation of different classifiers, an idea very similar to the one proposed in [134].

So, the concept of utility can be used for two different purposes: A first one is the one showed in this chapter, i.e., the optimization of a particular BCI, by computing the expected performance for different values of tuning parameters (in the cases shown above the introduction or not of an ErrP-based correction mechanism). A second purpose is the comparison of different classifiers, by comparing the utility of one or more predetermined BCIs.

8 Conclusions

Where many paths and errands meet.
And whither then? I cannot say.
J. R. R. Tolkien – *The Lord Of The Rings*

This thesis has presented methods, validated by experiments, to automatically identify event-related potentials in EEG activity for use in BCI; two particular potentials, P300 and ErrP, have been the focus of the work.

For P300 detection, we first applied a method based on ARX models, already used in the literature to study event-related potentials, to filter EEG signals, and used some standard classifiers to classify the filtered signals. The results were not satisfying; in particular, we tried the method on a data set from a BCI competition (recorded with a P300 speller), and there was a significant difference between the performances of our method and of the winning entries.

We tried to understand if the performance of our method on the competition data set depended on the ARX filtering or on the feature extraction and classification steps. Thus, we replicated a method from the winners of the BCI competition, and used it to classify the output of the ARX filtering. The results obtained on the components discarded by the ARX model convinced us that the problem lied in the ARX filtering.

We explored then the path of automatic extraction of features from raw EEG recordings. To this aim, we developed a genetic algorithm that, together with a standard logistic classifier, was able to achieve 100% correct classification on the competition data set we used before. The same algorithm performed well also on the EEG data we recorded specifically for this work with a P300-based speller, even when compared to standard algorithms from the literature.

The genetic algorithm was applied also to recognize P300s in EEG data recorded from subjects affected by ALS, with some success. Preliminary online tests showed that the classifier found by the genetic algorithm can be applied in real time in an actual P300 speller and also to drive a wheelchair, in a very noisy environment.

We found an interesting interpretation of the features extracted by the genetic algorithm combined with the classifier trained on them. This interpretation shows how the parts of an EEG signal that are most useful for the detection of P300 are not always those expected from the information found in the literature. In fact, the genetic algorithm is blind, and it is not tailored for a particular potential: Results indicate that it can be effective not only for P300, but also for ErrP detection.

We explored a way to improve BCI performance by using ErrPs to identify BCI mistakes. We first made experiments to detect ErrPs in settings different from a BCI, with subjects performing tasks where the knowledge about ErrPs seemed more firm in the literature. We developed and tested new techniques to automatically detect ErrPs and they proved to be effective. We moved then to a task more similar to the typical interaction between a user and a BCI, and at last we experimented with a BCI task: the P300 speller. Even in this case, automatic detection of ErrP was possible and effective. Online experiments were also carried on and confirmed the viability of online ErrP detection.

We analyzed the performance gain that ErrP detection may give to a BCI. We pointed out some limitations of a metric widely used in the literature to measure BCI performance, the information transfer rate, and then introduced a metric based on the concept of utility, which is measured in a way related to the task performed by the BCI. Utility measures the performance of a BCI from the point of view of its user, and hence it can be used to tune and optimize the parameters of the BCI. For the P300 speller, we chose the expected number of correct letters per time unit as the measure of utility. Using this task-oriented approach, we determined the constraints on the accuracy of an ErrP detector in order to be helpful for a subject, and the performance gain achieved. We think that a more widespread use of a task-based approach to measuring BCI performance could be useful to the whole BCI community.

The utility for a BCI can be computed by modeling the BCI behavior as a function of classical performance measures for classifiers, like accuracy, recall, and confusion matrix. So, it is not too far fetched to envision a fixed set of standard BCI applications that could work as a test bed to compare different classification and signal processing techniques.

The major part of the experimental work has been done *offline*. This partly depends on the fact that we developed new classification algorithms, which is better done offline; also, some limitations of the EEG hardware we have been using have slowed the port of the algorithm to an online settings. The development of new software components to interface the hardware have permitted the to begin the work online. While we are making further experiments, the performance of the online experiments done so far seems to be comparable to what we obtained in offline experiments.

8.1 Future Work

More online experiments would permit also to validate the predictions of the formulas in Chapter 6 about the possible improvements to a BCI by means of ErrP detection and to assess the validity of the utility as a way to optimize project parameters.

We are moving forward in developing the system described in Chapter 3, with menus controlled by a P300-based BCI, and its integration with an autonomous wheelchair. The completion of this system will help in gathering more experimental data in a more realistic setting.

The genetic algorithm presented in Section 5.4 seems to be effective also for ErrP detection, but experiments with more data are needed to confirm this impression. A possible extension of the application of the genetic algorithm is to channel selection: It should be possible to extract a measure of the importance of an EEG channel from the resulting template, e.g., the energy of the template, and use it to choose the best subset of channels. No work has been done on this aspect yet, because we concentrated our efforts on other topics.

The final output of our genetic algorithm is a set of templates; this is not necessarily the best approach. For potentials that are less stable in time, maybe it would be useful to use features more complex than those we used, associated with a classifier faster to train than the logistic one in order limit the running time of the algorithm.

We have analyzed the gain in the performance of a BCI by means of ErrP detection, but in that analysis we have considered the accuracy and the speed of the BCI as fixed parameters. This is not entirely the case, because it is possible to vary some parameters of the BCI, e.g., the number of repetitions in a P300 speller, and influence speed and accuracy. An analysis of the impact of such variations on the overall performance of the system would be interesting.

An adaptive system could be also conceived, where parameters of the BCI are modified in real time to maintain the best trade-off between speed and accuracy. For example, in the P300 speller the number of repetitions could be adjusted based on measure of the confidence of the classification.

As usual in the research, every time one make a step, many new possibilities open and many other paths could be taken. The possibilities hinted above are just a small fraction of the ideas that come to the mind, and, sadly, it is not possible to explore every single path. But the fascination and the wonder of new discoveries lie also in this: There is ever more to discover.

Still round the corner there may wait
A new road or a secret gate,
And though we pass them by today,
Tomorrow we may come this way
And take the hidden paths that run
Towards the Moon or to the Sun.
J. R. R. Tolkien – *The Lord Of The Rings*

Bibliography

[1] Jonathan R. Wolpaw, Niels Birbaumer, Dennis J. McFarland, Gert Pfurtscheller, and Theresa M. Vaughan, "Brain-computer interfaces for communication and control," *Clinical Neurophysiology*, vol. 113, pp. 767–791, 2002.

[2] "AIRLab: Artificial Intelligence and Robotics Lab," http://airlab.elet.polimi.it/, Web site.

[3] Robert M. Berne and Matthew N. Levy, *Physiology*, Mosby-Year Book, St. Louis, Missouri, USA, 3 edition, 1993.

[4] R. Douglas Fields, "Myelination: An overlooked mechanism of synaptic plasticity?," *The Neuroscientist*, vol. 11, no. 6, pp. 528–531, 2005.

[5] Laura Kauhanen, Tommi Nykopp, Janne Lehtonen, Pasi Jylänki, Jukka Heikkonen, Pekka Rantanen, Hannu Alaranta, and Mikko Sams, "EEG and MEG brain-computer interface for tetraplegic patients," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 14, no. 2, pp. 190–193, June 2006.

[6] B. Graimann, Jane E. Huggins, S. P. Levine, and Gert Pfurtscheller, "Toward a direct brain interface based on human subdural recordings and wavelet-packet analysis," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 954–962, June 2004.

[7] J. A. Wilson, E. A. Felton, P. C. Garell, Gerwin Schalk, and J. C. Williams, "ECoG factors underlying multimodal control of a brain-computer interface," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 14, no. 2, pp. 246–250, June 2006.

[8] Gerwin Schalk, Kai J. Miller, Nicholas R. Anderson, J. A. Wilson, M. D. Smyth, Jeffrey G. Ojemann, Daniel W. Moran, Jonathan R. Wolpaw, and Eric C. Leuthardt, "Two-dimensional movement control using electrocorticographic signals in humans," *Journal of Neural Engineering*, vol. 5, no. 1, pp. 75–84, 2008.

[9] P. R. Kennedy and R. A. Bakay, "Restoration of neural output from a paralyzed patient by a direct brain connection," *Neuroreport*, vol. 9, no. 8, pp. 1707–1711, June 1998.

[10] Philip R. Kennedy, M. T. Kirby, Melody M. Moore, B. King, and A. Mallory, "Computer control using human intracortical local field potentials," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 12, no. 3, pp. 339–344, September 2004.

[11] Leigh R. Hochberg, Mijail D. Serruya, Gerhard M. Friehs, Jon A. Mukand, Maryam Saleh, Abraham H. Caplan, Almut Branner, David Chen, Richard D. Penn, and John P. Donoghue, "Neuronal ensemble control of prosthetic devices by a human with tetraplegia," *Nature*, vol. 442, pp. 164–171, July 2006.

[12] Johan Wessberg, Christopher R. Stambaugh, Jerald D. Kralik, Pamela D. Beck, Mark Laubach, John K. Chapin, Jung Kim, S. James Biggs, Mandayam A. Srinivasan, and Miguel A. L. Nicolelis, "Real-time prediction of hand trajectory by ensembles of cortical neurons in primates," *Nature*, vol. 408, pp. 361–265, November 2000.

[13] Jose M. Carmena, Mikhail A. Lebedev, Roy E. Crist, Joseph E. O'Doherty, David M. Santucci, Dragan F. Dimitrov, Parag G. Patil, Craig S. Henriquez, and Miguel A. L. Nicolelis, "Learning to control a brain-machine interface for reaching and grasping by primates," *PLoS Biology*, vol. 1, no. 2, pp. e42, 2003.

[14] Matthew D. Johnson, Robert K. Franklin, Matthew D. Gibson, Richard B. Brown, and Daryl R. Kipke, "Implantable microelectrode arrays for simultaneous electrophysiological and neurochemical recordings," *Journal of Neuroscience Methods*, vol. 174, no. 1, pp. 62–70, September 2008.

[15] Meel Velliste, Sagi Perel, M. Chance Spalding, Andrew S. Whitford, and Andrew B. Schwartz, "Cortical control of a prosthetic arm for self-feeding," *Nature*, vol. 453, pp. 1098–1101, June 2008.

[16] Michael E. Phelps, Edward J. Hoffman, Nizar A. Mullani, and Michel M. Ter-Pogossian, "Application of annihilation coincidence detection to transaxial reconstruction tomography," *The Journal of Nuclear Medicine*, vol. 16, no. 3, pp. 210–224, 1975.

[17] Nikolaus Weiskopf, Klaus Mathiak, Simon W. Bock, Frank Scharnowski, Ralf Veit, Wolfgang Grodd, Rainer Goebel, and Niels Birbaumer, "Principles of a brain-computer interface (BCI) based on real-time functional magnetic resonance imaging (fMRI)," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 966–970, 2004.

[18] Ranganatha Sitaram, Haihong Zhang, Cuntai Guan, Manoj Thulasidas, Yoko Hoshi, Akihiro Ishikawa, Koji Shimizu, and Niels Birbaumer, "Temporal classification of multichannel near-infrared spectroscopy signals of motor imagery for developing a brain-computer interface.," *Neuroimage*, vol. 34, no. 4, pp. 1416–1427, February 2007.

[19] Jaakko Malmivuo and Robert Plonsey, *Bioelectromagnetism*, Oxford University Press, 1995.

[20] Hans Berger, "Über das Elektroenkephalogramm des Menschen," *Archiv für Psychiatrie und Nervenkrankheiten*, vol. 87, pp. 527–570, 1929.

[21] Sandra K. Loo, "EEG and neurofeedback findings in ADHD," *The ADHD Report*, vol. 11, no. 3, pp. 1–6, June 2003.

[22] Jacques J. Vidal, "Toward direct brain-computer communication," *Annual Review of Biophysiology and Bioengineering*, vol. 2, pp. 157–180, 1973.

[23] Dennis J. McFarland, William A. Sarnacki, Theresa M. Vaughan, and Jonathan R. Wolpaw, "Brain-computer interface (BCI) operation: signal and noise during early training sessions," *Clinical Neurophysiology*, vol. 116, pp. 56–62, 2005.

[24] G. Gratton, M. G. Coles, and E. Donchin, "A new method for off-line removal of ocular artifact," *Electroencephalography and Clinical Neurophysiology*, vol. 55, no. 4, pp. 458–484, April 1983.

[25] R. J. Croft and R. J. Barry, "Removal of ocular artifact from the EEG: A review," *Clinical Neurophysiology*, vol. 30, no. 1, pp. 5–19, February 2000.

[26] Tzyy-Ping Jung, Scott Makeig, Martin J. McKeown, Anthony J. Bell, Te-Won Lee, and Terrence J. Sejnowski, "Imaging brain dynamics using independent component analysis," *Proceedings of the IEEE*, vol. 89, no. 7, pp. 1107–1122, July 2001.

[27] David A. Peterson, James N. Knight, Michael J. Kirby, Charles W. Anderson, and Michael H. Thaut, "Feature selection and blind source separation in an EEG-based brain-computer interface," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 19, pp. 3128–3140, November 2005.

[28] Charles W. Anderson and Zlatko Sijerčić, "Classification of EEG signals from four subjects during five mental tasks," in *Solving Engineering Problems with Neural Networks: Proceedings of the Conference on Engineering Applications in Neural Networks (EANN'96)*, A. B. Bulsari, S. Kallio, and D. Tsaptsinos, Eds. Systems Engineering Association, Finland, 1996, pp. 407–414.

[29] Jacques J. Vidal, "Real-time detection of brain events in EEG," *Proceedings of the IEEE*, vol. 65, no. 5, pp. 633–641, May 1977.

[30] Xiaorong Gao, Dingfeng Xu, Ming Cheng, and Shangkai Gao, "A BCI-based environmental controller for the motion-disabled," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 11, no. 2, pp. 137–140, 2003.

[31] Matthew Middendorf, Grant McMillan, Gloria Calhoun, and Keith S. Jones, "Brain-computer interfaces based on the steady-state visual-evoked response," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, no. 2, pp. 211–214, June 2000.

[32] Thilo Hinterberger, Stefan Schmidt, Nicola Neumann, Jürgen Mellinger, Benjamin Blankertz, Gabriel Curio, and Niels Birbaumer, "Brain-computer communication and slow cortical potentials," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 1011–1018, June 2004.

[33] Georg E. Fabiani, Dennis J. McFarland, Jonathan R. Wolpaw, and Gert Pfurtscheller, "Conversion of EEG activity into cursor movement by a brain-computer interface (BCI)," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 12, no. 3, pp. 331–338, September 2004.

[34] Christoph Guger, H. Ramoser, and Gert Pfurtscheller, "Real-time EEG analysis with subject-specific spatial patterns for a brain-computer interface (BCI)," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, no. 4, pp. 447–456, December 2000.

[35] José del R. Millán, Frédéric Renkens, Josep Mouriño, and Wulfram Gerstner, "Noninvasive brain-actuated control of a mobile robot

by human EEG," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 1026–1033, 2004.

[36] José del R. Millán, "MAIA project," http://www.maia-project.org/, Web site.

[37] Johan Philips, José del R. Millán, Gerolf Vanacker, Eileen Lew, Ferran Galán, Pierre W. Ferrez, Hendrik Van Brussel, and Marnix Nuttin, "Adaptive shared control of a brain-actuated simulated wheelchair," in *Proceedings of the 2007 IEEE 10th International Conference on Rehabilitation Robotics, June 12-15, Noordwijk, The Netherlands*, 2007, pp. 408–414.

[38] Alexander Hüntemann, Eric Demeester, Dirk Vanhooydonck, Gerolf Vanacker, Alexandra Degeest, Hendrik Van Brussel, and Marnix Nuttin, "Bayesian estimation of human intent for driving a powered wheelchair," in *3rd International Workshop on Advances in Service Robotics*, Vienna – Austria, July 2006.

[39] Roland Grave de Peralta Menendez, Sara L. González Andino, S. Morand, C. M. Michel, and T. Landis, "Imaging the electrical activity of the brain: ELECTRA," *Human Brain Mapping*, vol. 9, no. 1, pp. 1–12, 2000.

[40] Rolando Grave de Peralta Menendez, Quentin Noirhomme, Febo Cincotti, Donatella Mattia, Fabio Aloise, and Sara González Andino, "Modern electrophysiological methods for brain-computer interfaces," *Computational Intelligence and Neuroscience*, vol. 2007, 2007, Article ID 56986, 8 pages.

[41] Febo Cincotti, Laura Kauhanen, Fabio Aloise, Tapio Palomäki, Nicholas Caporusso, Pasi Jylänki, Donatella Mattia, Fabio Babiloni, Gerolf Vanacker, Marnix Nuttin, Maria Grazia Marciani, and José del R. Millán, "Vibrotactile feedback for brain-computer interface operation," *Computational Intelligence and Neuroscience*, vol. 2007, 2007, Article ID 48937, 12 pages.

[42] Gert Pfurtscheller and Christa Neuper, "Motor imagery and direct brain-computer communication," *Proceedings of the IEEE*, vol. 89, no. 7, pp. 1123–1134, 2001.

[43] Robert Leeb, Doron Friedman, Gernot R. Müller-Putz, Reinhold Scherer, Mel Slater, and Gert Pfurtscheller, "Self-paced (asynchronous) BCI control of a wheelchair in virtual

environments: A case study with a tetraplegic," *Computational Intelligence and Neuroscience*, pp. 1–8, 2007.

[44] Gert Pfurtscheller, Gernot R. Müller-Putz, Jörg Pfurtscheller, and Rüdiger Rupp, "EEG-based asynchronous BCI controls functional electrical stimulation in a tetraplegic patient," *EURASIP Journal on Applied Signal Processing*, vol. 19, pp. 3152–3155, 2005.

[45] Jonathan R. Wolpaw and Dennis J. McFarland, "Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 51, pp. 17849–17854, December 2004.

[46] Benjamin Blankertz, Guido Dornhege, Matthias Krauledat, Klaus-Robert Müller, Volker Kunzmann, Florian Losch, and Gabriel Curio, "The Berlin brain-computer interface: EEG-based communication without subject training," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 14, no. 2, pp. 147–152, 2006.

[47] Benjamin Blankertz, Guido Dornhege, Matthias Krauledat, Klaus-Robert Müller, and Gabriel Curio, "The non-invasive Berlin brain–computer interface: Fast acquisition of effective performance in untrained subjects," *Neuroimage*, vol. 37, no. 2, pp. 539–550, 2007.

[48] John K. Chapin, Karen A. Moxona, Ronald S. Markowitz, and Miguel A. L. Nicolelis, "Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex," *Nature Neuroscience*, vol. 2, pp. 664–670, 1999.

[49] Andrew B. Schwartz, X. Tracy Cui, Douglas J. Weber, and Daniel W. Moran, "Brain-controlled interfaces: Movement restoration with neural prosthetics," *Neuron*, vol. 52, no. 1, pp. 205–220, October 2006.

[50] W. Grey Walter, R. Cooper, V. J. Aldridge, W. C. McCallum, and A. L. Winter, "Contingent negative variation: An electric sign of sensori-motor association and expectancy," *Nature*, vol. 203, pp. 380–384, 1964.

[51] Benjamin Blankertz, Gabriel Curio, and Klaus-Robert Müller, "Classifying single trial EEG: Towards brain computer interfacing," in *Advances in Neural Information*

Processing Systems (NIPS 01), T. G. Diettrich, S. Becker, and Z. Ghahramani, Eds. 2002, vol. 14, pp. 157–164, MIT Press.

[52] Benjamin Blankertz, Guido Dornhege, Roman Krepki Christin Schäfer, Jens Kohlmorgen, Volker Kunzmann Klaus-Robert Müller, Florian Losch, and Gabriel Curio, "Boosting bit rates and error detection for the classification of fast-paced motor commands based on single-trial EEG analysis," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 11, no. 2, pp. 127–131, June 2003.

[53] Samuel Sutton, Margery Braren, Joseph Zubin, and E. R. John, "Evoked-potential correlates of stimulus uncertainty," *Science*, vol. 150, no. 3700, pp. 1187–1188, 1965.

[54] Eric W. Sellers, Dean J. Krusienski, d Dennis J. McFarlan, Theresa M. Vaughan, and Jonathan R. Wolpaw, "A P300 event-related potential brain-computer interface (BCI): the effects of matrix size and inter stimulus interval on performance," *Biological Psychology*, vol. 73, no. 3, pp. 242–252, October 2006.

[55] Lawrence A. Farwell and Emanuel Donchin, "Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials," *Electroencephalography and Clinical Neurophysiology*, vol. 6, no. 70, pp. 510–523, December 1988.

[56] Emanuel Donchin, Kevin M. Spencer, and Ranjith Wijesinghe, "The mental prosthesis: Assessing the speed of a P300-based brain-computer interface," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, no. 2, pp. 174–179, June 2000.

[57] Brice Rebsamen, Etienne Burdet, Cuntai Guan, Haihong Zhang, Chee Leong Teo, Qiang Zeng, Marcelo H. Ang Jr., and Christian Laugier, "A brain-controlled wheelchair based on P300 and path guidance," in *Biomedical Robotics and Biomechatronics, 2006. BioRob 2006. The First IEEE/RAS-EMBS International Conference on*, 2006, pp. 1101–1106.

[58] Matthias Kaper, Peter Meinicke, Ulf Grossekathoefer, Thomas Lingner, and Helge Ritter, "BCI competition 2003 – data set IIb: support vector machines for the P300 speller paradigm," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 1073–1076, June 2004.

[59] Benjamin Blankertz, Klaus-Robert Müller, Gabriel Curio, Theresa M. Vaughan, Gerwin Schalk, Jonathan R. Wolpaw, Alois Schlögl, Christa Neuper, Gert Pfurtscheller, Thilo Hinterberger, Michael Schröder, and Niels Birbaumer, "The BCI competition 2003: Progress and perspectives in detection and discrimination of EEG single trials," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 1044–1051, 2004.

[60] "BCI competition 2003," `http://ida.first.fraunhofer.de/projects/bci/competition_ii/index.html`, 2003.

[61] F. Piccione, F. Giorgi, P. Tonin, K. Priftis, S. Giove, S. Silvoni, G. Palmas, and F. Beverina, "P300-based brain computer interface: reliability and performance in healthy and paralysed participants.," *Clinical Neurophysiology*, vol. 117, no. 3, pp. 531–537, March 2006.

[62] Reinhold Scherer, Gernot R. Müller, Christa Neuper, Bernhard Graimann, and Gert Pfurtscheller, "An asynchronously controlled EEG-based virtual keyboard: Improvement of the spelling rate," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 979–984, June 2004.

[63] Michael Falkenstein, Joachim Hohnsbein, Jörg Hoormann, and Ludger Blanke, "Effects of crossmodal divided attention on late ERP components. II. Error processing in choice reaction tasks.," *Electroencephalography and Clinical Neurophysiology*, vol. 78, no. 6, pp. 447–455, June 1991.

[64] W. J. Gehring, B. Goss, M. G. H. Coles, D. E. Meyer, and E. Donchin, "A neural system for error detection and compensation," *Psychological Science*, vol. 4, no. 6, pp. 385–390, 1993.

[65] Michael Falkenstein, "ERP correlates of erroneous performance," in *Errors, Conflicts, and the Brain. Current Opinions on Performance Monitoring*, Markus Ullsperger and Michael Falkenstein, Eds., 2004, pp. 5–14.

[66] Sander Nieuwenhuis, K. Richard Ridderinkhof, Jos Blom, Guido P. H. Band, and Albert Kok, "Error-related brain potentials are differentially related to awareness of response errors: Evidence from an antisaccade task," *Psychophysiology*, vol. 38, no. 5, pp. 752–760, 2001.

[67] Wolfgang H. R. Miltner, Christoph H. Braun, and Michael G. H. Coles, "Event-related brain potentials following incorrect feedback in a time-estimation task: Evidence for a 'generic' neural system for error-detection," *Journal of Cognitive Neuroscience*, vol. 9, no. 6, pp. 788–798, 1997.

[68] William J. Gehring and Adrian R. Willoughby, "The medial frontal cortex and the rapid processing of monetary gains and losses," *Science*, vol. 295, no. 5563, pp. 2279–2282, March 2002.

[69] Gerwin Schalk, Jonathan R. Wolpaw, Dennis J. McFarland, and Gert Pfurtscheller, "EEG-based communication: presence of an error potential," *Clinical Neurophysiology*, vol. 111, no. 12, pp. 2138–2144, 2000.

[70] Benjamin Blankertz, Guido Dornhege, and Gabriel Curio Christin Schäfer, "Single trial detection of eeg error potentials: A tool for increasing bci transmission rates," in *Proceedings of International Conference on Artificial Neural Networks - ICANN 2002*, 2002, LNCS, pp. 1137–1143.

[71] Anna Buttfield, Pierre W. Ferrez, and José del R. Millán, "Towards a robust BCI: Error potentials and online learning," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 14, no. 2, pp. 164–168, June 2006.

[72] Pierre W. Ferrez and José del R. Millán, "You are wrong!— automatic detection of interaction errors from brain waves," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 2005, pp. 1413–1418.

[73] Pierre W. Ferrez and José del R. Millán, "Error-related EEG potentials generated during simulated brain-computer interaction," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 3, pp. 923–929, March 2008.

[74] Pierre W. Ferrez and José del R. Millán, "EEG-based brain-computer interaction: Improved accuracy by automatic single-trial error detection," in *Advances in Neural Information Processing Systems 20*, 2007, pp. 441–448.

[75] Pierre W. Ferrez and José del R. Millán, "Simultaneous real-time detection of motor imagery and error-related potentials for improved BCI accuracy," in *Proceedings of the 4th International*

Brain-Computer Interface Workshop & Training Course, Graz, Austria, Sept. 2008, Technischen Universität Graz, pp. 197–202.

[76] Jessica D. Bayliss, Samuel A. Inverso, and Aleksey Tentler, "Changing the P300 brain computer interface," *Cyberpsychology & Behavior*, vol. 7, no. 6, pp. 694–704, 2004.

[77] Theresa M. Vaughan, Dennis J. Mcfarland, Gerwin Schalk, William A. Sarnacki, Dean J. Krusienski, Eric W. Sellers, and Jonathan R. Wolpaw, "The Wadsworth BCI research and development program: At home with BCI," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 14, no. 2, pp. 229–233, 2006.

[78] M. Matteucci, R. Carabalona, M. Casella, E. Di Fabrizio, F. Gramatica, M. Di Rienzo, E. Snidero, L. Gavioli, and M. Sancrotti, "Micropatterned dry electrodes for brain-computer interface," *Microelectronic Engineering*, vol. 84, pp. 1737–1740, 2007.

[79] José del R. Millán, "On the need for on-line learning in brain-computer interfaces," in *Proc. Int. Joint Conf. on Neural Networks*, 2004.

[80] Pradeep Shenoy, Matthias Krauledat, Benjamin Blankertz, Rajesh P N Rao, and Klaus-Robert Müller, "Towards adaptive classification for bci," *Journal of Neural Engineering*, vol. 3, pp. R13–R23, 2006.

[81] Niels Birbaumer, Andrea Kübler, Nimr Ghanayim, Thilo Hinterberger, Jouri Perelmouter, Jochen Kaiser, Iver Iversen, Boris Kotchoubey, Nicola Neumann, and Herta Flor, "The thought translation device (TTD) for completely paralyzed patients," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, no. 2, pp. 190–193, June 2000.

[82] "EBNeuro," `http://www.ebneuro.biz/`, Web site.

[83] Gerwin Schalk, Dennis J. McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R. Wolpaw, "BCI2000: a general-purpose brain-computer interface (BCI) system," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 1034–1043, June 2004.

[84] Jürgen Mellinger and Gerwin Schalk, "BCI2000: A general-purpose software platform for BCI research," In Dornhege et al. [135].

[85] Rossella Blatt, Simone Ceriani, Bernardo Dal Seno, Giulio Fontana, Matteo Matteucci, and Davide Migliore, "Brain control of a smart wheelchair," in *Intelligent Autonomous Systems 10 – IAS-10*, Wolfram Burgard, Rüdiger Dillmann, Christian Plagemann, and Nikolaus Vahrenkamp, Eds., Baden Baden, Germany, July 2008, pp. 221–228, IOS Press.

[86] Andrea Bonarini, Matteo Matteucci, and Marcello Restelli, "MRT: Robotics off-the-shelf with the modular robotic toolkit," in *Software Engineering for Experimental Robotics*, Davide Brugali, Ed. April 2007, vol. 30 of *Springer Tracts in Advanced Robotics*, Springer - Verlag.

[87] Daniel Wagner and Dieter Schmalstieg., "ARToolKitPlus for pose tracking on mobile devices," *In Computer Vision Winter Workshop*, 2007.

[88] Simone Ceriani, "Sviluppo di una carrozzina autonoma d'ausilio ai disabili motori," M.S. thesis, Politecnico di Milano, 2008.

[89] Andrea Bonarini, Matteo Matteucci, and Marcello Restelli, "A novel model to rule behavior interaction," in *Proceedings of the 8th Conference on Intelligent Autonomous Systems (IAS-8)*. 2004, pp. 199–206, IOS Press, Amsterdam.

[90] Andrea Bonarini, Giovanni Invernizzi, Thomas Halva Labella, and Matteo Matteucci, "An architecture to coordinate fuzzy behaviors to control an autonomous robot," *Fuzzy sets and systems*, vol. 134, no. 1, pp. 101–115, 2003.

[91] "The MathWorks – MATLAB," `http://www.mathworks.com/products/matlab/`, Web site.

[92] Bob Kempa, Alpo Värrib, Agostinho C. Rosac, Kim D. Nielsend, and John Gade, "A simple format for exchange of digitized polygraphic recordings," *Electroencephalography and Clinical Neurophysiology*, vol. 82, no. 5, pp. 391–393, May 1992.

[93] "Full specification of EDF," `http://www.edfplus.info/specs/edf.html`, Web site.

[94] Alois Schlögl, Gernot Müller, Reinhold Scherer, and Gert Pfurtscheller, "BIOSIG: An open source software package for biomedical signal processing," in *2nd OpenECG Workshop*, 2004, pp. 77–78.

[95] "The BioSig project," http://biosig.sourceforge.net/, Web site.

[96] Ian H. Witten and Eibe Frank, *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann, 2 edition, 2005.

[97] "Weka 3: Data mining software in Java," http://www.cs.waikato.ac.nz/ml/weka/, Web site.

[98] *Chih-Chung Chang and Chih-Jen Lin*, 2001, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[99] Luca T. Mainardi, Jukka Kupila, Kari Nieminen, Ilkka Korhonen, Anna Maria Bianchi, Linda Pattini, Jukka Takala, Jari Karhu, and Sergio Cerutti, "Single sweep analysis of event related auditory potentials for the monitoring of sedation in cardiac surgery patients," *Computer Methods and Programs in Biomedicine*, vol. 63, no. 3, pp. 219–227, 2000.

[100] Lorenzo Rossi, Anna Maria Bianchi, Anna Merzagora, Alberto Gaggiani, Sergio Cerutti, and Francesco Bracchi, "Single trial somatosensory evoked potential extraction with ARX filtering for a combined spinal cord intraoperative neuromonitoring technique," *BioMedical Engineering OnLine*, vol. 6, 2007.

[101] Eliahu I. Jury, *Theory and application of the Z-transform method*, Wiley, 1964.

[102] Will Gersch, "Spectral analysis of EEG's by autoregressive decomposition of time series," *Mathematical Biosciences*, vol. 7, pp. 205–222, February 1970.

[103] S. Cerutti, D. Liberati, and P. Mascellani, "Parameter extraction in EEG processing during riskful neurosurgical operations," *Signal processing*, vol. 9, pp. 25–35, 1985.

[104] Hirotugu Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, pp. 716–723, 1974.

[105] G. M. Ljung and G. E. P. Box, “On a measure of lack of fit in time series models,” *Biometrika*, vol. 65, no. 2, pp. 297–303, August 1978.

[106] S. le Cessie and J. C. van Houwelingen, “Ridge estimators in logistic regression,” *Applied Statistics*, vol. 41, no. 1, pp. 191–201, 1992.

[107] Vladimir N. Vapnik, “An overview of statistical learning theory,” *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, September 1999.

[108] Vladimir N. Vapnik, *Statistical learning theory*, Wiley, 1998.

[109] John H. Holland, *Adaptation in Neural and Artificial Systems*, University of Michigan Press, 1975.

[110] Ingo Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Fromman-Holzboog, 1973.

[111] Randy L. Haupt and Sue Ellen Haupt, *Practical Genetic Algorithms*, Wiley, 2004.

[112] Finn V. Jensen, *Bayesian Networks and Decision Graphs*, Springer, 2001.

[113] Tom Mitchell, *Machine Learning*, McGraw Hill, 1997.

[114] Christopher M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[115] Robert E. Schapire, “The boosting approach to machine learning: An overview,” in *Nonlinear Estimation and Classification*, D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, and B. Yu, Eds. 2003, Springer.

[116] Jens Kohlmorgen and Benjamin Blankertz, “Bayesian classification of single-trial event-related potentials in EEG,” *International Journal of Bifurcation and Chaos*, vol. 14, no. 2, pp. 719–726, 2004.

[117] R. Boostani, B. Graimann, M. H. Moradi, and G. Pfurtscheller, “A comparison approach toward finding the best feature and classifier in cue-based BCI,” *Medical and Biological Engineering and Computing*, vol. 45, no. 4, pp. 403–412, April 2007.

[118] Luca Citi, Riccardo Poli, Caterina Cinel, and Francisco Sepulveda, "Feature selection and classification in brain computer interfaces by a genetic algorithm," in *Late-breaking papers of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, 2004, vol. CD-ROM.

[119] John Polich, "Updating P300: An integrative theory of P3a and P3b," *Clinical Neurophysiology*, vol. 118, pp. 2128–2148, 2007.

[120] Eckart Altenmüller, "Psychophysiology and EEG," in *Electroencephalography*, Ernst Niedermeyer and Fernando Lopes Da Silva, Eds. Williams & Wilkins, Baltimore, USA, 3rd edition, 1993.

[121] B. A. Eriksen and C. W. Eriksen, "Effects of noise letters upon the identification of a target letter in a nonsearch task," *Perception & Psychophysics*, vol. 16, pp. 143–149, 1974.

[122] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd int. Conf. on Knowledge Discovery and Data Mining*, Portland, Oregon, USA, August 1996, pp. 226–231, AAAI Press.

[123] Jerome H. Friedman, "Regularized discriminant analysis," *Journal of the American Statistical Association*, vol. 84, no. 405, pp. 165–175, 1989.

[124] Evelyn Fix and J. L. Hodges Jr, "Discriminatory analysis. Nonparametric discrimination: Consistency properties," Tech. Rep. 4, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[125] Belur V. Dasarathy, Ed., *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, Ieee Computer Society, 1990.

[126] Jacob Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, pp. 37–46, 1960.

[127] Alois Schlögl, C. Keinrath, Reinhold Scherer, and Gert Pfurtscheller, "Information transfer of an EEG-based brain computer interface," in *Proceedings of the 1st International IEEE EMBS Conference on Neural Engineering*, 2003, pp. 641–644.

[128] Jonathan R. Wolpaw, Niels Birbaumer, William J. Heetderks, Dennis J. McFarland, Gerwin Schalk P. Hunter Peckham, Emanuel Donchin, Louis A. Quatrano, Charles J. Robinson, and Theresa M. Vaughan, "Brain-computer interface technology: A review of the first international meeting," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, no. 2, pp. 164–173, June 2000.

[129] Claude E. Shannon and Warren Weaver, *The Mathematical Theory of Communication*, The University of Illinois Press, 1949.

[130] Alois Schlögl, Julien Kronegg, Jane E. Huggins, and Steve G. Mason, "Evaluation criteria for BCI research," In Dornhege et al. [135], pp. 327–342.

[131] Gudio Dornhege, *Increasing Information Transfer Rates for Brain-Computer Interfacing*, Ph.D. thesis, University of Potsdam, Germany, 2006.

[132] A. A. Markov, "Extension of the law of large numbers to dependent quantities," *Izvestiya Fiziko-Matematicheskikh Obschestva Kazan University*, vol. 15, pp. 135–156, 1906, In Russian.

[133] Stefan Waner and Steven Costenoble, *Finite Mathematics*, Brooks Cole, 4th edition edition, 2006.

[134] Luigi Bianchi, Lucia Rita Quitadamo, Girolamo Garreffa, Gian Carlo Cardarilli, and Maria Grazia Marciani, "Performances evaluation and optimization of brain computer interface systems in a copy spelling task," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 15, no. 2, pp. 207–216, June 2007.

[135] Guido Dornhege, José del R. Millán, Thilo Hinterberger, Dennis J. McFarland, and Klaus-Robert Müller, Eds., *Towards Brain-Computer Interfacing*, MIT Press, 2007.

[136] J. R. R. Tolkien, *The Lord of the Rings*, HarperCollins, 2001.

[137] J. R. R. Tolkien, *The Hobbit*, HarperCollins, 1995.

[138] J. R. R. Tolkien, *The Silmarillion*, HarperCollins, 1999.

We shouldn't be here at all, if we'd known more about it before we started. But I suppose it's often that way.

J. R. R. Tolkien – *The Lord Of The Rings*

And it is all too likely that some will say at this point: "Shut the book now, dad; we don't want to read any more."

J. R. R. Tolkien – *The Lord Of The Rings*