ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

CLASSIFICATION METHODS FOR MOTOR IMAGERY BASED BRAIN COMPUTER INTERFACES

Ph.D. THESIS

Ayhan YÜKSEL

Department of Electronics and Communications Engineering

Electronics Engineering Programme

JULY 2016

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

CLASSIFICATION METHODS FOR MOTOR IMAGERY BASED BRAIN COMPUTER INTERFACES

Ph.D. THESIS

Ayhan YÜKSEL
(504082201)

Department of Electronics and Communications Engineering

Electronics Engineering Programme

Thesis Advisor: Prof. Dr. Tamer ÖLMEZ

JULY 2016

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

MOTOR HAREKET HAYALİ TABANLI BEYİN BİLGİSAYAR ARAYÜZLERİ İÇİN SINIFLANDIRMA METOTLARI

DOKTORA TEZİ

Ayhan YÜKSEL
(504082201)

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Tamer ÖLMEZ

TEMMUZ 2016

Ayhan YÜKSEL, a Ph.D. student of ITU Graduate School of Science Engineering and Technology 504082201 successfully defended the thesis entitled "CLASSIFICATION METHODS FOR MOTOR IMAGERY BASED BRAIN COMPUTER INTERFACES", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Tamer ÖLMEZ** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Tayfun GÜNEL** İstanbul Technical University	.....
	Prof. Dr. Nizamettin AYDIN Yıldız Technical University	.....
	Prof. Dr. Mustafa BAĞRIYANIK İstanbul Technical University	.....
	Yrd. Doç. Dr. Gökhan BİLGİN Yıldız Technical University	.....

Date of Submission : **26 May 2016**
Date of Defense : **18 July 2016**

To my father,

FOREWORD

This thesis is the result of my three years of studying at Istanbul Technical University, Electrical and Electronics Enginnering Faculty, Medical Electronics Laboratory. The purpose of the research is to study brain computer interfacing and motor imagery classification.

I am deeply grateful to my PhD supervisor Prof. Dr. Tamer Ölmez for his supervision, advice and key scientific insights guiding me to the completion of this thesis. Whenever I lost track on how to proceed he ordered helpful advices. On the other hand, he encouraged me to expand my horizon. He also made numerous suggestions on how to improve this thesis and without his input many issues would not have been spotted. I was lucky to have Prof. Dr. Tamer Ölmez as my supervisor.

Also i appreciate very valuable advices and supervision for my jury members, Prof. Dr. Ahmet Coşkun Sönmez, who passed away suddenly, may his gentle soul rest in peace, Prof. Dr. Tayfun Günel and Prof. Dr. Nizamettin Aydın for their very valuable advices and supervision.

I want to thank to people who i met during my academic study and influenced me any way, Prof. Dr. Mehmet Korürek, may his gentle soul rest in peace, Prof. Dr. Zümray Dokur Ölmez and Dr. Kenan Tekindağ from AORT medical company.

I would like to thank the Scientific and Technological Research Council of Turkey (TÜBİTAK) Science Fellowships and Grant Programmes Department (BİDEB) for my scholarship.

I like to give my very special thanks from deep of my heart to my father, I wish he could be with us now; my mother, my sisters and, my dear wife, who has a great role for encouraging me to finish this thesis.

July 2016 Ayhan YÜKSEL

TABLE OF CONTENTS

Page

ABBREVIATIONS

BCI : Brain Computer Interface
CPU : Central Processing Unit
ALS : Amyotrophic Lateral Sclerosis
VEP : Visual Evoked Potentials
SMA : Supplementary Motor Area
ERD : Event Related Desynchronization
ERS : Event Related Synchronization
EEG : Electroencephalography
SSVEP : Steady State Visual Evoked Potentials
MI : Motor Imagery
SCP : Slow Cortical Potentials
CSP :Common Spatial Patterns
LDA :Linear Discriminant Analysis
RCSP : Regularized Common Spatial Patterns
TRCSP :Tikhonov Regularized Common Spatial Patterns
iCSP :Invariant Common Spatial Patterns
SRCSP :Spatially Regulaized Common Spatial Patterns
SCSP :Stationary Common Spatial Patterns
TR&SR-CSP:Task Related & Spatially Regularized Common Spatial Patterns
SFN : Spatial Filter Network
BP : Back Propagation
LM : Levenberg-Marquardt
CSSP : Common Spatio-Spectral Patterns
CSSP : Common Sparse Spectral Spatial Patterns
SBCSP : Sub Band Common Spatial Patterns
FBCSP : Filter Bank Common Spatial Patterns
Spec-CSP : Spectrally Weighted Common Spatial Patterns
DFBCSP : Discriminative Filter Bank Common Spatial Patterns
FBCSSP : Filter Bank Common Spatio-Spectral Patterns

SYMBOLS

N	: Number of EEG channels
T	: Number of samples in an epoch
K	: Number of epochs in a dataset
R	: Coavariance matrix
$\vec{w}$	: Spatial filter
W	: Spatial filter matrix
C	: Number of classes
m	: Number of spatial filters per class
α	: Multiplier parameter for regularized CSP method
r	: Scale parameter for electrode distance for regularized CSP method
μ	: Learning rate for SFN method
τ	: Delay parameter for CSSP method
B	: Sparsification parameter for CSSSP method
F	: Number of filters for FBCSSP method
P	: Filter degree for FBCSSP method
m_1, m_2	: Number of spatial filters per class for FBCSSP

LIST OF TABLES

Page

LIST OF FIGURES

Page

CLASSIFICATION METHODS FOR MOTOR IMAGERY BASED BRAIN COMPUTER INTERFACES

SUMMARY

Brain computer interfacing (BCI) is an emerging topic which is applied to several areas from gaming equipment to health assistive devices. BCI technology aims establishing a direct communication pathway between the user's brain and any electronic device. Motor imagery is a BCI methodology in which the user's imagining of moving a limb is detected without any actual physical movement. Among different BCI techniques, motor imagery is the most popular BCI methodology because of its practicality and being an independent BCI method. Generally, electroencephalogram (EEG) is used for acquiring motor imagery signals since it is a practical, cheap, fast and non-invasive technique for analyzing brain signals. However, classification of motor imagery signals is a challenging topic. Poor spatial resolution of EEG signal makes it difficult to clearly extract motor imagery signals directly. Poor spatial resolution causes motor imagery signals to be mixed up with the signals from the signal sources in the brain which are much stronger.

In this study, novel methods for classification of motor imagery signals were developed. For this purpose, existing methods and proposed methods were presented and their classification performances were analyzed.

In this thesis, firstly, BCI concept and main BCI methodologies were presented. Motor imagery paradigm and physiological sources and main properties of motor imagery signals were described. Then, an extensive literature review about classification of motor imagery signals was exhibited. Next, the state of art method in the motor imagery classification called *common spatial patterns* (CSP) method was analyzed and then, regularized CSP methods which addresses some drawbacks of CSP were described. Next, the first contribution of this thesis, *task related & spatially regularized CSP* method was presented as a regularized CSP algorithm. After that, the second contribution of this thesis, a spatial filtering and classification structure named *spatial filter network* (SFN) method was presented. After presenting the spatial filtering algorithms, spectral and spatial filtering methodologies were presented. In this manner, a spatio-spectral filtering method called *filter bank common spatio-spectral patterns* (FBCSSP) method was proposed. Before running the proposed methods, datasets used in the study were introduced. Then, selected configurations of the methods were described.

Obtained results of the proposed methods of this study are promising. Their performance evaluations were reported along with important methods from the literature. Developed methods increased the classification performance of the given datasets. Also the physiological suitability of the proposed methods was demonstrated by analyzing obtained spatial and spectral filters. Results showed the effectiveness of the proposed methods.

MOTOR HAREKET HAYALİ TABANLI BEYİN BİLGİSAYAR ARAYÜZLERİ İÇİN SINIFLANDIRMA METOTLARI

ÖZET

Beyin bilgisayar ara yüzü (BBA), son yıllarda oldukça gelişme sağlayan bir araştırma konusudur. Oyun ekipmanlarından yapay organlara kadar çok çeşitli alanlarda kullanım alanlarına sahip BBA teknolojisinin temel amacı, BBA kullanıcısının beyni ve elektronik bir cihaz arasında herhangi bir çevresel sinir yollarına bağlı olmayan aracısız bir haberleşme kanalı kurmaktır. Motor hareket hayali (MHH), kullanıcının, motor bir hareketi hayal etmesi sırasında alınan beyin sinyallerinden o hareketin tahmin edilmesi esasına dayanan bir BBA yöntemidir. Bağımsız bir BBA türü olması ve pratik olması gibi nedenlerden dolayı, motor hayali çeşitli BBA türleri arasında en popüler olanıdır. Motor hareket hayali sinyalleri beyinin motor korteks olarak adlandırılan, istemli hareketlerden sorumlu bölgesinden elde edilir. Bu sinyallerin alınması için fonksiyonel manyetik rezonans görüntüleme (fMRI), pozitron emisyon tomografi (PET), Elektrokortikogram (EKoG) ya da Elektroansefalografi (EEG) gibi işaret alma metotları kullanılabilir. Bu sinyal türleri içerisinde pratik, ucuz, hızlı ve girişimsiz bir yöntem olduğundan, genellikle EEG tercih edilir. Popüler olmasına rağmen, motor hareket hayali işaretlerinin sınıflandırılması oldukça zordur. Bunun temel nedeni ise, düşük uzamsal çözünürlüktür. Düşük uzamsal çözünürlük nedeniyle motor hareket hayali ile ilişkili sinyaller beynin farklı bölgelerinde bulunan başka sinyal kaynakları ile karışır ve bu, elde edilen EEG sinyalinden motor hareket hayali sinyallerinin ortaya çıkarılmasını güçleştirir. Ayrıca motor hareket hayali sinyal karakteristiklerinin kişiden kişiye hatta aynı kişi için zamanla değişebilir olması, sınıf sayısının sınırlı olması, EEG işaretinin durağan olmaması ve deneklerin motor hareketlerin hayal edilmesi konusunda tecrübesiz olması da bu tarz işaretlerin sınıflandırılmasını güçleştiren unsurlardandır.

Tezin giriş kısmında BBA hakkında temel bilgiler ve önemli BBA metotlarından bahsedilmiştir. Bu BBA metotları şu şekilde sıralanabilir: i) Durağan görsel uyarılmış potansiyel (Steady state visual evoked potentials) tabanlı BBA, ii) P300 tabanlı BBA, iii) Yavaş kortikal potansiyeller (Slow cortical potentials) tabanlı BBA, iv) Kortex-neron aktivasyon potansiyeli (Cortical-neuronal activation potentials) tabanlı BBA, v) Motor hareket hayali (Motor imagery) tabanlı BBA. Tez çalışması konusu motor hareket hayali olduğu için, MH hakkında detaylı bilgiler verilmiştir. MH sinyallerinin fizyolojik temelleri, sinyal karakteristikleri, MH sinyallerinin işlenmesi sırasında karşılaşılan zorluklar gibi konulara değinilmiştir. Ardından, motor hareket hayali işaretlerinin sınıflandırılmasına yönelik ayrıntılı bir literatür araştırması sunulmuştur.

Motor hareket hayali sırasında, motor korteks bölgesinde olay ilişki senkronizasyon (event related synchronisation, ERS) ve olay ilişkili desenkronizasyon (event related desynchronisation, ERD) olarak adlandırılan güç değişimleri meydana gelir. ERD, belirli bir frekans bandında ölçülen işaretteki güç düşümüne, ERS ise belirli bir

frekansta ölçülen işaretteki güç artışına karşılık gelir. Motor hareket hayali sırasında en belirleyici işaret, 8-16 Hz arasındaki μ bandındaki güç düşümüdür. Ayrıca 20-30 Hz arasında da ERS işaretleri motor hareket hayali ile birlikte görülmektedir.

Çalışmada motor hareket hayali olarak adlandırılan, kişinin kaslarını hareket ettirmesi ya da ettirmeye niyetlenmesi sırasında beynin motor korteks bölgesinde ortaya çıkan güç değişimlerini analiz eden beyin bilgisayar ara yüzü konusunda mevcut sınıflandırma metotları araştırılmış ve tez çalışmasında yeni metotlar geliştirilmiştir. Bu çalışmada, motor hareket hayali işaretlerinin sınıflandırılması için yeni metotlar geliştirilmiştir. Bu amaçla literatürdeki mevcut metotlar ile beraber, tez kapsamında geliştirilen metotlar sunulmuş ve tüm bu metotların sınıflandırma performansları incelenmiştir.

Metotlar kısmında, MH sınıflandırmasına yönelik literatürdeki belli başlı yöntemler anlatılmıştır. Öncelikle, MH sınıflandırmasına yönelik genel bir çerçeve çizilmiş, ardından, her bir işlem adımı detaylı bir biçimde, literatürdeki mevcut yayınlardan bahsedilerek anlatılmıştır. MH sınıflandırmada çok önemli bir uzamsal sınıflandırma metodu olan "Ortak uzamsal örüntüler" (Common Spatial Patterns, CSP) metodu anlatılmış ve CSP metoduna yapılan iyileştirmelerden bahsedilmiştir.

Metotlar kısımda, Tezin katkılarından ilki olan "Görev ilişkili & uzamsal düzenlemeli ortak uzamsal örüntüler" (Task Related & Spatially Regulaized Common Spatial Patterns, TR&SR-CSP) isimli çalışma anlatılmıştır. Bu çalışmada düzenlenmiş bir CSP metodu önerilmiştir. Metot motor hareket hayali sinyallerinin beyindeki oluşum noktalarını kullanan bir düzenlenmiş (regularized) CSP metodudur. Bu metotta, uzamsal filtrelerin eğitimi sırasında özel olarak hazırlanmış bir ceza matrisi oluşturma algoritması tanıtılmıştır. Bu ceza matrisi, verilen görevlere ilişkin motor kortekseki konumları göz önünde bulundurarak uzamsal filtrelerin korteks üzerinde bu bölgelere odaklanmasını sağlamıştır. Çalışma sonuçları incelendiğinde, fizyolojik verilerle uyumlu sonuçların elde edildiği gözlemlenir. Çalışma 2014 senesinde biyo-informatik ve biyomedikal mühendisliği uluslar arası konferansı" (IWBBIO) konferansında sunulmuştur.

Metotlar kısmında ikinci olarak CSP'nin eksikliklerine değinilerek "Uzamsal filtre ağı" (Spatial Filter Network, SFN) metodu sunulmuştur. Bu metot, bir uzamsal filtre ve bir sınıflandırıcının birlikte optimizasyonunu sağlayan çok katmanlı bir yapıdır. Önerilen yöntem, CSP metodunun iki problemini adresler ve bunlara çözüm arar. Bu problemler, i) CSP metodunun yalnızca sınıflar arası saçılımları iyileştirmesi, buna rağmen, sınıf içi saçılımlar ile ilgilenmemesi, ii)CSP metodunun sınıflandırma performansı ile değil, verilen optimizasyon fonksiyonunu iyileştirmeye çalışmasıdır. SFN ise eğitim kümesindeki her elemanı tek tek ağa sunarak, hem uzamsal filtreyi, hem de sınıflandırıcıyı eğitir. SFN ağının eğitimi için yapay sinir ağlarında kullanılan geriye yayılım yöntemi kullanılmıştır. Bunun için ağa sunulan her eğitim kümesi elemanı için ağın oluşturduğu çıkış incelenmiş ve hem uzamsal filtre ağırlıkları, hem de sınıflandırıcı ağrırlıkları güncellenmiştir. Optimizasyon yöntemi olarak yapay sinir ağlarının eğitiminde kullanılan Levenberg-Marquardt (LM) ve back propogation (BP) metotlarından yararlanılmıştır. Tez içersinde SFN metodunun çalıştırılmasına ve eğitimine yönelik matematiksel denklemler sunulmuştur. SFN metoduna ilişkin yayın, PLoS One isimli dergide yayınlanmıştır.

Metotlar kısmında son olarak uzamsal – spektral filtreleme metotlarına değinilmiştir. Bu metotlar hem uzamsal hem de spektral düzlemde optimizasyonlar yapmaktadırlar.

CSP basitliği ile beraber güçlü bir metot olmasına karşın, bazı eksiklikleri vardır. Motor hareket hayali tabanlı beyin bilgisayar ara yüzlerinde CSP'nin başarısı büyük oranda ERD (olay tabanlı desenkronizasyon) ve ERS (olay tabanlı senkronizasyon) olarak adlandırılan fizyolojik fenomenlere bağlıdır. Halbuki pratikte ERD'nin bulunduğu frekans bandı kişiden kişiye farklılık gösterir. Bu, pratik bir BCI tasarlarken karşılaşılan en büyük problemlerden biridir. Yakın zamana kadar CSP kullanılırken frekans bandı ya geniş bant kullanılarak tanımsız bırakılmaktaydı ya da manüel ayarlanmaktaydı. Genel olarak, CSP'yi EEG işaretini filtrelemeden ya da uygun olmayan bir frekans bandında filtreleyerek uygulamak düşük bir sınıflandırma başarımı verecektir. Bu durumda yapılacak bir iş, zaman harcayıcı bir araştırmalar ve bazı manüel ayarlamalar ile her bir denek için en iyi frekans bandını bulmak olacaktır. Bu şekilde sınıflandırmanın başarımı artırdığı gösterilmiş olsa da, zaman harcayıcı ve zahmetli bir iştir. Bu nedenle son zamanlarda uzamsal filtrelerin frekans filtreleri ile eş zamanlı optimizasyonuna ilişkin yöntemlerin araştırılması oldukça önem kazanmıştır. Bu nedenlerden dolayı, CSP gibi sadece uzamsal düzlemde çalışan metotlar yerine filtrelerin spektral karakteristiklerinin de otomatik olarak iyileştirilmesi amaçlanıştır. Literatürdeki mevcut spatio-spectral metotlar anlatılmış ve tezin son çıktısı olan "Filtre bankası temelli ortak uzamsal örüntüler" (Filter bank common spatio spectral patterns, FBCSSP) isimli, hem spektral hem de uzamsal düzlemde filtre iyileştirilmesi yapan bir metot geliştirilmiştir. Sunulan metot, çeşitli frekanslarda filtreleme yapan bir filtre bankası ve arka arkaya dizilmiş iki adet CSP katmanından oluşur. İlk CSP katmanı, her bir filtre bankası çıkışını uzamsal olarak filtreler böylece, EEG işareti dar bantlarda uzamsal filtrelenmiş olur. İkinci CSP katmanı ise ilk katmandan gelen uzamsal filtrelenmiş işaretleri alarak en önemli işaretleri ortaya çıkartmaya çalışır. Bu nedenle ikinci katman bir nevi frekans seçimi yapmaktadır. İki CSP katmanı ise spatio-spektral bir filtre yapısı oluşturmuş olur. Sonuçlar incelendiğinde, yüksek sınıflandırma başarımlarına ulaşılabildiği görülmektedir. Sunulan çalışma "Biyo-medikal ve biyo informatik alanlarında bilgi teknolojileri" (ITBAM 2016) isimli konferansta sunulmak üzere kabul almıştır. Çalışma "Bilgisayar bilimlerinde konferans notları" (LNCS) isimli dergide yayınlanacaktır.

Sonuçlar kısmında, kullanılan veri kaynaklarından bahsedilmiş, veri kümelerinin özelliklerinden bahsedilmiştir. Daha sonra, sonuçların elde edilmesine yönelik bir çerçeve sunulmuş ve yapılacak değerlendirmeler anlatılmıştır. Ayrıca sonuçlar elde edilirken kullanılan metotlara ilişkin bütün parametre ayarlamaları detaylıca sunulmuştur. Sonuçlar kısmında hem sayısal hem de görsel sonuçlar karşılaştırmalı olarak verilmiştir. Sonuçlar incelendiğinde, önerilen metotların başarılı sonuçlar elde ettiği görülmüştür. Literatürdeki diğer metotlara ilişkin sonuçlar ile değerlendirildiğinde, önerilmiş metotlardan elde edilen sınıflandırma performansları ümit vericidir. Önerilen metotların çalışılan veri kümelerinde performansı yukarı çektiği görülmektedir. Sayısal performans değerlendirmesinin yanında ayrıca, önerilen metotların motor hareket hayali fizyolojisi ile uygunluğu elde edilen uzamsal ve spektral filtrelerin analiz edilmesi ile gözlemlenmiştir. Bütün bu sonuçlar önerilen metotların etkili ve başarılı olduğunu göstermektedir.

1. INTRODUCTION

Human brain is the most complex organ in the human body. Every thought, feeling, memory, experience or action is produced by our brain. Weighting about 1.4 kilograms, this jelly like mass of tissue contains nearly one hundred billion nerve cells called neurons. The complexity of the connectivity between these cells is enormous. Each neuron can make contact with thousands or even tens of thousands of others, via tiny structures called synapses. Also, the pattern and strength of the connections is constantly changing and it is impossible to find two brains alike.

Brain can be thought as a central processing unit (CPU) of an electronic device in which CPU perceives the outer world with various sensors such as temperature sensor, camera or microphone input while executing physical actions with servo motors, speakers or LEDs. The signal is transmitted between the CPU and those peripherals via conductive traces, similar to our peripheral nerves. However, in some cases such as amyotrophic lateral sclerosis (ALS) syndrome, the path between the brain and the peripheral limb may be broken and brain loses the ability to control the limbs. Therefore, an alternative path between the outer world and the brain becomes essential.

Brain computer interface (BCI) is a communication and control mechanism between the user and the system. According to its original definition, a BCI is a communication and control system that does not depend in any way on the brain's normal neuromuscular output channels [1]. It is obvious that, a BCI system gives the user ability to send control commands to an electronic device. Such an interface is the only way for a disabled person to use electronic devices. In order to control a BCI system properly, the user should generate different brain activation signals for different commands. While analyzing the brain signals of the user and converting them to an output device called neuro-prosthesis, a BCI system creates a direct link between the thoughts of the user and the outer world. In the first international BCI technology conference organized in 1999, a BCI system is defined as a "communication and control channel between the brain and the computer that does not depend on the

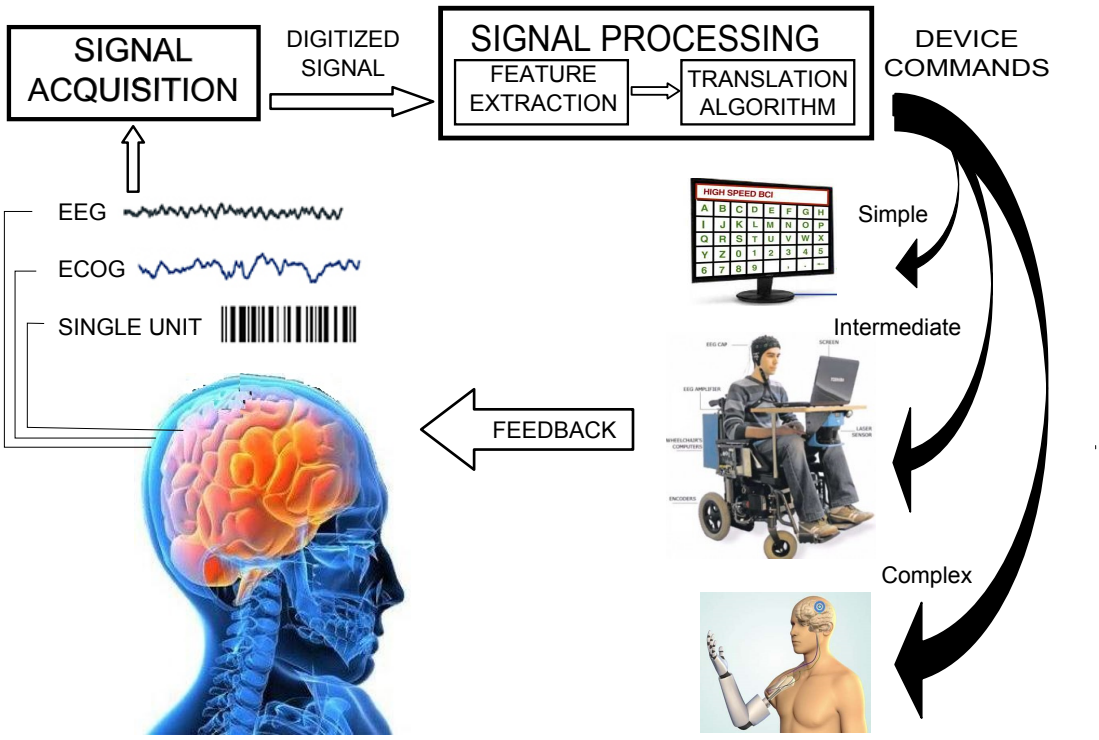

Figure 1.1 : Basic design and operation of a BCI system.

brain's normal output channels of peripheral nerves and muscles" [2]. In Figure 1.1, basic design and working principle of a BCI system is given. Signals from the brain are acquired by electrodes on the scalp or in the head and processed to extract specific signal features (e.g. amplitudes of evoked potentials or sensorimotor cortex rhythms, firing rates of cortical neurons) that reflect the user's intention. These features are translated into commands that operate a device (e.g. a simple word processing program, a wheelchair, or a neuroprosthesis). Evoked potentials [3], sensor-motor cortex rhythms [4, 5] and neuronal activation frequencies [6] are examples for features used in BCI systems.

In the next step, obtained features are converted to the control commands for a controlling device. The controlling device may be any electronic device such as a health assistive device like a neuro-prosthesis, a wheel chair or a word processing program. The success of a BCI depends on how effective the two systems (user and the controlling device) interact with each other. In this system, the user should transmit various control commands by using his/her brain. In this case, the brain may be called as a signal source in a BCI system. The role of this signal source is to generate different patterns for different control commands. And for the BCI system, it should select and

extract the right features from obtained brain signals, convert them to the commands and send to the controlled device effectively [7].

The brain computer interface sets up a communication channel between the patient's brain and the outer world. In this sense, BCI technology may be considered as an assistive technology that improves the life standards of the physically handicapped individuals. Amyotrophic lateral sclerosis (ALS), brainstem stroke, brain or spinal cord injury, cerebral palsy, muscular dystrophies, multiple sclerosis, and numerous other diseases impair the neural pathways that control muscles or impair the muscles themselves. They affect nearly two million people in the United States alone, and far more around the world [8–11]. Those most severely affected may lose all voluntary muscle control, including eye movements and respiration, and may be completely locked in to their bodies, unable to communicate in any way. The immediate goal is to provide these users, who may be completely paralyzed, or 'locked in', with basic communication capabilities so that they can express their wishes to caregivers or even operate word processing programs or neuro prosthesis. There are many BCI studies in the literature about rehabilitation of patients [12–15]. In addition to its benefits for highly disabled individuals, BCI technology is gaining importance in non medical applications such as neuroscience researches, robotic control, gaming and virtual reality [16–23].

A BCI system uses brain signals while recognizing the thoughts or intents of the user. Generally a BCI system uses electroencephalography (EEG) signals that are collected non invasively with electrodes located over the head [24–26]. Apart from EEG there are many studies in the literature about BCI with Magneto encephalography (MEG) [27, 28] functional magnetic resonance imaging (fMRI) [29, 30] and optical imaging methods [31, 32]. MEG, fMRI and optical imaging methods are impractical and expensive methods. Also, fMRI and optical imaging methods measure blood hemodynamic. Therefore, they can't generate quick reactions. With its simplicity, practicality and having quick response, EEG is the most suitable discipline along the mentioned BCI methods [7, 33]. According to the signal type and signal source in the brain, various types of BCI systems are defined. Steady state visual evoked potentials (SSVEP), P300 evoked potentials, slow cortical potentials (SCP), cortical – neuronal activation potentials and motor imagery (MI) are the most known BCI

methodologies. Among this methods, motor imagery is the most popular method and widely used in BCI applications [34, 35]. Motor imagery is a mental process by which an individual rehearses or simulates a given physical action [36]. A motor imagery based brain–computer interface translates a subject's motor intention into a command signal through real-time detection of motor imagery states, e.g. imagination of left and right hand movement [37]. The studies in this thesis are based on BCI systems with motor imagery. Detailed description about motor imagery among other methodologies will be given in the next chapter. In this thesis, new methods for classifying motor imagery signals are researched. Users' intentions about moving their limb reflects some activation changes in motor cortex area. There are so many methods for extracting and classifying this activation patterns in the literature. This thesis proposes new classification methods to the literature which will be analyzed deeply in the following chapters. Each contribution is proposed as a new methodology for a corresponding motor imagery classification concept. The outline of the thesis is as follows: In chapter 2, motor imagery and other BCI methodologies are described. Physiological sources of motor imagery signals and brain activation patterns are mentioned. Also some background about motor imagery signal classification is given. Chapter 3 introduces the methods used for motor imagery signal classification. It starts with common problems faced while working with MI signals and then, continues with the spatial filtering paradigm and the most popular method used in BCI systems, common spatial patterns (CSP). After that, spatial filter regularizing methods in the literature are described along with the firstly proposed method of this thesis: "task related & spatially regularized common spatial patterns. (TR&SR-CSP)" Then, another contribution of this paper about spatial filtering is presented: "spatial filter network (SFN)". Then, it introduces a newer paradigm called spatio-spectral filtering and gives some background about spatio-spectral filtering. Chapter 3 finally presents the last contribution of the thesis: "filter bank common spatio - spectral patterns (FBCSSP)". The computer simulations of the methods presented in the previous chapters are given in Chapter 4. This chapter introduces the data sources used in the MI signal classification and then, shows the results of the proposed methods compared with other state-of-art methods in their topics. Chapter 5 summarizes the thesis, describes the difficulties met, criticizes the proposed methods with their handicaps, gives examples about future works and finally concludes the thesis.

2. BRAIN COMPUTER INTERFACES

In respect to their electrophysiological properties, current BCI systems may be summed up in five groups:

- Steady state visually evoked potentials (SSVEP)
- Slow cortical potentials (SCP)
- P300 evoked potentials (P300)
- Motor imagery (MI)
- Cortical neuronal activation potentials

These BCI methods may be called dependent or independent according to the underlying neuronal mechanisms for the given modality. SSVEP is a dependent BCI while other four are independent BCIs [7].

A dependent BCI does not use the brain's normal output pathways to carry the message, but activity in these pathways is needed to generate the brain activity (e.g. EEG) that does carry it [7]. For example, a BCI system uses matrix of letters that flash one at a time, and the user looks directly at a specific letter so that the visual evoked potentials (VEP) recorded over the visual cortex at the interval of that letter is much larger than the intervals of other letters [38]. In this case, the brain's output channel is EEG, but the generation of the EEG signal depends on gaze direction, and therefore on extra ocular muscles and the cranial nerves that activate them. A dependent BCI is essentially an alternative method for detecting messages carried in the brain's normal output pathways: in the present example, gaze direction is detected by monitoring EEG rather than by monitoring eye position directly. While a dependent BCI does not give the brain a new communication channel that is independent of conventional channels, it can still be useful [39].

An independent BCI does not depend on brain's normal output channels in any way. Peripheral nerves and muscles are not the carrier of the transmitted message and

furthermore, any activity in these muscles and nerves is not needed for generation of the brain activity which involves the message. For example, a BCI system which uses P300 modality is an independent BCI. In the study of Farwell and Donchin [3], the user looks at a matrix of letters which flashes one at a time. The user waits for a specific letter to be flashed. When it is flashed, P300 evoked potential is generated in the brain. In this case, the output channel of the brain is EEG and the generation of the EEG signal depends only on the user's intent, not on the orientation of the gaze like SSVEP.

Peripheral nerves and muscles don't play a critical role on the independent BCIs. Independent BCI systems utilize totally nonconventional output pathways of the brain, so that, they are of greater theoretical interest than dependent BCIs. Also, for patients with highly severe neuromuscular disabilities who may lack all normal output channels (including ocular muscles), independent BCI systems are likely to be more useful.

2.1 Steady State Visual Evoked Potentials (SSVEP)

Steady state evoked potentials (SSEP) are sinusoidal vibrations observed in EEG when optical, auditory or somatic senses are stimulated with rapid and periodic stimulations. In case of Steady state visual evoked potentials based BCI, various stimulators (e.g. LEDs blinking in different frequencies) are displayed to the user in order to stimulate his/her occipital cortex [6, 40, 41]. When the user focuses on one of the blinking stimulators, the corresponding frequency component of the EEG signal received from occipital area will be much higher than that of the other blinking stimulators. Therefore, it will be very straightforward to find out which stimulator user had focused his attention on without any training. In Figure 2.1, sample EEG signal in time, frequency and time/frequency domains is shown while looking at a stimulator vibrating at 10 Hz frequency in a SSVEP BCI system.

Besides visually stimulated ones, BCI systems related with auditory (Steady State Auditory Evoked Potential, SSAEP) [43, 44] and somatic (Steady-state somatosensory evoked potentials, SSSEP) [45, 46] senses are developed in the literature.

Frequencies of stimulator used in SSVEP based BCI systems are generally less than 20 Hz. Also, maximum SNR is obtained at about 10 Hz which is inside the frequency

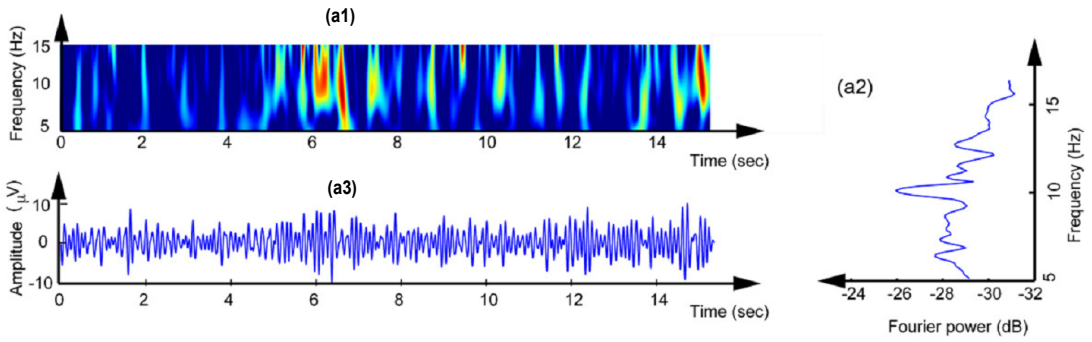

Figure 2.1 : SSVEP signal while looking at a target vibrating at 10 Hz. a1: time, a2: frequency, a3: time-frequency axis (From: Vialatte et.al., 2010 [42]).

band (8 – 12 Hz) of the alpha waves in occipital area. Although it is possible to get the SSVEP signal components above 10 Hz, separability of various frequencies reduces [42].

Thanks to its shorter response times, higher information transfer rate (>25 bits/minute), requirement of less electrodes than other BCI modalities and no requirement for user training, SSVEP based BCI systems has gained extremely wide application areas [47]. In the study of Muller et al. [6], a prosthesis hand was controlled in four directions using SSVEP. They reported that the developed prosthesis hand could execute a new movement between 2 and 5 seconds. In another study of Calhoun et al. [48], a flight simulator controlled with two buttons was realized by using SSVEP based BCI system. In this study, average decision time was measured as 2.1 seconds with 92% accuracy. Cheng et al. [40], realized a multi class BCI system involving 10 number and 2 control buttons. They reported that the maximum information rate was measured 27.15 bits/minute. Lalor et al. [49] controlled a computer game using SSVEP. In this BCI system with two classes, gamers were able to play the game with mean performance of 89.5%. These SSVEP based BCI systems are called as dependent BCI systems since they depend on the user's ability to control gaze direction. However, it was reported that, VEP amplitude in these systems reflects attention as well as gaze direction [50], and so that, they may be independent of neuromuscular function to some extent.

2.2 P300 Evoked Potentials

Infrequent or particularly significant auditory, visual, or somatosensory stimuli, when interspersed with frequent or routine stimuli, typically evokes a negative sudden peak at about 300 ms in the EEG over parietal cortex [51–53]. The ERP (event related

potential) called P300 or P3 wave is generally obtained in an interval between 300 ms and 500 ms after an infrequent stimulus occurs. The origin of P300 wave is unclear, some theories state that this phenomenon arises with the fetching of processed sensor data up to consciousness level [54, 55]. P300 differs from SSVEP since obtained waveform is a result of activities related to intra-processes of the brain, not directly related to the given stimuli. Also, it is possible to obtain P300 signal with various types of stimulators (visual, auditory, somatic). Generally, P300 based BCI studies uses oddball method. In this method, target stimulus is interspersed in routine stimulus (non target) and showed up non frequently and randomized. This non frequent target stimulus causes P300 waves to occur in the EEG signal.

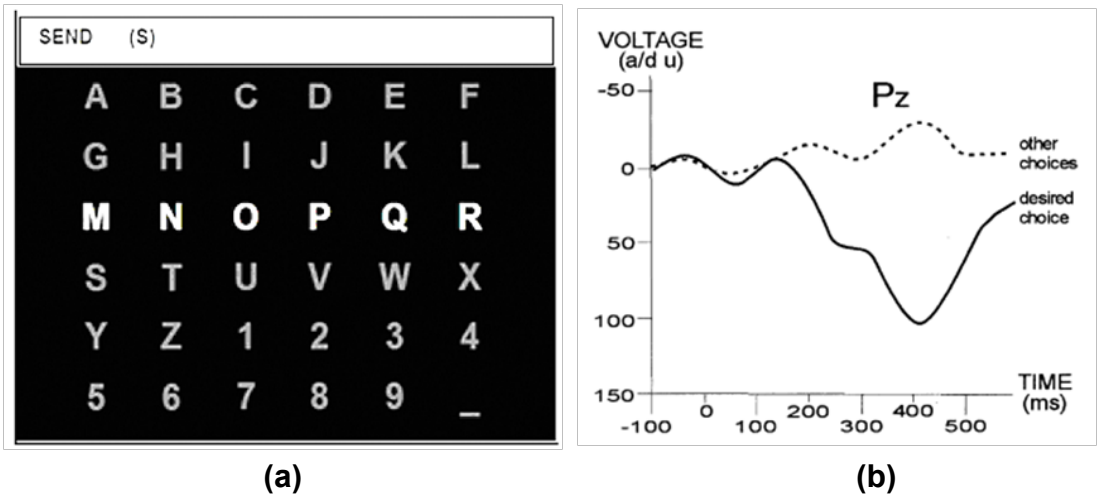

Figure 2.2 : a) P300 speller system showing matrix of letters (From: Farwell and Donchin,1988, [3]). b) P300 signal (From: Vialatte et.al., 2010 [42]).

BCI speller designed by Farwell and Donchin in 1988 [3] was based on this oddball method. Since its original design in 1988, the speller became the most used BCI speller in the world [56]. Speller interface is given in Figure 2.2. In this speller, letters and numbers forming a matrix with 6 rows and 6 columns are shown to the user on the computer screen. At every 125 ms, a row or a column of this matrix lights up. In order to create the P300 waveform, the user should select a character from the matrix and wait for that character to be illuminated. The selected character should be highlighted several times. When the rows and columns are being highlighted randomly, the user tries to count the number of times that specific character blinked so that he/she can focus on the matrix. While this process continues, EEG signal is collected from user's parietal cortex and average amplitude is calculated for each row and column separately. Finally, the amplitude for the column and row of the selected character will be higher

than that of other characters. Obtained cumulative P300 signal for selected character is shown in Figure 2.2.

As in SSVEP, no training procedure is needed in order to use a P300 BCI system. P300 wave is a natural reaction in all humans related to perceptual selectivity [7].

2.3 Slow Cortical Potentials (SCP)

Slow cortical potentials (SCP) are negative or positive voltage changes observed in scalp EEG and last for 0.5 to 10 seconds with amplitude of nearly 50 μV. As a result of Birbaumer and other researchers in similar area, it was observed that, with the help of visual and auditory feedbacks, not only healthy individuals, but also severely physically disabled patients may learn to control their SCPs in any direction [57]. Therefore, the voluntary control makes use of SCPs as a brain computer interface [58–61]. Birbaumer developed a device called thought translation device (TTD) with the help of SCPs. TTD was a computer program which helps the user to control his/her SCPs by providing auditory and visual feedbacks to the user. TTD was tested in people with last-stage amyotrophic lateral sclerosis (ALS) patients and has shown capability for basic communication skills [62]. In Figure 2.3, an experiment about slow cortical potentials is given [63]. The user in the experiment is able to control mouse pointer to the given target (cueN or cueP). When the target is shown, SCP is observed in one second period. In this example, two electrodes were used.

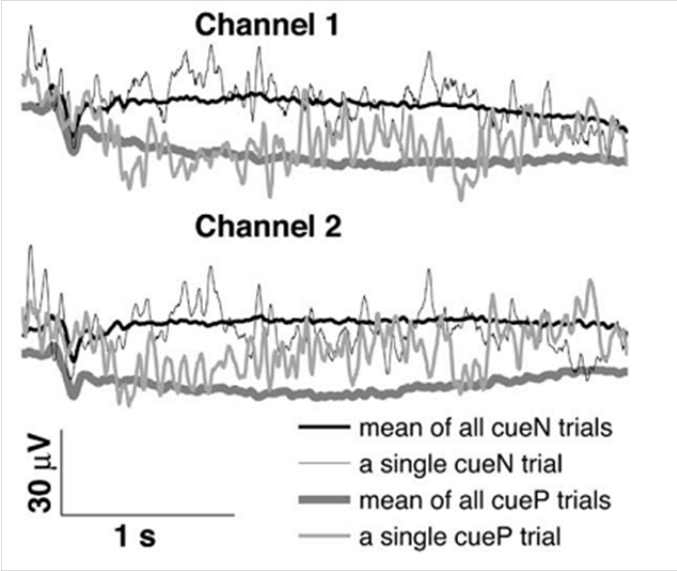

Figure 2.3 : Slow cortical potentials observed in an experiment including two types of tasks (cueN and cueP) (From: Mensh et.al. 2004 [63]).

2.4 Cortical Neuronal Activation Potentials

Since 1960s, it was possible to record the activation potentials of one single neuron during a physical activity of an animal by using metal micro-electrodes. Researchers had studied on the animals and tried to teach them controlling of firing of a single neuron. Some studies reports that monkeys was able to learn firing a single neuron on the motor cortex [64–67]. These studies gave hopes to develop neuro-prosthesises for physically handicapped individuals. However, placing of the recording electrode over the motor cortex is an invasive operation and contact of the placed electrode with the related neuron may be lost over time so that, this method was not widely accepted. In Figure 2.4, a representative image of intra-cortical electrode developed by Keneddy in 1989 is shown. Such an electrode was installed on monkeys and patients with ALS or brain stem injury and enabled researchers to receive neuronal signals stably over a period of one year [68, 69].

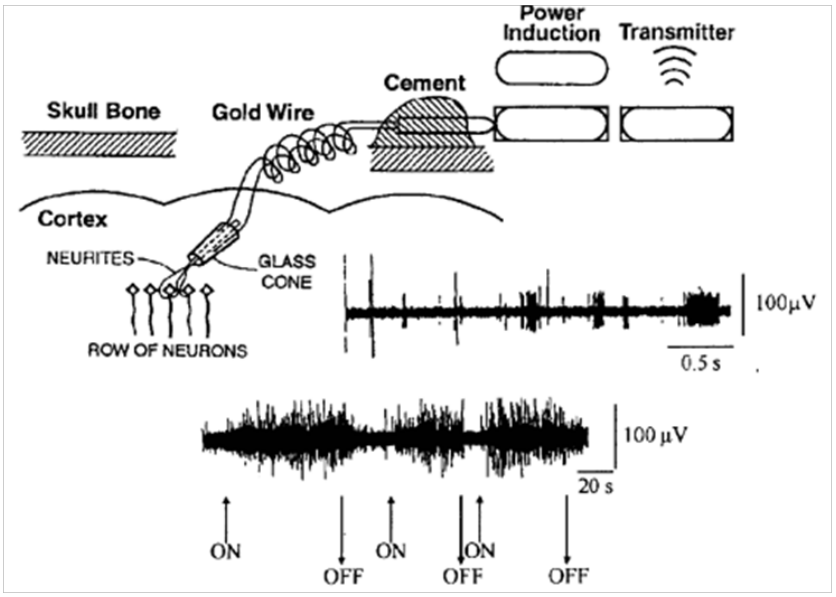

Figure 2.4 : Cortical neuronal BCI. A cone shaped electrode records one single neurons activation. The users were able to control the firing rate of that neuron in order to control a device (From: Wolpaw et.al. 2002 [7]).

2.5 Motor Imagery (MI)

A motor imagery based BCI translates subject's motor intention into a command signal by analyzing motor imagery states such as imagination of left and right

hand movement. According to the Khorshidtalab et al. [70], motor imagery electroencephalogram signals are the only bio-signals that enable locked-in patients, who have lost control over every motor output, to communicate with and control their surroundings. The following sections will introduce the parts of the brain related with motor imagery and nature of motor imagery signals by giving example studies in the literature.

2.5.1 Motor cortex

Human brain may be inspected in four parts which are frontal, parietal, occipital and temporal areas. Motor cortex is a part of the brain which is responsible for the planning, controlling and execution of the voluntary movements and it is found on the cerebral cortex on the frontal area. A figure representing the motor cortex is given in Figure 2.5. Motor cortex is divided in following sub areas:

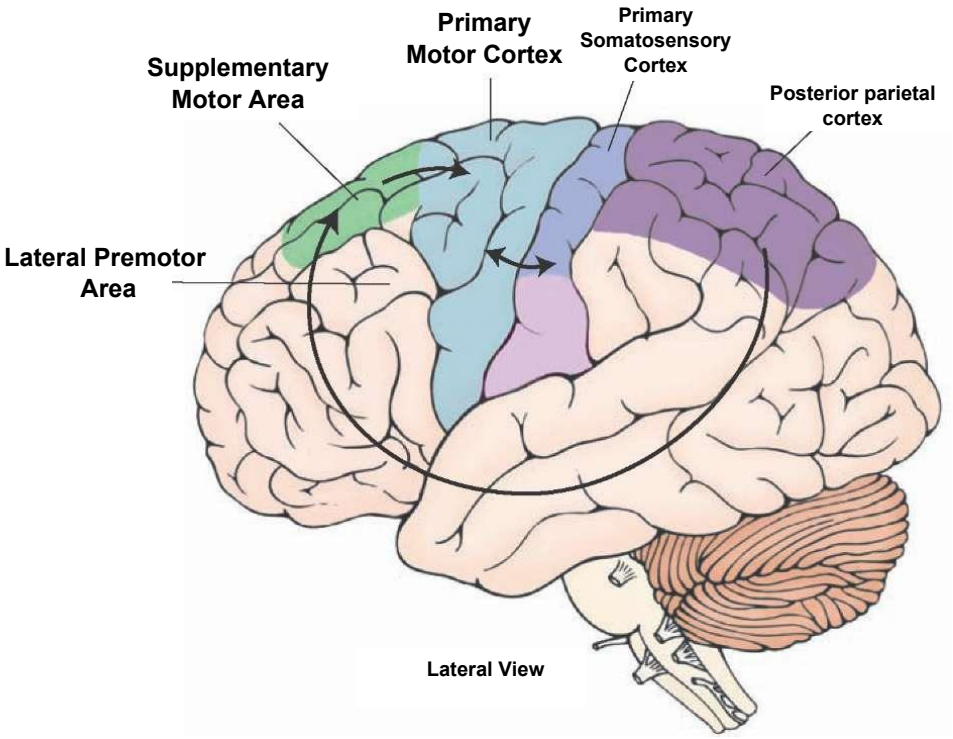

Figure 2.5 : Human Motor Cortex.

Primary motor cortex: Primary motor cortex is the main source of the neural impulses that pass down to the spinal cord and controls the execution of movement. However, some of the other motor areas in the brain also play a role in this function. The primary motor cortex contains cells with giant cell bodies known as "Betz cells". It is located on the area 4 of the precentral gyrus. Its location was confirmed in the

mid-20th century in brain operations performed by neurosurgeons such as Dr. Wilder Penfield, in Montreal. While performing operations to alleviate patients' epileptic symptoms, Penfield stimulated various areas of the cortex to identify vital ones that should not be removed. In this process, he discovered that stimulations applied to the precentral gyrus triggered highly localized muscle contractions on the contralateral side of the body and that there was a somatotopic representation of the corresponding parts of the body in Area 4 in the primary motor cortex.

Premotor Cortex: The premotor cortex is an area of motor cortex lying within the frontal lobe of the brain just anterior to the primary motor cortex. It has been studied mainly in primates, including monkeys and humans. The functions of the premotor cortex are diverse and not fully understood. It projects directly to the spinal cord and therefore may play a role in the direct control of behavior, with a relative emphasis on the trunk muscles of the body. It may also play a role in planning movement and in the spatial guidance of movement by integrating sensory information. Also, it controls the muscles that are closest to the body's main axis.

Supplemantary Motor Area: Penfield described the supplementary motor area (SMA), on the top or dorsal part of the cortex [71]. Each neuron in the SMA may influence many muscles, many body parts, and both sides of the body. The map of the body in SMA is therefore extensively overlapping [72]. Also, SMA projects directly to the spinal cord and may play some direct role in the control of movement. Roland [73] suggested that the SMA was especially active during the internally generated plan to make a sequence of movements. In the monkey brain, neurons in the SMA are active in association with specific learned sequences of movement [74]. Others have suggested that, because the SMA appears to control movement bilaterally, it may play a role in inter-manual coordination [75]. Also, it is suggested that, because of the direct projection of SMA to the spinal cord and because of its activity during simple movements, it may play a direct role in motor control rather than solely a high level role in planning sequences [76,77]. On the basis of the movements evoked during electrical stimulation, it has been suggested that the SMA may have evolved in primates as a specialist in the part of the motor repertoire involving climbing and other complex locomotion [78,79].

Besides these areas, other brain regions outside the cerebral cortex are also of great importance to motor function, most notably the posterior parietal cortex, primary somatosensory cortex, cerebellum, basal ganglia, red nucleus as well as motor nuclei.

The pioneer study of Penfield and Boldrey in 1937 revealed important information about spatial organization of motor cortex. They reported that, any muscle group in the body is presented at a specific area on the motor cortex [80]. The sizes of these areas are proportional with the usage skills of the corresponding limb rather than the limb's real size. The popular figure named *homunculus* resizes the limbs proportional to their areas occupied on the motor cortex which is shown in Figure 2.6. They obtained the size and the location of these regions by stimulating each area at a surgery.

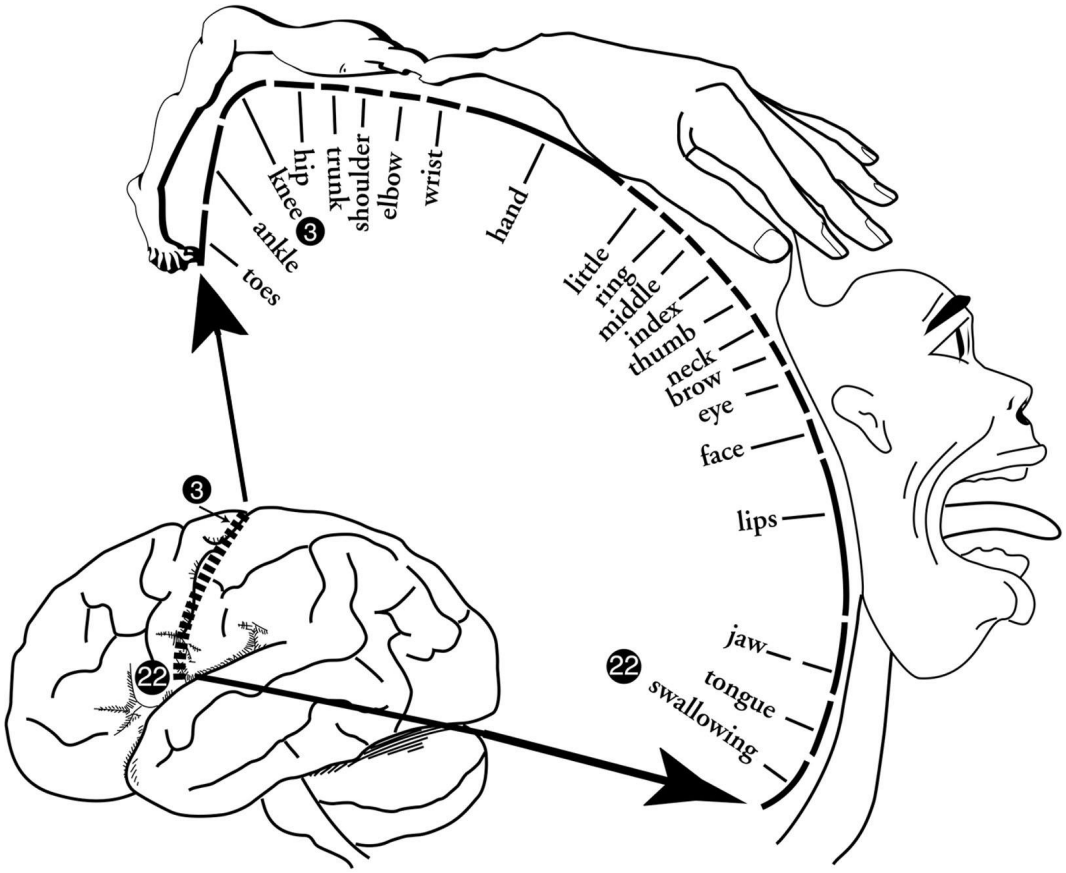

Figure 2.6 : The Homunculus figure obtained by Wilder Penfield, while stimulating various areas on motor cortex .(From: Sage, 1971. [81]).

2.5.2 Imagination of motor movement

A BCI design based on motor imagery perceives motor intention of a subject from his/her brain and trans-codes it to a digital command for the computer system. In motor imagery experiments, it was observed that there were variations in measured power

at special frequencies which are called μ (8-12 Hz) band and β (18-26 Hz) bands. In the first BCI design based on motor imagery [4], EEG signal related to the right and left hand imaginations were classified by analyzing the power of μ and β bands. Power decrease or increase may be considered to be due to a decrease or an increase in synchrony of the underlying neuronal populations, respectively. The former case is called event-related desynchronization or ERD [82, 83] and the latter event-related synchronization (ERS) [84]. Generally, ERD or ERS oscillations are observed with motor or mental activities or stimulations. These oscillations can be, e.g., due to modulating influences of neurochemical brain systems, changes in the strength of synaptic interactions, or changes of intrinsic membrane properties of the local neurons. The dynamics of brain oscillations associated with sensory and cognitive processing and motor behavior can form complex spatio-temporal patterns. For example, a synchronization of higher frequency components embedded in a desynchronization of lower frequency components can be found on a specific electrode location at the same moment of time. Simultaneous desynchronization and synchronization of 10-Hz components are possible on different scalp locations [26]. An example figure showing up ERD and ERS patterns is given in Figure 2.7 from the study of from Pfurtscheller et. al. (2001). In this figure, upper panel shows the band power obtained from C3 electrode at three different frequency bands (10-12 Hz, 14-18 Hz, 36-40 Hz) during right index filter lifting. Vertical line indicates the trigger of movement onset (t=0s). Note that, ERD starts 2.5s prior to the movement, peak gamma ERS shows up immediately prior to the movement onset and beta ERS appears within one second after the movement. The lower panel gives EEG time frame showing ERD and ERS signals for different electrodes regarding to upper tab.

Motor imagery can be thought as a mental preview of a movement without any physical output. It is widely accepted that, similar areas in brain are active during mental imagery or during execution of a movement. Therefore, they have the same functional relationship to the imagined or represented movement and the same causal role in the generation of this movement [85]. According to this concept, the imagery of a motor movement differs from execution of it only by being blocked at cortex - spinal level [86]. Recent functional brain imaging studies revealed similar activation patterns during mental imagery or physical execution of a movement by analyzing

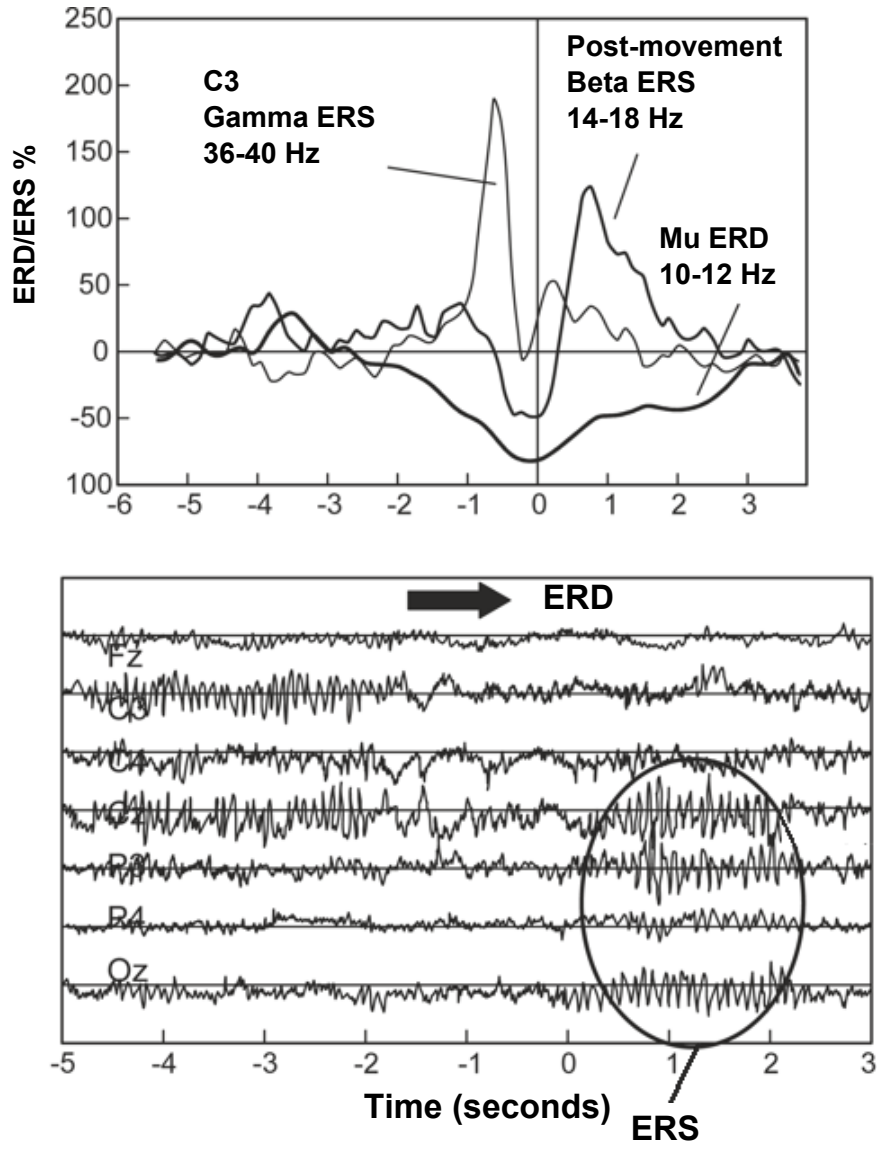

Figure 2.7 : An example figure showing ERD and ERS signals.

regional cerebral blood flow (rCBF). Roland [73] observed increased rCBF over supplementary motor area duing a sequenced finger movement experiment. Also, similar activation patterns were observed with imagination of various motor actions with positron emission tomography (PET) and functional magnetic resonance imagery (fMRI) [87, 88].

3. CLASSIFICATION OF MOTOR IMAGERY SIGNALS

This chapter describes basic concepts about motor imagery signal classification and addresses recent studies found in the literature. In this context, firstly, the problems generally encountered while dealing with the motor imagery signals will be briefly described. Then, a general framework regarding motor imagery signal classification will be presented and each processing step of the framework will be studied elaborately.

3.1 Properties of Motor Imagery Signals

According to the literature, it is possible to list the encountered difficulties that cause increasing misclassification rates while working with the motor imagery based brain computer interfaces.

3.1.1 Low spatial resolution

The spatial resolution is directly related with the number of electrodes placed over the head, increasing that gives higher resolution. However, this brings more computational burden and therefore, delayed response of brain computer interface. Besides, using more electrodes makes BCI an unpractical and uncomfortable design. There are some studies in the literature for optimizing the locations and number of placed electrodes to lower the computational load without decreasing the classification performance [89–92].

A better set of EEG channels with less number of electrodes helps eliminating irrelevant signals with motor imagery or noise so that, increasing the BCI performance while offering a more comfortable BCI experience to the subject. Arvaneh et.al. [89] proposed "sparse common spatial patterns (SCSP)" method for optimizing the channel selection. In the proposed method, they aimed minimal set of electrodes with the constraint of classification performance.

The volume conduction effect defined as the transmission of electric or magnetic fields from an electric primary current source through biological tissue towards measurement sensors [93], causes any electrode to acquire EEG signal variations not only from where it is located, but also from all over inside the head. The literature suggests numerous number of solutions to this problem [89, 91, 92, 94–99]. These methods obtain underlying signal sources by extracting independent components from EEG data.

3.1.2 Variations between subjects and sessions

Imagination of the same movement greatly varies between subjects, even between two sessions at different time for the same subject. The variability of ERD and ERS signal characteristics in spatial, spectral and temporal domains weakens the generalizing capability of the classifier. The same classifier may give high performance for one subject while showing poor performance for another subject. This guides us to the importance of training. In the training period, not only the classifier adapts its weights for the given subjects but also, the subject learns to control his/her ERD and ERS signals properly. It was reported that, classifier performance greatly increases with the training period [95, 100]

3.1.3 A few number of classes

In a motor imagery study, generally the movement intentions of different limbs are considered as different classes since all limbs are represented spatially at different locations in the homunculus given in Figure 2.6. Therefore, it is possible to differentiate each class or limb by eliciting the specific spatial location of the generated activation patterns. This gives the opportunity to be able to design high performance motor imagery brain computer interfaces easily [95]. However, the problem is that the number of classes to be classified is limited with the number of limbs to be imagined. An alternative to increase the number of classes is considering imagery of different types of actions of the same limb for different classes such as several movements of the wrist. However, it will be even harder to achieve classifying with low error since the activation patterns will be generated at the same spatial location.

Another solution to increase the class number may be imagination of different movements simultaneously. Vuckovic and Sepulveda [101] obtained six different classes by combination of four different movement imaginations that are extension / flexion of the wrist and pronation (down) / supination (up) movements of the palm. They reported that the performance of the BCI system was about 80%.

3.1.4 Non-stationary EEG

Non-stationary nature of EEG is another factor in misclassified motor imagery signal. In the literature, there are some methods for reducing the non-stationaries for motor imagery classification. As an example, Samek and his friends proposed "stationary common spatial patterns (sCSP)" method [102] which focuses on stationary components in the EEG signal in order to increase the generalizing capability and classification performance of the classifier.

3.1.5 Inexperienced subjects

In a motor imagery study, generally the subjects are chosen from those who had never attained to a BCI experiment for providing improvement in the objectivity of the results. However, the subjects are inexperienced in the way of imaging the movements that is presented to him/her. They can not easily comprehend imaging a movement like *feeling* it. For example, they just imagine a visual scene containing the requested movement. In this case, the BCI acquires EEG signals that are not related with the usual motor imagery activation patterns, which may mislead the training system. Therefore, preceding to the experiment, a training period is needed for proper imaging of a motor action. Feedbacks may be given to the training subject for faster learning. For example, they may receive auditory and/or visual alerts when they succeed to imagine properly [100] .

3.2 A General Framework for Classification of Motor Imagery Signals

A basic diagram regarding classification of a motor imagery EEG signal is presented in Figure 3.1. This framework consists of six processing blocks which are listed as i) motor imagery experiment design block, ii) signal acquisition block, iii) signal preprocessing block, iv) spatial filtering block, v) feature extraction block and vi)

classification block. In the following subsections, all processing blocks of the given framework will be described in details.

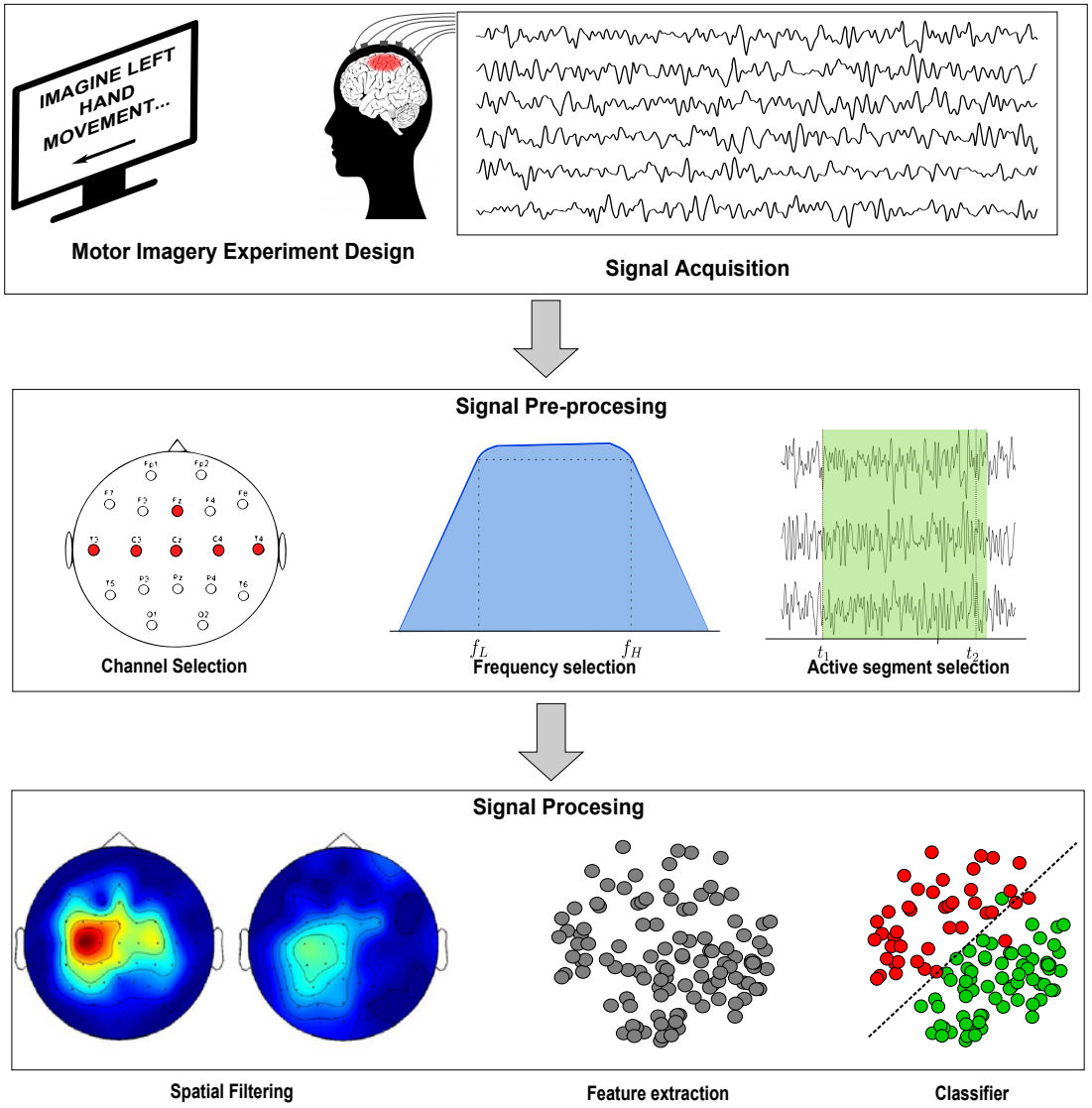

Figure 3.1 : A general flow diagram about motor imagery classification.

3.2.1 Motor imagery experiment design

Imaging of same motor movement varies fairly over different subjects in spatial and spectral domain. Therefore, varying spectral and spatial distributions of ERD and ERS signals make the classifiers generalizing ability to reduce. When using a feature set for a subject gives high classification performance, while the same classifier with the same feature set may fail to give high accuracy. Therefore, it was observed that subject training had been highly important in motor magery classification. Vuvckovic [95] and Hwang [100] reported considerable increased classification accuracy when studying with trained subjects in reference to untrained subjects.

In order to train a subjcet and the classifier with the subject's training data, an experimental set up is needed. Generally, a visual set up is prepared and presented to the user on the computer screen. The experiment software cues the subject about the movement to be imagined at random intervals while recording brain signals at the background continuously. Not only the brain signal, but also the label (i.e. *left hand movement* , *right hand movement*, etc..) and the time index of the presented cue are stored for post processing. As an example, Figure 3.2 illustrates the motor imagery trial from the experiment described in Gouy et.al. [5] . The trial begins with a fixation cross for preparing the subject for the cue to be presented. Next, the cue that indicates the movement to be imagined is shown on the screen for nearly 1 seconds. After the cue, user imagines according to the given cue. However, timing of this period is unclear, it should be in an interval inside the motor imagery period. The trial ends with a break indication on the screen and the subject waits for the next trial. At the end of the motor imagery experiment, numerous trials are recorded so that a rich training set for the classifier is obtained.

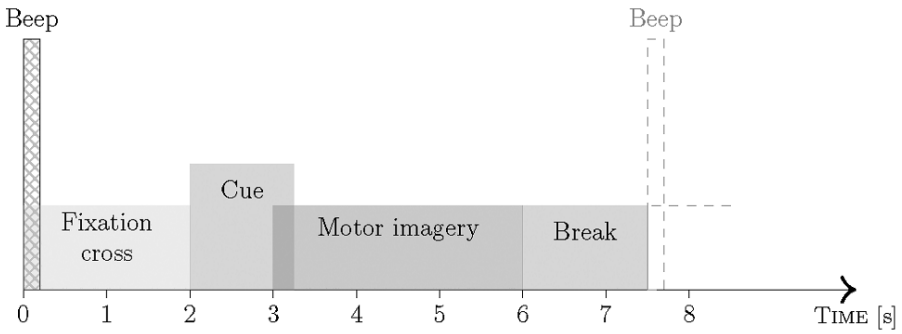

Figure 3.2 : A trial of a motor imagery experiment.

3.2.2 Motor imagery signal acquisition

In a BCI system, the activity of the brain is measured and then converted to the control commands for the controlled device. There are many techniques for measuring the brain activity such as functional magnetic resonance imaging (FMRI), Magnetoencephalography (MEG), Positron Emission Tomography (PET), Electrocorticogram or electroencephalography (EEG). Among them, EEG is the preferred way of acquiring brain signals because it has short time constants which means shorter reaction time and requires relatively simple and inexpensive equipment [103–105].

When placing the EEG electrodes over subject's head, generally, standard 10-20 system is used. The 10–20 system or International 10–20 system is an internationally recognized method to describe and apply the location of scalp electrodes in the context of an EEG test or experiment. This system is based on the relationship between the location of an electrode and the underlying area of cerebral cortex. The "10" and "20" refer to the fact that the actual distances between adjacent electrodes are either 10% or 20% of the total front–back or right–left distance of the skull. Figure 3.3, displays standart 10-20 electrode locations with 10% division, based on the guide from American Electroencephalographic Society [106].

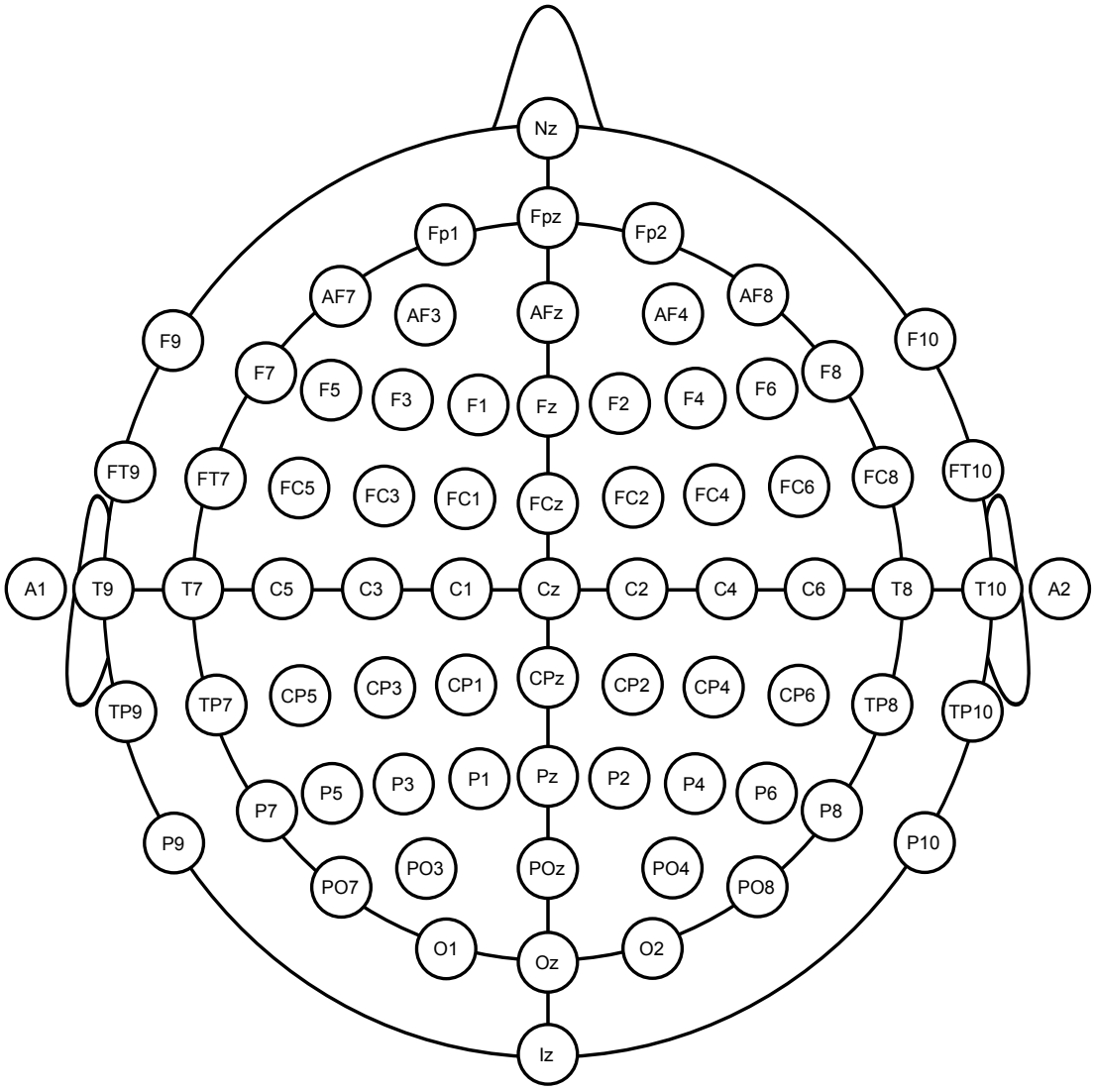

Figure 3.3 : EEG electrode locations acording to the international 10-20 standard.

Each site has a letter to identify the lobe and a number to identify the hemisphere location. The letters F, T, C, P and O stand for frontal, temporal, central, parietal, and occipital lobes, respectively. (Note that there exists no central lobe; the "C" letter

is used only for identification purposes.) Even numbers (2,4,6,8) refer to electrode positions on the right hemisphere, whereas odd numbers (1,3,5,7) refer to those on the left hemisphere. A "z" (zero) refers to an electrode placed on the midline. In addition to these combinations, the letter codes A, Pg and Fp identify the earlobes, nasopharyngeal and frontal polar sites respectively.

The amplitude of the EEG is about 100 μV when measured on the scalp, and about 1-2 mV when measured on the surface of the brain. The bandwidth of this signal is from under 1 Hz to about 50 Hz, as demonstrated in Figure 3.4 .

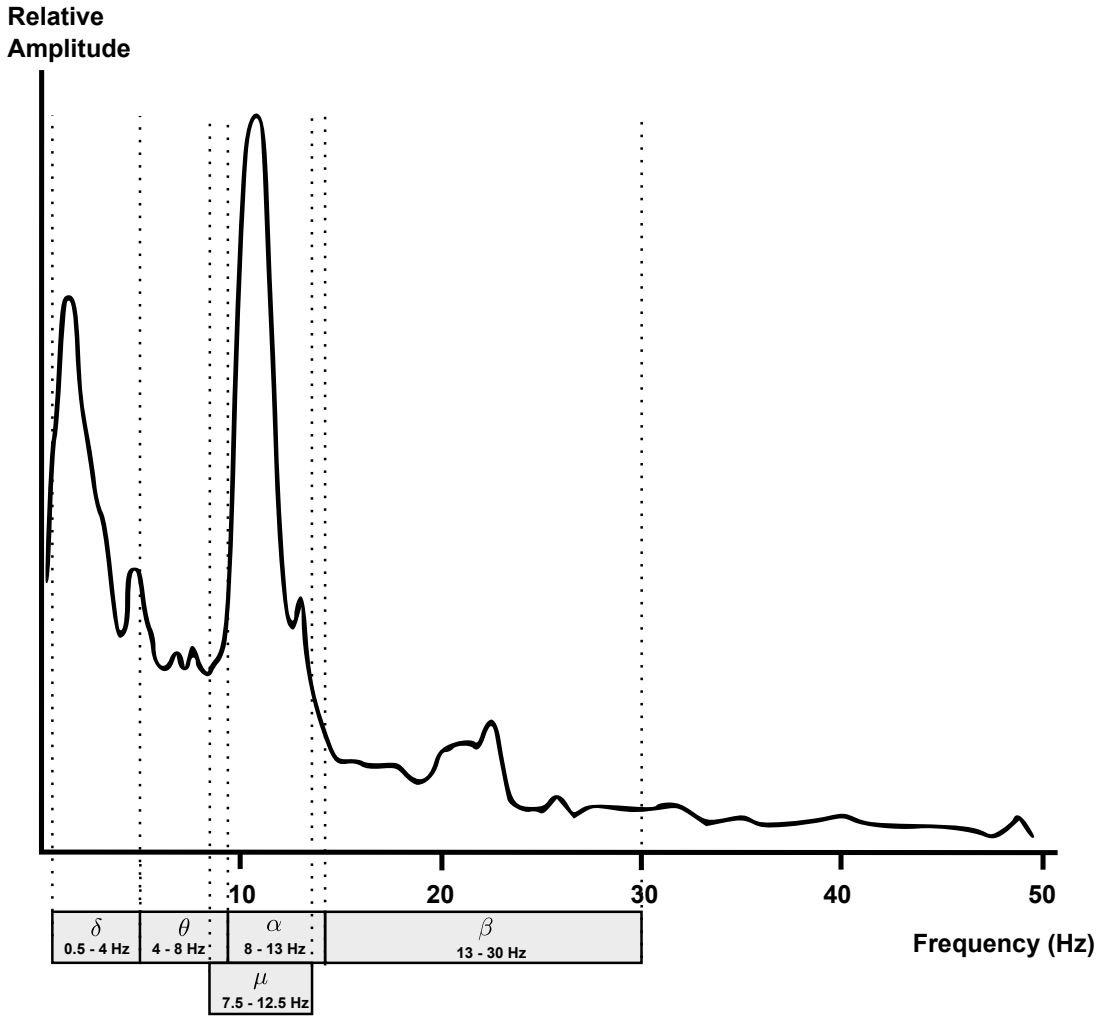

Figure 3.4 : Frequency spectrum of EEG signal.

EEG signal spectrum is partitioned into special frequency bands called alpha (α), beta (β), delta (δ) and theta (θ) waves. The alpha waves have the frequency spectrum of 8-13 Hz and can be measured from the occipital region in an awake person when the eyes are closed. The frequency band of the beta waves is 13-30 Hz; these are detectable over the parietal and frontal lobes. The delta waves have the frequency range

of 0.5-4 Hz and are detectable in infants and sleeping adults. The theta waves have the frequency range of 4-8 Hz and are obtained from children and sleeping adults [107]. Also, mu waves (μ) , also known as sensorimotor rhythms, are synchronized patterns of electrical activity involving large numbers of neurons in the part of the brain that controls voluntary movement. These patterns as measured by electroencephalography (EEG), magnetoencephalography (MEG), or electrocorticography (ECoG), repeat at a frequency of 7.5–12.5 (and primarily 9–11) Hz, and are most prominent when the body is physically at rest [108]. Unlike the alpha wave, which occurs at a similar frequency over the resting visual cortex at the back of the scalp, the mu wave is found over the motor cortex.

3.2.3 EEG data preprocessing

In the preprocessing phase, acquired EEG signal is *selected* in terms of channels, frequency and time due to wipe out information unrelated with motor imagery. This phase transforms the EEG signal with the help of prior knowledge about motor imagery. Generally, preprocessing phase includes the following items:

3.2.3.1 Channel selection

It is known that, motor cortex is an area of the frontal lobe located in the dorsal precentral gyrus immediately anterior to the central sulcus which corresponds to the area around electrodes C5 through C6 according to the 10-20 electrode standard. Therefore, a motor imagery classification system should mainly use those electrodes. Also, it is a common practice to select a wide area that coarsely cover the motor cortex because of the volume conduction effect which is defined as the transmission of electric or magnetic fields from an electric primary current source through biological tissue towards measurement sensors [93]. For example, Samek et. al. had used a total of 68 EEG electrodes which roughly cover the motor cortex [102].

There are many methods in the literature that apply automatic channel selection methods prior to the classification. Kim et. al. [109] used a binary particle swarm optimization (BPSO) as an optimal channel selection method. Another work submitted by He et.al. [110] suggests genetic algorithm based optimal channel selection method for motor imagey classification. Indeed, channel selection is not a critical issue for

classification accuracy since spatial filtering methods which will be covered in the following sections assign a weight for each channel so that redundant channels are eliminated automatically by having smaller weights.

3.2.3.2 Frequency selection

As reported in Wang et. al. [37], electroencephalogram (EEG) signals accompany power changes in movement related μ (8–12 Hz) and β(18–26 Hz) rhythms, representing a power increase or decrease named event-related desynchronisation and sychronisation (ERD/ERS) in specific motor cortex areas during motor imagery. Figure 3.5 shows frequency spectrum of EEG on channel CP4 during motor imagery of left and right hands.

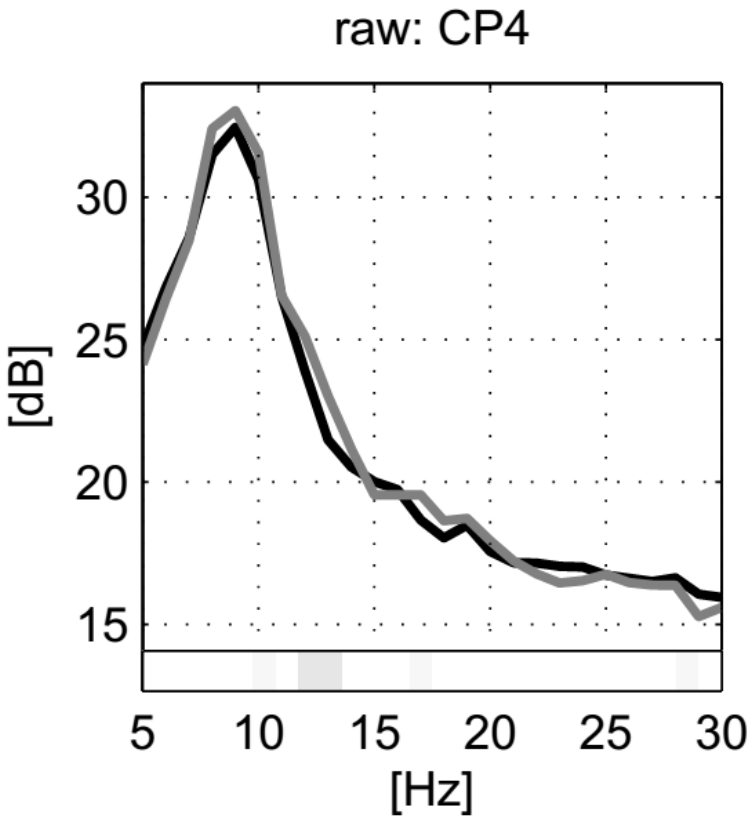

Figure 3.5 : Frequency spectrum of EEG signal during motor imagery.

For a motor imagery classification application, a filter with a broad pass band is sufficient most of the time because it encompasses the alpha and beta frequency bands [111]. Müller-Gerking et.al. compared differed frequency bands and reported higher classifying performance when using a broadband filter with 8 - 30 Hz pass band [112]. However in practice, frequency band which reflects ERD varies from subject to subject [4], the filter should be tuned adequately, which is one of the most

challenging issues when designing a practical BCI [113]. To focus on the ERD and ERS signals and to achieve a high classification performance, a filter with narrower pass band is required. This brings the automatic optimal frequency selection problem, which will be covered in the following sections by details.

3.2.3.3 Active segment selection

In order to train a classifier and measure its performance, a training and a test set is necessary. As described in the previous section, in a motor imagery experiment, brain signals are continuously recorded as well as the time indexes of the markers such as *start of trial*, *start of cue*, etc.. Most of the methods in the literature including those submitted by this thesis assume trial based motor imagery classification, which use a number of multi channel EEG time segments called *epoch*. For creating the epoches from a given EEG recording, one needs to determine the time offset and window width of the active segment where the subject imagines the motor movement. It should be noted that, starting of the imagery period can not be determined exactly since it varies from trial to trial for any subject. Therefore, an average window should be set. As an example, in Figure 3.6, ERD signals over C3 and C4 electrodes for two types of motor imageries are displayed. This image was created by averaging many trials over and over. The zero on time axis indicates the start of cue. It is clearly seen that, there is a decrease in relative power from 1000 ms to 4000 ms which is shaded gray.

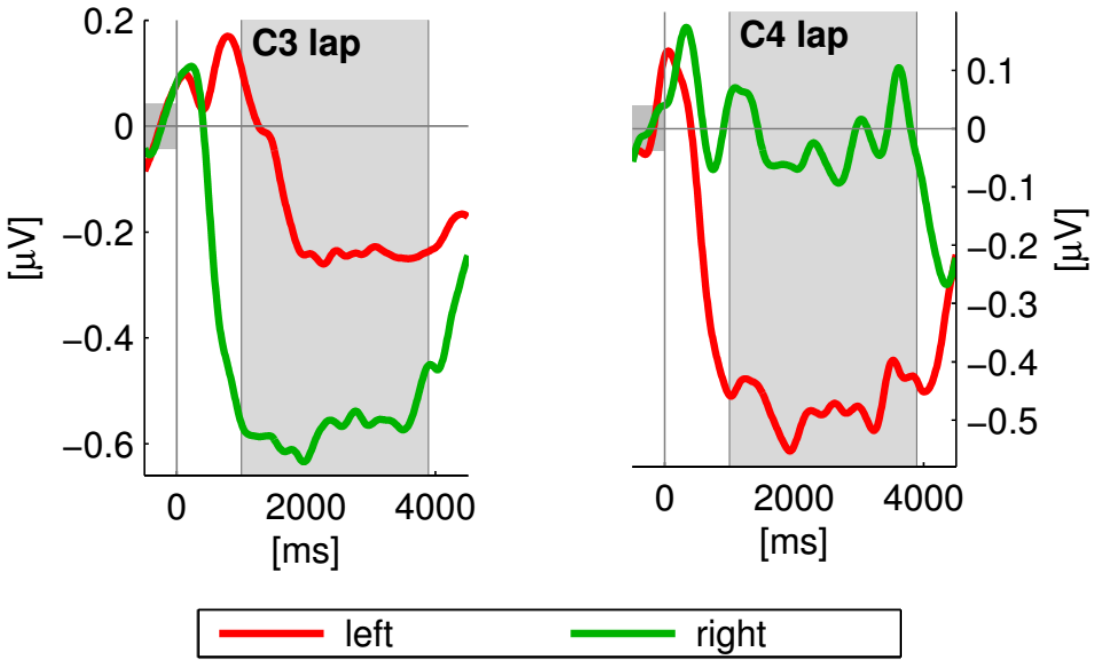

Figure 3.6 : Determination of motor imagery active segment.

The ERD/ERS is defined as the percentage power decrease (ERD) or power increase (ERS) in relation to a one-second reference interval (1 seconds) before the warning tone [114]. The duration of μ-ERD after the movement onset is strongly related with the duration of the kinesthetic execution of real movement [115]. Generally, active segment is determined manually in a motor imagery experiment for extracting epochs. For example, Naeem et.al. [116] considered the EEG record between 2.5 -3.5 seconds after the indication of the cue. Also, Hsu and Yen [117] proposed an automatic method for active segment selection using continuous wavelet transform (CWT).

3.2.4 Spatial filtering for EEG data processing

As stated in the previous chapter, any muscle group in the body corresponds to a specific area on the motor cortex. Such a topographical organization yields classification of different motor imagery tasks based on related ERD rhythms' specific spatial location on the motor cortex. Therefore, a good spatial resolution is needed in order to determine the exact location of the underlying ERD signal source. However, due to the volume conduction effect, scalp EEG signals recorded from a specific area involve a mixture of several cortical sources located in different areas. Thus, raw scalp EEG potentials have poor spatial resolution [118]. An illustration regarding volume conduction on scalp EEG [119] is given in Figure 3.7.

According to Dornhege et.al. [120], there are two problems when using ERD features for BCI control: (a) The strength of the sensorimotor idle rhythms as measured by scalp EEG is known to vary strongly between subjects. This introduces a high intersubject variability on the accuracy with which an ERD-based BCI system works. (b) The precentral μ-rhythm is often superimposed by the much stronger posterior α-rhythm, which is the idle rhythm of the visual system. Since the μ and α rhythms have similar spectral features, identifying the ERD rhythm from idling rhythm of the visual system is not easy without determining the location of the underlying signal source. Thus, to eliminate the volume conduction effect and reach the actual underlying signal sources, a spatial filtering step is an indispensable technique [121].

The main goal of spatial filtering in a BCI system is removal of blurring effect caused by the volume conduction effect, called deblurring [122]. A sophisticated example proposed by Jian Le and Alan Gevins [123] who attempted to deblur

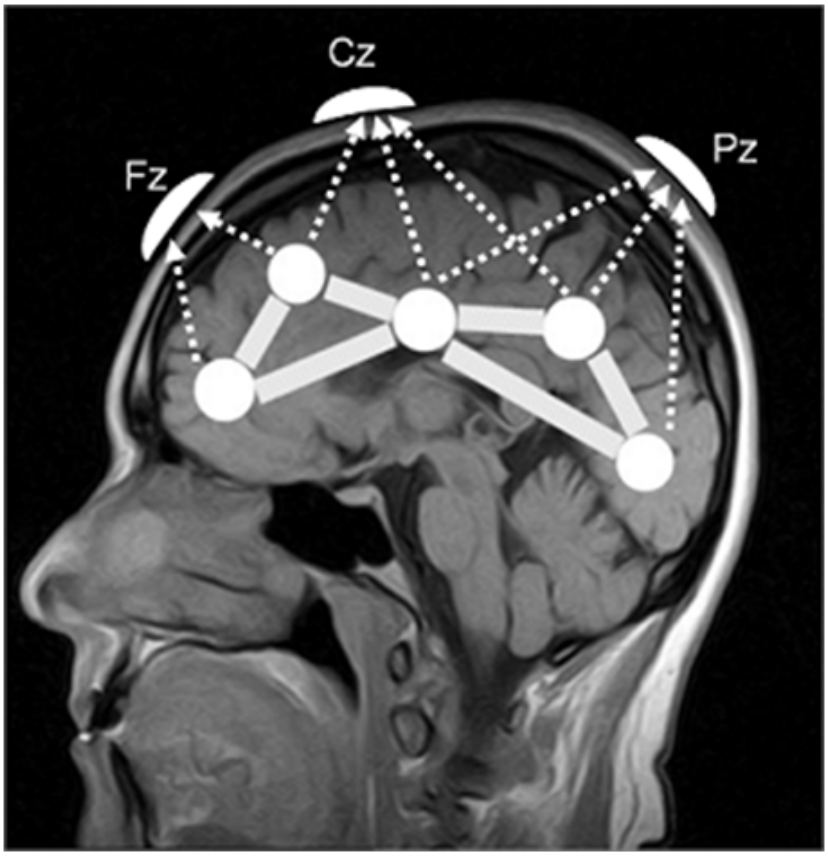

Figure 3.7 : Volume conduction effect in scalp EEG. [119]

signals using a realistic biophysical head model that is specific to each subject and whose parameters are derived from magnetic resonance imaging (MRI). While these approaches do increase signal quality in carefully controlled experiments, they are currently impractical for most experiments and for clinical applications.

Common average reference (CAR) and Laplacian method (LAP) are simple spatial filtering methods. In the CAR method, the average value of the entire electrode montage (the common average) is subtracted from the channel of interest. CAR provides EEG recording that is nearly reference-free. Because it emphasizes components that are present in a large proportion of the electrode population, the CAR reduces such components and thereby functions as a high-pass spatial filter (it accentuates components with highly focal distributions) [124].

The Laplacian method calculates for each electrode location the second derivative of the instantaneous spatial voltage distribution, and thereby emphasizes activity originating in radial sources immediately below the electrode [125]. Thus, it is a high-pass spatial filter that emphasizes localized activity and reduces more diffuse activity. High spatial resolution can be achieved by using many electrodes (e.g.

64) spread over the entire scalp. The value of the Laplacian at each electrode location is calculated by combining the value at that location with the values of a set of surrounding electrodes. The distances to the set of surrounding electrodes determine the spatial filtering characteristics of the Laplacian. As distance decreases, the Laplacian becomes more sensitive to potentials with higher spatial frequencies and less sensitive to those with lower spatial frequencies [124].

Independent component analysis (ICA) has been used in EEG signals to decompose brain signals into statistically independent components [126] for accessing the underlying signal sources. ICA is an unsupervised algorithm, so there is no previous knowledge about the class labels of the motor imagery data [127]. Furthermore, while ICA optimizes statistical independence of brain signals, and can thus lead to more compact signal representations, it does not guarantee to optimize the discriminability of different brain signals in different tasks [122].

Common spatial patterns (CSP) is a very popular and powerful spatial filtering method used in motor imagery EEG classification. As a a highly successful spatial filtering algorithm, as evidenced by BCI Competition II and III [128, 129], the CSP (also known as Fukunaga–Koontz transform in the machine learning field) which was firstly proposed as an application of the Karhunen–Loève transform for classification by Fukunaga et al. [130], and was later introduced for use in BCI systems by Ramoser et.al. [111]. When using band power features, CSP computes spatial filters, aiming to obtain optimal discrimination between two classes [102]. CSP finds optimal spatial filters that maximize the ratio of average variances that belong to two different classes. Computationally, CSP is solved by simultaneously diagonalizing the two covariance matrices of the two classes [131]. A computed CSP spatial filter projects the multi-dimensional EEG time domain signal to a one-dimensional time domain signal in which the power (variance) of one class is maximized while the power of the other class is minimized. Unlike PCA, CSP handles two classes at the same time and simultaneously diagonalizes the covariance matrices of both classes [132]. With respect to the topographic patterns of brain rhythm modulations the CSP algorithm has proven to be very useful to extract subject-specific, discriminative spatial filters [120].

Although CSP is a powerful technique, with simplicity, it has some drawbacks. Due to its optimization function which tries to maximize the ratio of variances, CSP algorithm

is very sensitive to outliers which cause over fitting [133]. Recently, some methods called *regularized common spatial patterns* (RCSP) have been proposed which aims computing more robust spatial patterns by adding a regularization term to the CSP formula [102, 121, 134]. The RCSP method uses some a priori knowledge and imposes various constraints in the CSP's formulation to obtain more robust spatial filters [135]. For example, Lotte et.al., [134] proposed spatially regularized CSP (SRCSP). He used the a priori knowledge that neighboring neurons tend to have similar functions, which supports the hypothesis that neighboring electrodes should measure similar brain signals. Another example of regularizing CSP is stationary common spatial patterns (sCSP), which was proposed by Samek et al. [102]. sCSP assumes that non-stationeries in EEG data arises as a result of the processes that are not task related, such as eye movements or electrode artifacts. Another study on robust CSP is CSP-L1 by Wang et al. [131], who attempted to express the CSP formulation in L1 norm. He states that the original formulation of CSP was L2-norm, which implies that CSP was sensitive to outliers. Wang attempted to optimize the proposed alternative CSP formulation using an iterative algorithm.

As an original RCSP method, *task related & spatially regularized common spatial patterns* (TR&SR-CSP) method [136] is the first contribution of this study. Proposed method unifies prior information about executed motor imagery tasks and spatial locations of EEG channels. In the proposed method, we designed a regularizing method which emphasizes the electrodes that are spatially close to the center of imaging task being executed. TR&SR-CSP method regularizes the CSP spatial filter by directing it towards to center electrodes, which is assumed to record task-related signals.

CSP optimizes the average power ratio of the two classes, and therefore, it requires only one average covariance matrix for each class. This may be a problem while handling non-stationary signals such as EEG because the covariance matrix of an EEG signal may change over time due to artifacts such as changes in EEG electrode-skin impedances, muscular activities or user background EEG activity [102]. Representing all of the epochs of a class in a training set by only one average covariance matrix should result in inaccurate spatial filters. Another disadvantage of CSP is its strict fitness function. CSP does not allow different types of fitness functions, which may

be more useful in different situations [135]. CSP attempts to optimize the Rayleigh quotient, i.e., the ratio of average variances of the two classes, which is very sensitive to outliers that cause over fitting [133]. Another contribution of this study is a spatial filtering method called *Spatial Filter Network* (SFN) [137] which addresses these problems and proposes a flexible spatial filter network which unifies spatial filtering and classifying in a classical multi layer neural network topology.

To focus on ERD and ERS signals and to achieve a high classification performance, it is necessary to filter the EEG signal with a band pass filter prior to CSP calculation. However, one problem is that, the frequency bands of these signals vary from subject to subject. Generally, the cut-off frequencies of the band pass filter are either selected manually or unspecifically set to a broad band filter [120], which results in poor classification performance. Manual searching of the best frequency band through the training set is laborious and time consuming [113]. Thus, optimizing a spectral filter along with the spatial filter is highly desirable [120].

Common spatio-spectral pattern (CSSP) algorithm [138] is the firstly proposed method to address this problem. CSSP embeds time delayed channels into the original EEG signal in order to create a first order FIR filter for each channel. Obtained results showed an improvement of the CSSP algorithm over CSP. However, a first order FIR filter is very limited to select a certain frequency band from the EEG spectrum. After that, an improvement to CSSP, common sparse spectral spatial pattern (CSSSP) was proposed [120]. CSSSP designs a FIR filter with any order common to all channels. Proposed method searches for a set of spectral-spatial filter coefficients by gradient search method which is computationally expensive with additional cost for sparsification and it needs some parameter tunings.

Sub-band common spatial patterns (SBCSP) [113] and filter bank common spatial patterns (FBCSP) [16] methods are based on optimizing spatial filters for multiple spectral filters that have different pass-bands. As reported in BCI competititon III and IV, FBCSP method achieved a high classification accuracy [129]. In these methods, a filter bank is used in order to decompose EEG signal into multiple frequency bands and a separate spatial filter is calculated for each band by CSP method. Then, features belong to different frequency bands are chosen by feature selection methods based on mutual information maximization.

Spectrally weighted CSP (SPEC-CSP) [139] and iterative spatio – spectral patterns learning (ISSPL) [18] algorithms design a spectral filter which works in frequency domain. Methods alternatively optimize spatial and spectral filters, where the spatial filter is optimized by CSP algorithm and the spectral filter is optimized by an optimization algorithm based on a cost function. Since the cost functions of spectral and spatial optimization problems are different, alternative solving method is not guaranteed to be converged [140]. Higashi et.al. recently proposed a method for simultaneous design of spectral and spatial filters [140] called discriminative filter bank CSP (DFBCSP). DFBCSP algorithm optimizes the coefficients of FIR filter(s) and corresponding spatial weights. DFBCSP proposes an iterative method to optimize the spatial and spectral filter coefficients by converting the spatial and spectral optimization problems into separate generalized eigenvalue problems. Since it is an iterative method, reaching the optimum point should take many steps and optimization speed of the DFBCSP method depends on the degree of the FIR filter to be optimized.

In this study, we also proposed another spatio-spectral filtering method which binds the spatial and the spectral filters in a mixed architecture that we call *filter bank common spatio - spectral patterns* (FBCSSP). FBCSSP finds out the required filter parameters with simple CSP calculations in one pass, without any iteration. FBCSSP method uses a network of a filter bank and two consecutive CSP layers so that proposed structure has a subject specific response in both spatial and spectral domains.

3.2.5 Feature extraction methods

The purpose of the feature extraction block in the framework given in Figure 3.1 is to convert the time domain EEG signal into a set of feature vectors in the defined feature space with maximum between class and minimum intra-class scatterings in order to maximize the Fisher's criterion [141].

There are numerous features used in the context of brain computer interfacing such as, power spectral density (PSD) [142, 143], EEG signal amplitude feature [144], band power (BP) [46, 145, 146], time-frequency features [147] , autoregressive (AR) parameters [148, 149] , and inverse model-based features [150–152]. Lotte [153] reported the following critical properties of features as listed below:

- **Noise and outliers**: Since EEG has a low signal to noise ratio, features are noisy and contains outliers. Therefore, robust methods are preferred for spatial filtering and feature extraction.

- **High dimensionality**: In BCI systems, the dimension of the feature vectors are often high. Spatial filtering and feature selection methods are used for dimension reduction.

- **Non-stationarity**: Obtained features is non stationary and vary over time or over sessions due to some factors such as electrode-skin impedances, muscular activities or user background EEG activity.

- **Small training sets**: The training sets are relatively small, since the training process is time consuming and demanding for the subjects

Imagined movements have been shown to produce changes in the μ (i.e., 8–12 Hz) or β (i.e., 18–25 Hz) frequency band. Thus, for the processing of μ/ β rhythm (i.e., sensorimotor) signals, the frequency domain is most often used [122].

The common spatial patterns algorithm organizes spatial filter coefficients which maximize the power for one class while minimizing it for the other which increases the discriminability of the two classes when using band power features for classification [154]. Therefore, when using a spatial filtering algorithm, band power is used as a feature extraction method, generally. In this study, we also validated the discriminability of the features when band power is used as a feature extraction method subsequent to the spatial filtering algorithm.

3.2.6 Classification methods for EEG data

In the presented framework give in Figure 3.1, the role of the spatial filtering and feature extraction blocks together is to obtain the features that have a good separability in the feature space. Research in motor imagery classification is mainly focused on optimizing spatial filters and extracting good features that are stationary as much as possible and have a high divergence. Therefore, a linear classifier is enough for finding the class boundary and labeling the unknown epochs.

Linear discriminant analysis (LDA) [141] classifier is used in great number of motor imagery based BCI systems [114]. LDA uses hyperplanes to separate the data representing the different classes. For a problem with two classes, the class label of a feature vector depends on which side of the hyperplane the vector is [155].

Support vector machines (SVM) classifier [156] is also commonly used with the motor imagery classification problems [157]. As in LDA, SVM uses a hyperplane to identify the class labels. However, the selected hyperplane in SVM is the one that maximizes the margins, that is defined as the distance from the nearest training (supporting) vectors. Maximizing the margins increases the generalization capabilities [158], which is an advantage for motor imagery classification.

Besides LDA and SVM, there are other classifiers used in BCI research. For example Obermaier et.al [159] used Hidden Markov Model (HMM) based classifier for classifying single trial motor imagery EEG data. They reported that, HMM based classifier is suitable for online classification with lower classification error than linear classifiers. Multilayer perceptron (MLP) had been also applied for motor imagery problems such as binary [160] and multiclass [161] problems. The fact that MLP can classify any number of classes, this makes this neural network a very flexible classifier that can adapt to a great variety of problems [153]. Other types of classifiers such as fuzzy hopfield neural network (FHNN) [162], learning vector quantization (LVQ) [26], radial basis function neural network (RBFNN) [163], fuzzy ARTMAP neural network [164], adaptive logic network (ALN) [165], finite impulse response neural network (FIRNN) [166], time-delay neural network (TDNN) [167], even simplest algorithms such as K-nearest neighbor (KNN) classifier [168] had been used for classification in the motor imagery based brain computer interfaces.

3.3 Common Spatial Patterns (CSP)

CSP is a spatial filtering method used frequently for discriminating separate motor imagery classes and increasing the spatial resolution of the acquired signals. Let a motor imagery experiment involving trials with two classes coded with C_1 and C_2 (i.e left and right hand imaginations). In this experiment, we call each single frame of acquired EEG signals as an *epoch*. Let a multi dimensional signal matrix $X_k \in \mathbb{R}^{NxT}$ represent an epoch, k is the epoch number and $k \in C_1$ or $k \in C_2$, N is the number of EEG

channels, and T is the number of samples in the epoch. An example epoch is given in Figure 3.8. Note that X_k should be a zero average signal (i.e., band pass filtered).

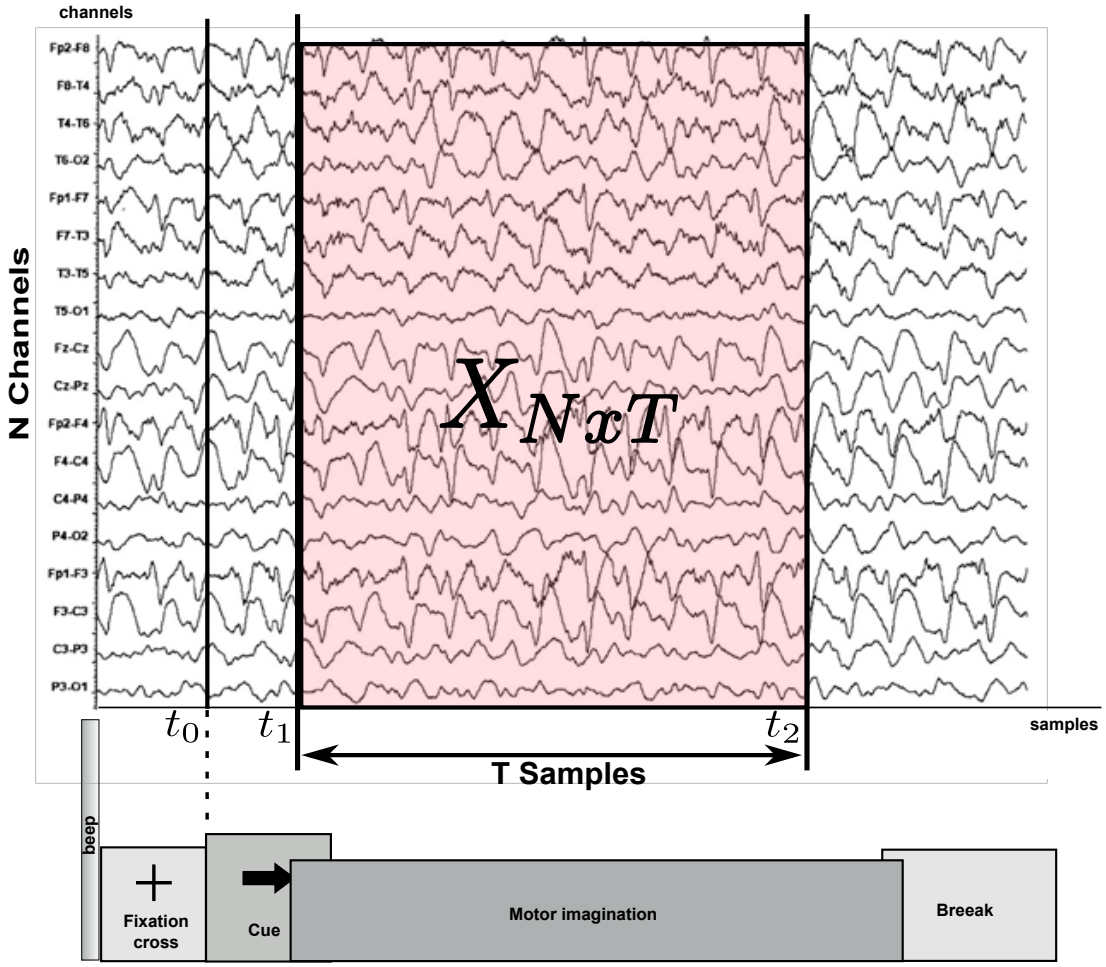

Figure 3.8 : Illustration of an epoch in a motor imagery experiment.

A spatial filter projects a multi dimensional signal into a single dimension with linear combination of the channels. In other words, let $\boldsymbol{w} \in \mathbb{R}^{Nx1}$ be a vector in N-dimensional space. A projection of an epoch onto this vector will be

$$\boldsymbol{y}_k = \boldsymbol{w}^\top X_k \tag{3.1}$$

where $\boldsymbol{y}_k \in \mathbb{R}^{1xT}$ denotes the projection of epoch X_k and $^\top$ is the transpose operation. The projected signal power $P_k \in \mathbb{R}^+$ can be written as follows:

$$P_k^\top = \boldsymbol{y}_k \boldsymbol{y}_k^\top = \boldsymbol{w}^\top X_k X_k^\top \boldsymbol{w} \tag{3.2}$$

Let $R_k \in \mathbb{R}^{NxN}$ be the covariance matrix of the band pass-filtered signal X_k and $R^c \in \mathbb{R}^{NxN}$ be the average covariance matrix of class c:

$$R_k = \frac{X_k X_k^\top}{tr(X_k X_k^\top)} \qquad R^{(c)} = \frac{1}{n_c} \sum_{k \in c}^{n_c} R_k \tag{3.3}$$

where tr is the trace function and n_c is the number of epochs in c. Here, it is better to describe about basic properties of a covariance matrix in order to make the subsequent equations clear. The covariance matrix defined by the left side of the equation 3.3 is said to be a *symetric positive define* (spd) matrix. A real NxN matrix R is called spd matrix if,

$$\boldsymbol{w}^\top R \boldsymbol{w} > 0 \tag{3.4}$$

For any $\boldsymbol{w} \in \mathbb{R}^{Nx1}$. Positive definite matrices are of both theoretical and computational importance in a wide variety of applications. They are used, for example, in optimization algorithms and in the construction of various linear regression models [169]. The following properties are some important properties of a positive definite matrix:

- All eigenvalues of a spd matrix are positive.
- Every positive definite matrix is invertible and its inverse is also positive definite.
- If M and N are positive definite, then the sum $M+N$ is also a positive define matrix. However, $M-N$ may not be a positive definite matrix.

Regarding the the last property in the above listing, the average covariance matrix R^c defined at the right hand side of the equation 3.3 must be also a spd matrix. Let the average power of class c be P^c. Then, P^c is calculated as follows:

$$P^{(c)} = \frac{1}{n_c} \sum_{k \in c}^{n_c} \boldsymbol{w}^\top X_k X_k^\top \boldsymbol{w} = \frac{1}{n_c} \sum_{k \in c}^{n_c} \boldsymbol{w}^\top R_k \boldsymbol{w} = \boldsymbol{w}^\top R^{(c)} \boldsymbol{w} \tag{3.5}$$

Since P^c is a For the two classes $(c = 1,2)$ case, CSP searches for the maximum power ratio of the classes on the projected w axis. Thus, the average power of one class is maximized while that of the other class is minimized. In other words, the spatial filter should maximize the following Rayleigh quotient problem [102],

$$\boldsymbol{w}_{csp} = \arg\max_{\boldsymbol{w}} \frac{\boldsymbol{w}^\top R^{(1)} \boldsymbol{w}}{\boldsymbol{w}^\top R^{(2)} \boldsymbol{w}} \tag{3.6}$$

Above equation may be solved by using constrained optimization techniques. Assume $\boldsymbol{w}_{csp}$ is the spatial filter vector that maximizes equation (3.6). By multiplying the upper and lower parts of the fraction with any scalar, one can set the denominator of the equation given in 3.6 to any value without changing the ratio. Thus, maximization of the Rayleigh quotient can be translated into the following constrained optimization problem:

$$\textit{maximize} \quad \boldsymbol{w}^\top R^{(1)} \boldsymbol{w} \qquad , \qquad \textit{w.r.t} \;\; \boldsymbol{w}^\top R^{(2)} \boldsymbol{w} = 1 \tag{3.7}$$

By using the Lagrange multiplier method [170], defined constrained optimization problem is solved easily. In mathematical optimization, the method of Lagrange multipliers (named after Joseph Louis Lagrange [171]) is a strategy for finding the local maxima and minima of a function subject to equality constraints. In this method, a Lagrange multiplier λ and Lagrange function $\mathscr{L}$ (*Lagrangian*) are introduced,

$$\mathscr{L}(\lambda, \boldsymbol{w}) = \boldsymbol{w}^\top R^{(1)} \boldsymbol{w} - \lambda (\boldsymbol{w}^\top R^{(2)} \boldsymbol{w}) \tag{3.8}$$

If $\boldsymbol{w}_0^\top R^{(1)} \boldsymbol{w}_0$ is a maximum of $\boldsymbol{w}^\top R^{(1)} \boldsymbol{w}$ for the original constrained problem, then there exists λ_0 such that (w_0, λ_0) is a stationary point for the Lagrange function where the partial derivatives of $\mathscr{L}$ are zero.

$$\frac{\partial \mathscr{L}(\lambda, \boldsymbol{w})}{\partial \boldsymbol{w}} = 2\boldsymbol{w}^T R^{(1)} - \lambda (2\boldsymbol{w}^T R^{(2)}) = 0 \tag{3.9}$$

Since $R^{(c)}$ is a symmetric matrix, the above equation can be written as a standard eigenvalue problem:

$$({R^{(2)}}^{-1} R^{(1)}) \boldsymbol{w} = \lambda \boldsymbol{w} \tag{3.10}$$

Where $(^{-1})$ is the matrix inverse operation. Note that, since it is a covariance matrix, inverse of $R^{(2)}$ is valid. According to Equation (3.10), $\boldsymbol{w}$ vector, which maximizes the Rayleigh quotient, is the eigenvector that corresponds to the largest eigenvalue of

$(R^{(2)^{-1}}R^{(1)})$. Note that above eigenvalue problem is easily solved by Matlab's *eig* command [172].

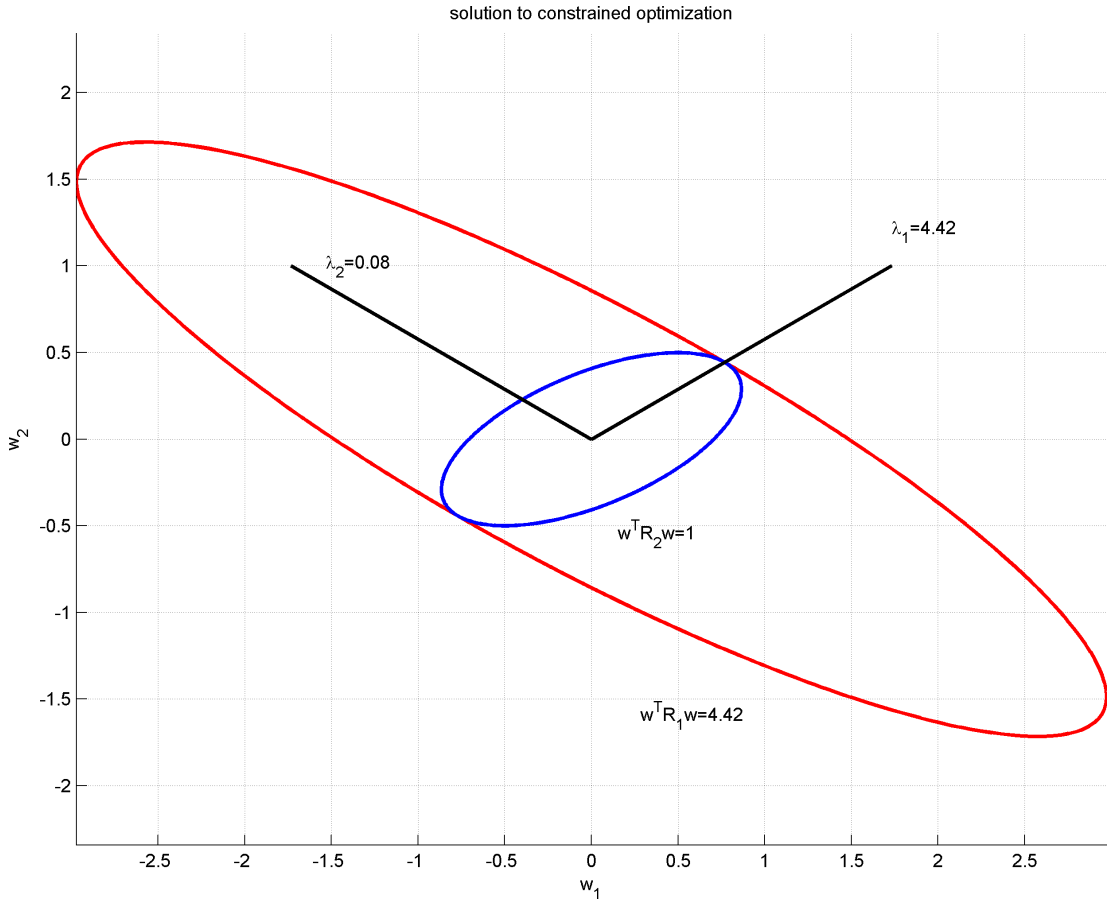

Figure 3.9 : An example figure for constrained optimization of Rayleigh ratio.

An example for Rayleigh ratio optimization in two dimension is given in Figure 3.9. In this figure, $2x2$ sized $R^{(1)}$ and $R^{(2)}$ covariance matrices are given as, $R^{(1)} = [2,3;3,6]$ and $R^{(2)} = [2,-2;-2,6]$. In the figure, w_1 and w_2 axes denotes the elements of $\boldsymbol{w}$ vector. Red ellipse is the set of points where $\boldsymbol{w}^\top R^{(2)}\boldsymbol{w} = 1$ and black one is the set of points where $\boldsymbol{w}^\top R^{(1)}\boldsymbol{w} = 4.42$ in which, 4.42 is the maximum value that can be assigned to $\boldsymbol{w}^\top R^{(1)}\boldsymbol{w}$ while having minimum one intersection points, so that the constraint is still satisfied. Blue lines in the figure show the directions of the eigenvectors corresponding to the calculated eigen values λ_1 and λ_2 according to the equation 3.10. Intersection points of the two ellipses satisfy the result of the Lagrange multiplier method. At these points, the $(\boldsymbol{w}^\top R^{(1)}\boldsymbol{w})/(\boldsymbol{w}^\top R^{(2)}\boldsymbol{w})$ is maximum, corresponding to the eigenvector of the maximum eigenvalue.

The described optimization method finds N eigenvalue - eigenvector pairs, for the covariance matrices of size NxN. CSP method then sorts all obtained eigenvalues

in descending order. Then, CSP spatial filter $W_{CSP} \in \mathbb{R}^{MxN}$ matrix is constructed by taking $M = 2m$ ($M \leq N$) eigenvectors corresponding to the m largest and m smallest eigenvalues:

$$\begin{aligned} &\lambda_1 > \lambda_2 > ... > \lambda_m > .....\lambda_{N-m+1} > ... > \lambda_{N-1} > \lambda_N \\ &W_{CSP} = [\boldsymbol{w}_{\lambda_1}, \boldsymbol{w}_{\lambda_2}, ...\boldsymbol{w}_{\lambda_m}, ...\boldsymbol{w}_{\lambda_{N-m+1}}, ...\boldsymbol{w}_{\lambda_{N-1}}, \boldsymbol{w}_{\lambda_N}]^T \end{aligned} \tag{3.11}$$

where w_{λ_i} is the eigenvector that corresponds to the eigenvalue λ_i. Any epoch X_k is spatially filtered by the constructed W_{CSP} matrix,

$$Y_k = W_{CSP} X_k \tag{3.12}$$

where $Z_k \in \mathbb{R}^{MxT}$ is the spatially filtered signal. Band power (variance) is used as a feature for the classifier. For an epoch k, the CSP feature vector is given by

$$\boldsymbol{fcsp}_k^i = log(\frac{var(Z_k^i)}{\sum_{l=1}^{2m} var(Z_k^l)}) \quad i = 1, 2, ...M \tag{3.13}$$

where $\boldsymbol{fcsp}_k^i$ is the i^{th} feature of feature vector $\boldsymbol{fcsp}_k \in \mathbb{R}^{Mx1}$ that belongs to epoch k and Z_k^i is the i^{th} row of Z_k. Here, the logarithm of the variance ratio is calculated in order to approximate the distribution of the features to a normal distribution [132]. Next, the features are used to train a linear classifier.

3.3.1 A toy data example

Here, an example about the effect of CSP spatial filtering on two dimensional data will be presented. Suppose two sets of samples in 2D, with different covariance matrices. The data are drawn from two Gaussian distributions whose covariances matrices are [2,3;3,6] and [2,-2;-2,6] for the two classes marked with red and blue dots, respectively. Figure 3.10 shows scatter plots for original and filtered data scatter plots as well as their variance features in x_1 and x_2 axes. In (a), The two classes of the original data scattered in either dimensions. the input data calculated by principle components are enclosed with ellipses. Note that the input data have the same variance in each dimension. If we get the variance features for each dimension preceding to the filtering operation, we obtain a feature set scattered at either dimensions for any class, which is shown in (b).

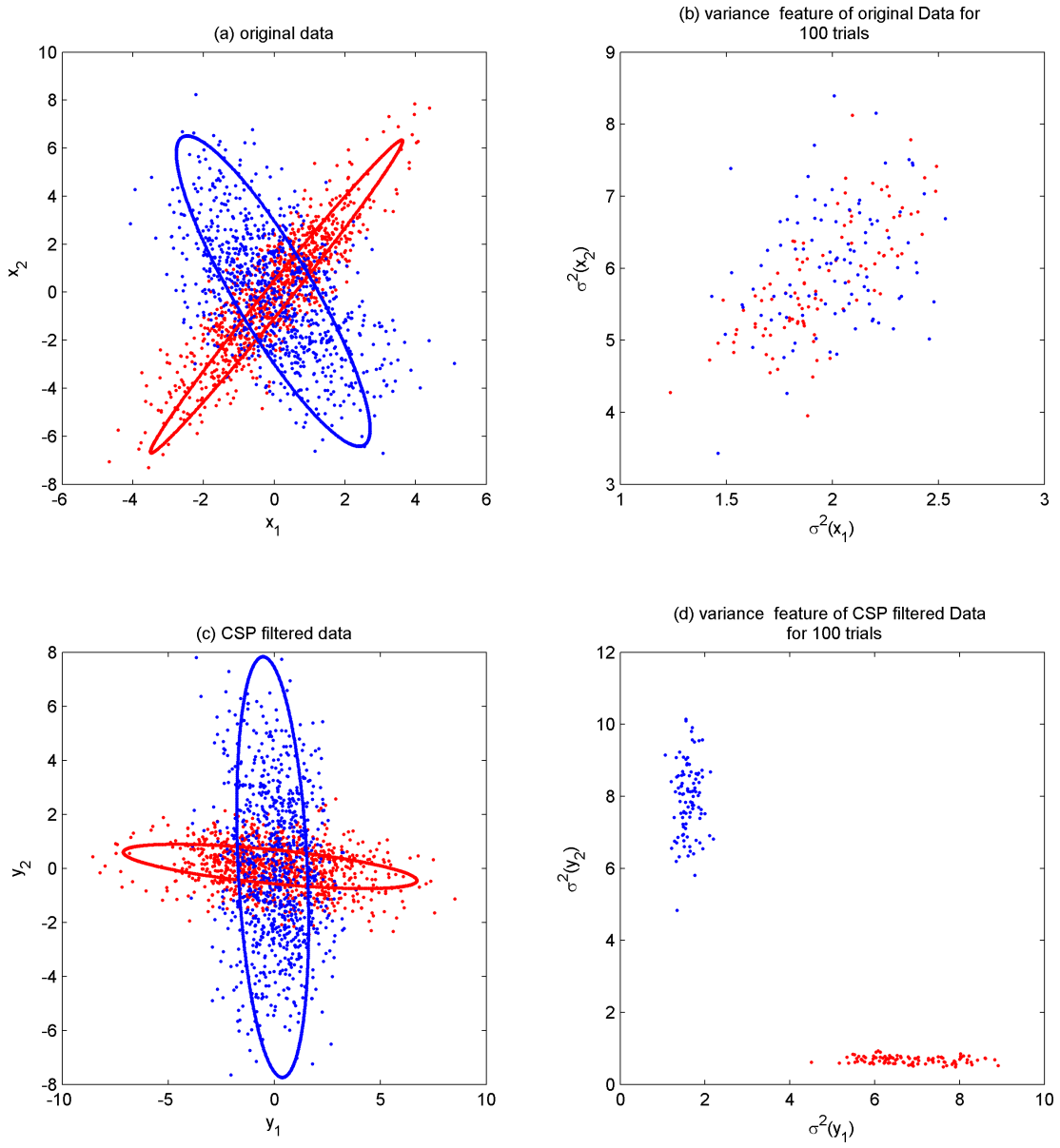

Figure 3.10 : A toy data example for CSP.

Therefore, it is impossible to separate the obtained data by variance feature without a spatial filtering operation. However, the goal of the CSP operation is to manipulate the incoming data so that different classes spread over only in a specific axis. In the example figure, (c) displays the scattering of the CSP filtered data. In this case, the ellipses, which shows the principal axes are orthogonal, which means the both classes are uncorrolated at the same time. Therefore, the data marked with red dots have maximum variance (power) on the y_1 axis while having minimum variance (power) on the y_2 axis. Also, the same is true for the blue marked data for the axes inverted. (d) shows the variance feature for the spatially filtered data. It is clearly seen that, red class have bigger values for the feature axis $\sigma^2(y_1)$ while blue class have bigger values for

the feature axis $\sigma^2(y_2)$, so that a simple linear classifier may easily classify the data by the variance feature.

3.3.2 A real data example

Here, an application of CSP to a real world example will be described. The data used in this example was obtained from BCI competition III - data set IV-A [129]. In this data set, there are two classes called *right arm* and *foot* for five subjects. The detailed description of the data set and procedure for obtaining the epoch data will be covered in the following chapter. However, in this example, the influence of CSP on the EEG data will be briefly demonstrated.

There are 68 EEG channels selected from the EEG data. First, the data was filtered and epochs were parsed. Then, the CSP method was applied. After CSP operation, obtained Rayleigh ratios λ for each eigenvector (spatial filter) is given in Figure 3.11. Here, λ values are sorted in descending order.

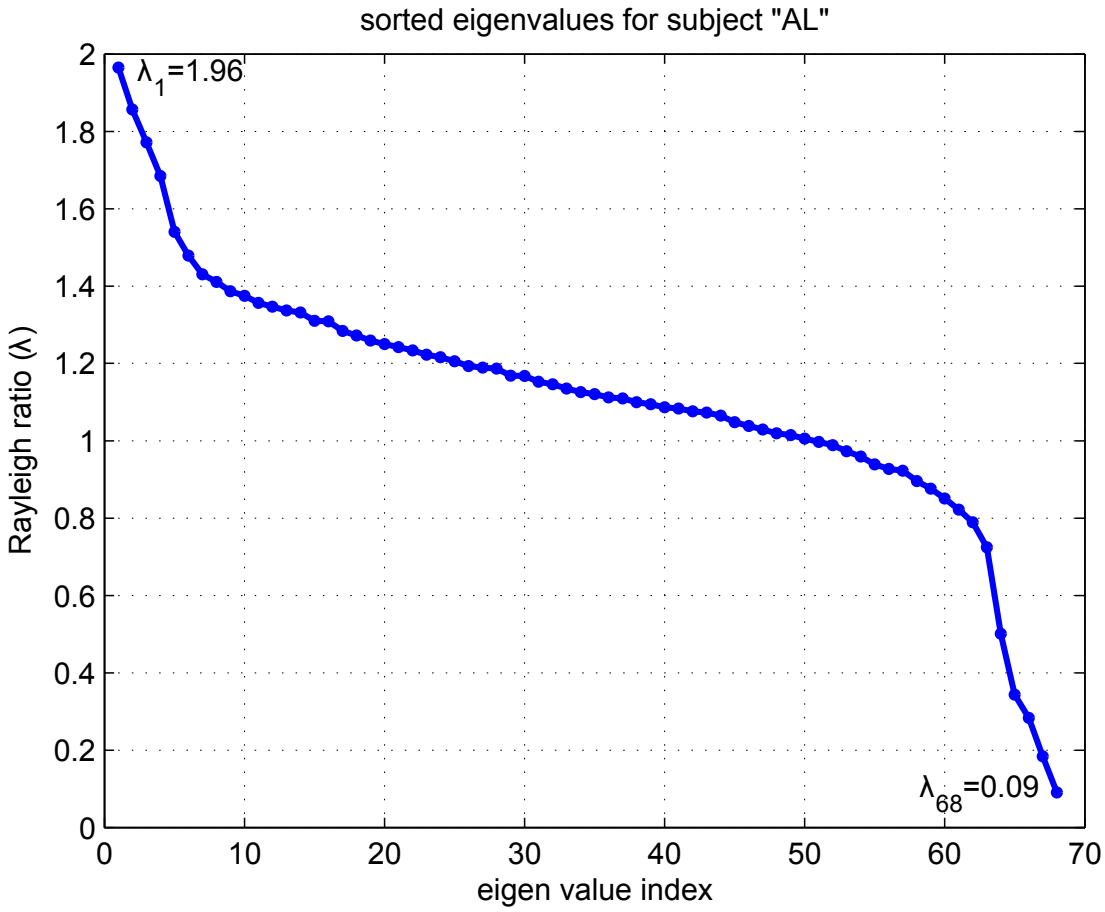

Figure 3.11 : Plot of obtained eigenvalues for EEG data set.

In order to get a 2 channel output signal for obtaining a scatter plot in 2D, two eigenvectors coreesponding to the biggest and the smallest eigenvalues were used. Variance feature for the filtered data is given in Figure 3.12. It is obvious that CSP is successful at un-correlating the input data.

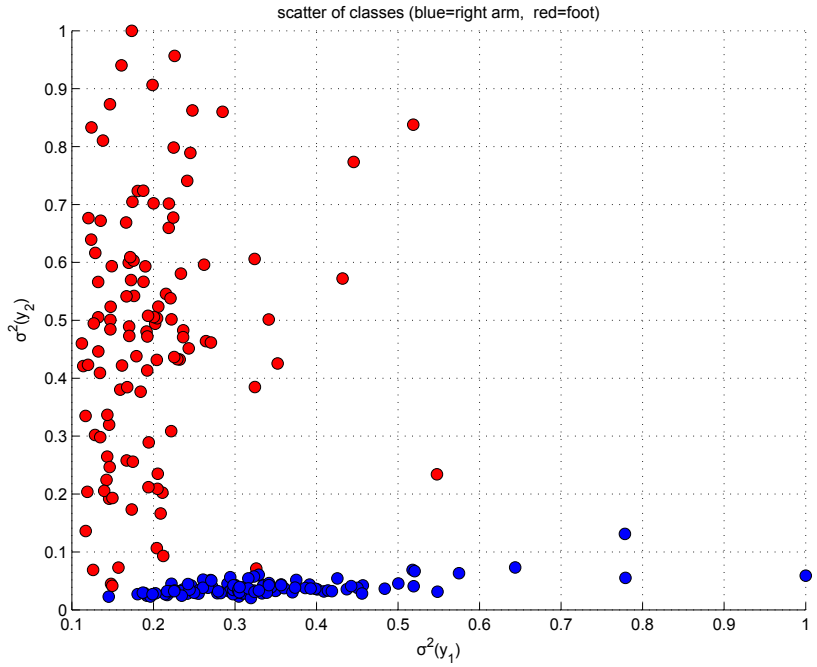

Figure 3.12 : Scattering of features for spatially filtered EEG data.

A spatial filter contains the coefficients to be multiplied with each of the input channel. The amplitude of the coefficient is a clue for the importance of the corresponding channel for a spatial filter. Higher amplitude means higher importance of that specific channel. For example, one expects a spatial filter correlated with *right arm* should focus on the left hemisphere of the head, so, the coefficients at there should have high amplitude. By visualizing the spatial filter coefficient amplitudes on a head figure with color coding, contour lines and interpolation, we may get a comprehensible image of the calculated spatial filter. Obtained spatial filters are displayed in Figure 3.13. Considering the spatial filter images and the Homunculus Figure at 2.6, we may conclude that the output of the CSP is plausible with the physiological facts.

3.3.3 Extending CSP for multiclass classification

The optimization function of CSP is defined for two classes. When there are more than two motor imagery classes (e.g., *left hand*, *right hand*, *foot*, *tongue*, etc..), the CSP

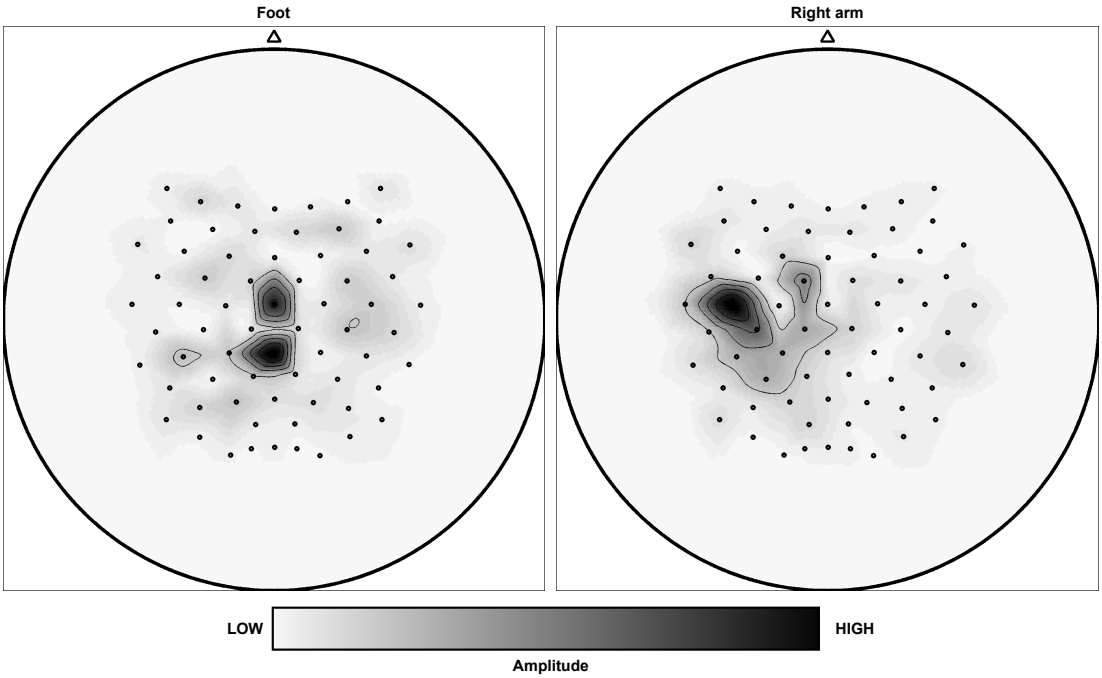

Figure 3.13 : Illustration of spatial filters obtained with the CSP method.

method requires some modifications. Dornhege et.al [173] suggested some alternatives as a multi class exension to CSP.

- **Using CSP Within the Classifier (IN):** In this method,a multi class problem is reduced to several binary classification problems. Two class combinations of all classes are applied to standard binary CSPs. Then, variances of the output projections of a CSP are employed as inputs to a linear classifier such as LDA. For labeling an input epoch, a voting method is used by considering the particular outputs of the classifiers. With this method, there is no modification to the original binary CSP.

- **One versus the rest CSP (OVR-CSP):** With OVR-CSP, optimization function of CSP (eq. 3.6) is modified for maximizingthe power of one class versus the total power of the rest of the classes. The OVR-CSP for class c is calculated as follows:

$$\boldsymbol{w_c} = \arg\max_{\boldsymbol{w}} \frac{\boldsymbol{w}^T R^{(c)} \boldsymbol{w}}{\sum_{j \neq c}^{C} \boldsymbol{w}^T R^{(j)} \boldsymbol{w}} \tag{3.14}$$

  where C is the total number of classes and $\boldsymbol{w_c}$ is the vector that maximizes the Rayleigh ratio between class c and the other classes. The CSP matrix is constructed by calculating the above equation for all classes and concatenating

preferred number of spatial filters for each class row by row. Note that OVR-CSP is a generalization for the CSP method, which is equal to (3.6) for the two classes ($C = 2$) case.

- **Simultaneous Diagonalization (SIM):** While uncorrelating the classes, CSP algorithm transforms the covariance matrices of both classes to diagonal matrices. This operation is called *simulatenous diagonalization* and it is possible to simultaneously diagonalize any two symetric positive definite covariance matrices. However, when there are more then two classes absolute simultaneous diagonalization is impossible, however there are many methods for *approximate simultaneous diagonalization* [174–183] which diagonalize all of the given matrices as much as possible. Mentioned diagonalization method is a mathematical method and is not related with CSP algorithm. Also, such methods aren't concerned with maximizing the Rayleigh ratio while diagonalizing the covariance matrices however, simultaneous diagonalization methods have been used in motor imagery classification [5, 116, 184]

In this study, we used OVR-CSP method for multi class data sets because of its practicability and computationally lightness. Also OVR-CSP method aims improving the Rayleigh ratio for the given class, which is very important for classification performance.

3.4 Regularized Common Spatial Patterns

Despite of its high popularity and success, CSP algorithm is prone to over fitting due to artifacts and outliers. CSP should be regularized in order to overcome its sensitivity to noise and over fitting [134]. Regularization of Rayleigh quotient given in Equation (3.6) is done by adding a penalty term to denominator. In this case, one should maximize spatial filters for each class seperately,

$$\boldsymbol{w}_1 = \underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{\boldsymbol{w}^T R^{(1)} \boldsymbol{w}}{(1-\alpha)\boldsymbol{w}^T \overline{R}^{(2)} \boldsymbol{w} + \alpha \boldsymbol{w}^T K_1 \boldsymbol{w}} \tag{3.15}$$

$$\boldsymbol{w}_2 = \underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{\boldsymbol{w}^T R^{(2)} \boldsymbol{w}}{(1-\alpha)\boldsymbol{w}^T R^{(1)} \boldsymbol{w} + \alpha \boldsymbol{w}^T K_2 \boldsymbol{w}} \tag{3.16}$$

where $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are the spatial filters maximizing the variances of class 1 and 2 respectively. α is user defined regularization parameter to set the effect of penalty term. K_1 and K_2 are NxN penalty matrices. There are many variants about computation of K matrices in the literature. In the following subsections, six regularized CSP variants including the proposed regularized CSP method will be reviewed.

3.4.1 Tikhonov regularized CSP (TRCSP)

In Tikhonov regularized CSP (TRCSP) [134], K matrices are set to identity matrices so that penalty term is equal to $\boldsymbol{w}^T I \boldsymbol{w} = \boldsymbol{w}^T \boldsymbol{w} = \|\boldsymbol{w}\|^2$. Thus, solutions with large weights are penalized. TRCSP is expected to generate filters with small norm hence reducing the effect of artifacts and outliers. Note that, TRCSP penalizes all channels equally, a channel with highly motor imagery related activity may be penalized. However, there should not be a penalty if it contains useful information.

3.4.2 Weighted Tikhonov regularized CSP (WTRCSP)

In TRCSP, each channel are penalized equally. However, it is known that some channels is more important than others. In Weighted Tikhonov Regularized CSP, K matrix is set as a diagonal matrix,

$$K = diag(\boldsymbol{u}) \tag{3.17}$$

where $\boldsymbol{u}$ is a coefficient vector and $diag(\boldsymbol{u})$ is the diagonal matrix of $\boldsymbol{u}$ which has the information of the penalty level of each channel [134]. Since manual selection of u is not easy, it is computed by using the data from other test subjects:

$$\boldsymbol{u} = \left(\frac{1}{2N_f|\Omega|} \sum_{i \in \Omega} \sum_{f=1}^{2N_f} \left| \frac{\boldsymbol{w}_f^i}{||\boldsymbol{w}_f^i||} \right| \right)^{-1} \tag{3.18}$$

where $\boldsymbol{w}_f^i$ is the f^{th} spatial filter of subject i and $2N_f$ is the number of spatial filters used for each subject. Shortly, in WTRCSP penalty of each channel is adjusted by looking at other subjects. If average absolute value of the normalized weight of a channel is large in the CSP filters of other subject, penalty term assigned to this channel is small. This method needs more than one subjects in order to create a penalty function. Also,

spatial filters of one subject can be contaminated by the outliers and noise in other subjects.

3.4.3 Invariant CSP

Invariant CSP (iCSP) method uses non task related EEG signal for building filters invariant to a given noise source [121]. Blankertz et. al. used disturbance covariance matrices from fluctuations in visual processing, parietal α-activity and used these covariance matrices as regularization matrices. Note that, there should be additional EEG measurements to compute the covariance matrix. For example, Samek et.al [102] recorded an extra session consisting of different eye movements, namely *eyes open*, *look left*, *look right*, *look up* and *look down* to generate an average covariance matrix K. Then adding $\boldsymbol{w}^T K \boldsymbol{w}$ as penalty term results in spatial filters which are invariant against changes generated by eye movements.

3.4.4 Spatially regulaized CSP

Spatially regulaized CSP (SRCSP) (see Lotte and Guan [134]) deals with spatial relations between EEG channels. Normally CSP ignores positions and spatial relations of EEG electrodes. SRCSP method utilizes the information that neighboring neurons have similar functions, so that neighboring electrodes should measure similar brain signals. SRCSP method penalizes solutions with non-smooth filters in which spatially close electrodes should be similar weights in CSP spatial filter.

SRSCP uses the following regularization matrix K:

$$G(i,j) = exp(-\frac{1}{2}\frac{||\boldsymbol{v}_i - \boldsymbol{v}_j||^2}{r^2}) \tag{3.19}$$

$$K = D_G - G \tag{3.20}$$

where $\boldsymbol{v}_i$ is the position vector of i^{th} electrode, D_G is a diagonal matrix such as $D_G = \sum_j G(i,j)$ and r is a hyper parameter representing the maximum distance between two electrodes. SRCSP uses spatial information about electrode channels but ignores about penalizing the channels with non-task-related information.

3.4.5 Stationary CSP

Stationary CSP (sCSP) [102] extends CSP to be invariant to non-stationaries in the data and regularizes CSP towards stationary subspaces. sCSP aims reducing the variations of the extracted features since it is assumed that variations like non-stationeries are the results of non task-related processes such as eye movements and electrode artefacts. sCSP penalizes the solutions with large variations in spatially filtered EEG data by analyzing training epochs. Generally, sCSP tries to minimize the following term in each class c:

$$D_c(\boldsymbol{w}) = \sum_k |\boldsymbol{w}^T R_c^{(k)} \boldsymbol{w} - \boldsymbol{w}^T \overline{R}_c \boldsymbol{w}| \tag{3.21}$$

where $R_c^{(k)}$ is the k^th trial of class c and $\overline{R}_c$ is the average covariance matrix of class c. Minimizing D_c should minimize the variation of projections of each particular class. Note that, despite the fact that, covariance matrices are symmetric positive definite matrices, their difference may not give symmetric positive matrices. Hence, $\boldsymbol{w}^T(R_c^{(k)}\overline{R}_c)\boldsymbol{w}$ may be a negative scalar. Therefore, Samek applied absolute value operation here. However, introducing this term as a penalty term into the regularized CSP equation is impossible because of the absolute value operation. In order to remove the absolute value operator, he then suggested a $\mathscr{F}$ operation which transforms any symmetric matrix to a positive definite matrix by decomposing into the eigenvalue eigenvector form, changing sign of the eigenvalues and then composing back to the matrix form. Consequently, proposed penalty term is defined as,

$$P(\boldsymbol{w}) = \sum_{k \in C_1} \boldsymbol{w}^\top \mathscr{F}(R_1^{(k)} - \overline{R}_1)\boldsymbol{w} + \sum_{k \in C_2} \boldsymbol{w}^\top \mathscr{F}(R_1^{(k)} - \overline{R}_1)\boldsymbol{w} \tag{3.22}$$

where C_1 and C_2 denotes the class labels and $P(\boldsymbol{w})$ is the penalty term to be added to the regularized CSP equation.

3.4.6 Task related & spatially regularized CSP (TR&SR-CSP)

In this study, Task Related & Spatially Regularized CSP (TR&SR-CSP) is proposed. As mentioned before, each muscle group has a special area on the motor cortex, which supports the idea that each motor imagery task activates a special area on the

motor cortex. We aim a regularizing method which emphasizes the electrodes that are spatially close to the center of imaging task being executed. TR&SR-CSP method regularizes the CSP spatial filter by directing it towards to center electrodes, which is assumed to record task-related signals. This is realized by penalizing the filter weights which are assumed as non-task related components according to their spatial location. TR&SR-CSP designs regularizing matrices as below:

$$K_C(i,j) = \begin{cases} 1 - exp(-\frac{||\boldsymbol{v}_i - \boldsymbol{v}_c||^2}{r^2}), & i = j \\ 0, & i \neq j \end{cases} \tag{3.23}$$

where K_C is the regularizing matrice of class C, $\boldsymbol{v}_i$ is the spatial location of i^{th} electrode, $\boldsymbol{v}_c$ is the electrode which is assumed to be the center electrode of task related signal and r is a hyper parameter representing the maximum distance between two electrodes. Note that in TR&SR-CSP, regularizing K matrix is different for each class. Also spatial smoothness of filter is ensured by the exponential function. Figure 3.14 plots penalty function with different r values. Note that, penalty value is minimum when the distance between the centre electrode and the given electrode position is minimum which drives calculated spatial filters towards to a point where signal obtained from centre electrode is emphasized. Also, in order to visualize the effect of the penalty values in an EEG experiment, Figure 2 illustrates penalty matrices on scalp figures with topographic maps. Diagonal values of K matrices for the two tasks (right hand and foot) had been plotted.

Despite similarities to SRCSP and WTRCSP, TR&SR-CSP method differs from SRCSP by incorporation of task relevant information when computing K. WTRCSP ignores spatial relations between electrodes but obtains a general spatial filter by analyzing data from other subjects, while TR&SR-CSP builds its regularizing matrices by using the information in the literature about mapping motor cortex and taking account of spatial relations between electrodes.

3.4.6.1 Multiclass TR&SR-CSP

In this subsection, TR&SR-CSP equations adapted to multi class case. For this purpose, we utilized OVR-CSP equations (3.14) and regularized CSP equations (3.15, 3.16). Each class (task) in TR&SR-CSP has a central electrode which results in different penalty matrices K_c, where c is the corresponding class label. Combining

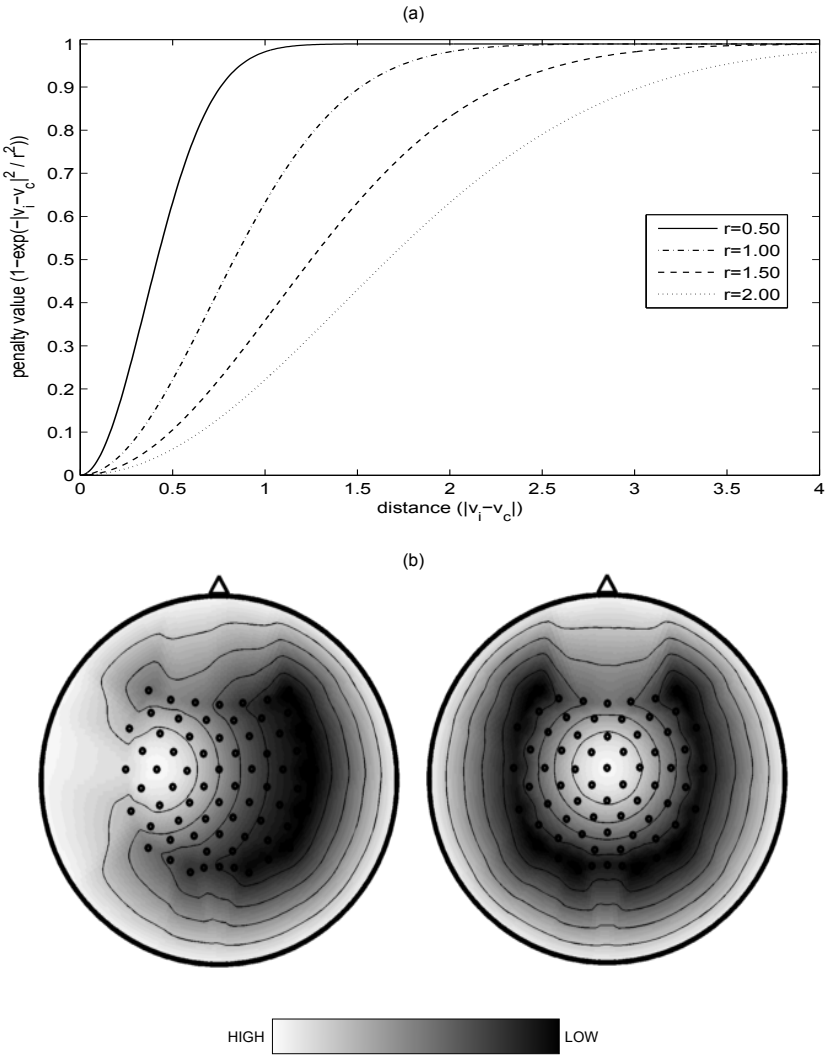

Figure 3.14 : (a) Plot of presented penalty function for various r values. (b) Illustration of penalty values for right hand and foot (r=0.3).

OVR-CSP equation and regularized CSP equations, a general equation for multi class regularized CSP is obtained:

$$\boldsymbol{w}_c = \underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{\boldsymbol{w}^\top \overline{R}_c \boldsymbol{w}}{(1-\alpha) \sum_{j \neq c}^{C} \boldsymbol{w}^\top \overline{R}_j \boldsymbol{w} + \alpha \boldsymbol{w}^\top K_c \boldsymbol{w}} \tag{3.24}$$

where C is the total number of classes. Firstly, a central electrode position v_c is defined and a penalty matrix K_c is calculated for each class c. Then, spatial filter specific to class c is calculated. Spatial filter matrix containing all of the spatial filters is constructed by selecting desired number of spatial filters for each class. Note that above equation is equal to the regularized CSP equations (3.15, 3.16) for two classes case.

TR&SR-CSP algorithm [136] was presented in 2nd International Work-Conference on Bioinformatics and Biomedical Engineering (IWBBIO) conference at Granada, Spain in 2014. Also extended version of the method was proposed for publishing in the Turkish journal of Electrical Engineering and Computer Sciences

3.5 Spatial Filter Network (SFN)

In this section, the first main contribution of this thesis will be presented. This study was published in PLoS One Journal with the title *A Neural Network-Based Optimal Spatial Filter Design Method for Motor Imagery Classification* [137]. In this study, we propose a general framework called spatial filter network (SFN), which calculates optimal spatial filters using a neural network approach. With SFN, each epoch in a training set is given to the network for learning the optimal spatial filters, unlike CSP, which only uses one average covariance matrix for each class. Additionally, SFN is trained to directly increase the classification accuracy, whereas the purpose of the CSP method is to maximize the given optimization function, which indirectly increases the classification accuracy. In this section, SFN is introduced along with its network structure and suggested training methods. Then, the outcome of a toy data applied to the SFN will be presented.

3.5.1 General structure of spatial filter network

SFN is depicted in Figure 3.15. SFN consists of a spatial filter layer (Layer-1) and a classification layer (Layer-2), which are connected to each other with non-linear mapping functions. Layer-1 is formed with a spatial filtering matrix W and feature extraction functions. The input-output relations of the spatial filter layer are given below:

$$y_m(t) = \frac{\sum_{n=1}^{N} W_{nm}\boldsymbol{x}(t)}{\sqrt{\sum_{n=1}^{N} W_{nm}^2}} = \frac{\boldsymbol{w}_m^T}{\|\boldsymbol{w}_m\|}\boldsymbol{x}(t) \tag{3.25}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^{Nx1}$ is the input data at time t of the EEG epoch with N channels and T samples, $0 < t \leq T$. $\boldsymbol{w}_m \in \mathbb{R}^{Nx1}$ is the m^{th} column of the spatial filter matrix $W \in \mathbb{R}^{NxM}$, in which each column is a spatial filter and $y_m(t)$ is the output data at time t for the

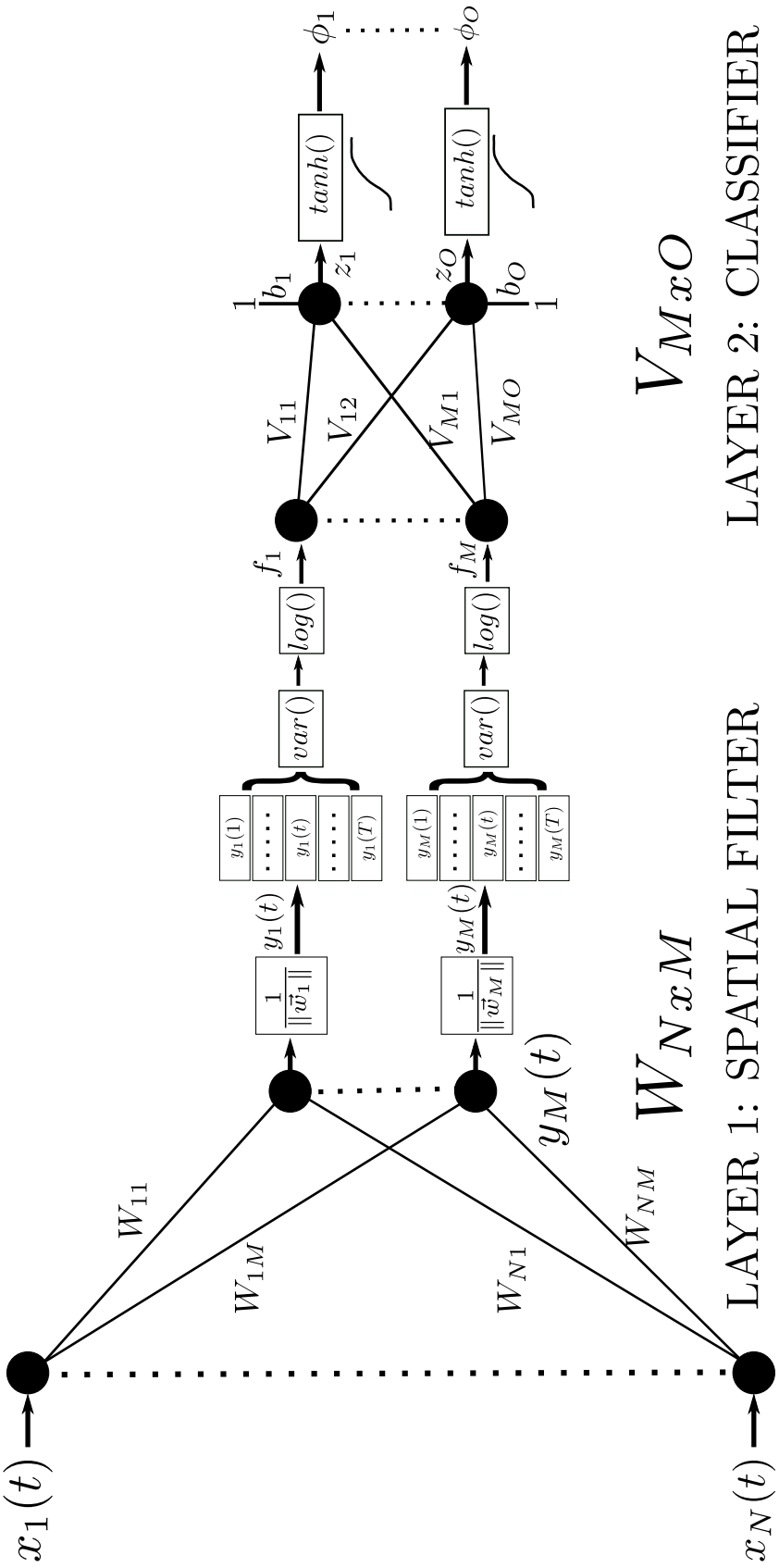

Figure 3.15 : Structure of the proposed spatial filter network (SFN).

m^{th} spatial filtering output. $\boldsymbol{w}_m$ is divided by its norm to ensure that the input signal is spatially filtered with unit norm filters. Thanks to the $1/\|\boldsymbol{w}_m\|$ block, SFN searches for optimal spatial filters over the surface a hypersphere in N dimensions. As shown in Fig. 3.16, the training algorithm, which moves $\boldsymbol{w}_m$ by d_w, actually moves the spatial filter over the hypersphere.

Spatial filtering is an important step in motor imagery classification. Redundant data that belong to irrelevant channels are weakened with spatial filtering. Spatial filtering is applied to the input data, and outputs $\boldsymbol{y}_m(t)$ are stored until all samples in one epoch are counted. After the spatial filtering phase, Layer-1 calculates the feature vector to be an input for Layer-2.

$$\boldsymbol{f_m} = log(var(\boldsymbol{y}_m)) = log(\frac{1}{T}\sum_{t=1}^{T}(y_m(t) - \mu_{\boldsymbol{y}_m})^2) \qquad m = 1,2,...M \tag{3.26}$$

where $var()$ is the variance operation and $log()$ is the natural logarithm function. Because input data X are zero mean, $\mu_{\boldsymbol{y}_m}$ will be zero. $\boldsymbol{f} \in \mathbb{R}^{Mx1}$ is the feature vector for a given epoch. Note that the logarithm is used as in CSP to approximate the distribution of the features to a normal distribution [132].

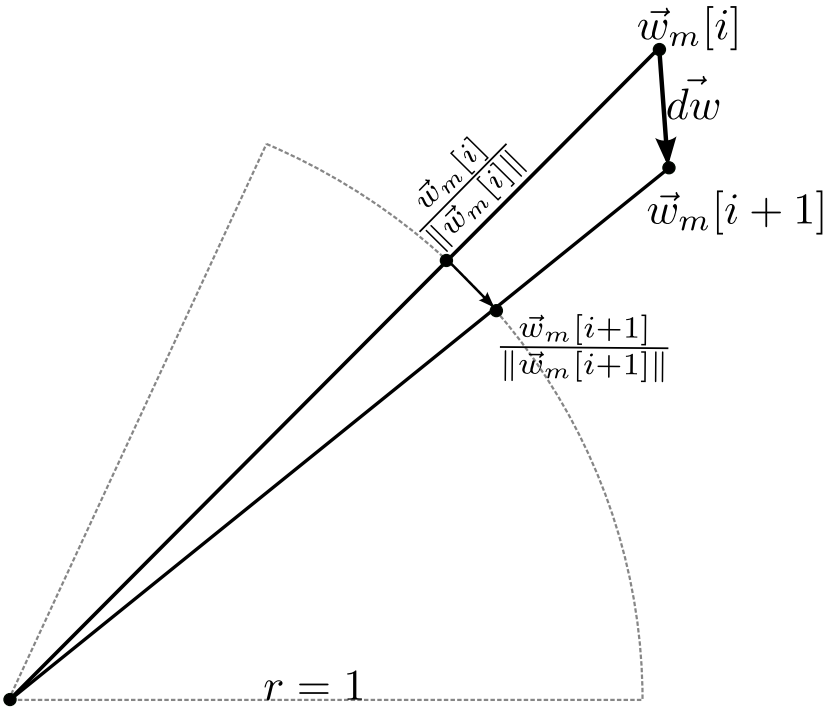

Figure 3.16 : A representative figure of searching the optimal spatial filter over the hypersphere of unit radius.

Layer-2 of SFN is the classification layer. An M-dimensional input feature vector (f) is mapped to an O-dimensional output vector z. The input-output relation of Layer-2 is given below:

$$\boldsymbol{z} = V^T \boldsymbol{f} + \boldsymbol{b} \tag{3.27}$$

$$\boldsymbol{\phi} = tanh(\boldsymbol{z}) \tag{3.28}$$

where $V \in \mathbb{R}^{MxO}$ is the weight matrix, $\boldsymbol{b} \in \mathbb{R}^{Ox1}$ is the bias vector, $tanh()$ is the tangent hyperbolic function used for clamping $\boldsymbol{z}$ to the range $(-1,+1)$, and $\boldsymbol{\phi}$ is the network output. Labeling of an input epoch X^k differs for binary and multi-category cases. For the two classes (binary) case, a single output neuron ($O = 1$) with a threshold value is used. Because *tanh* is used as an activation function, the threshold value will be 0. For a multi-classes case, the number of output neurons should be equal to the number of classes ($O = C$). In this case, the class label of an input epoch X^k is assigned by selecting the output neuron with the highest $\boldsymbol{\phi}$ value among all output neurons.

$$Class(X^k) = \arg\max_c(\boldsymbol{\phi_c^k}) \qquad c = 1,2,...C \tag{3.29}$$

Because the class label is generated by selecting the maximum output value, the resulting classifier is called a *linear machine*, in which no ambiguous region exists. A linear machine divides the feature space into C decision regions, with $\boldsymbol{\phi_c}$ being the largest discriminant if X^k in region R_c [155].

3.5.2 Training of SFN

We used two methods to train the SFN: Backpropagation [185] and Levenberg-Marquardt [186]. Both methods use partial derivatives to optimize the spatial filter coefficients W_{ij}. For the Backpropagation method, the coefficients are updated after each epoch is presented to SFN, whereas for the Levenberg-Marquardt method, updating is performed after all epochs in the training set are presented to the network. For both methods, the error function is calculated according to the network output and the class label of the epoch presented to the network. Additionally, the

initial weights for W, V and $\boldsymbol{b}$ are set randomly with a normal distribution $\sigma = 0.1$ and $\mu = 0$.

3.5.2.1 Error function

SFN requires an error function E to optimize the spatial filter W_{nm} and classifier coefficients V_{mo}. E should be minimum when SFN successfully discriminates different classes in the training set. We used the Euclidean distance between the target class vector $\boldsymbol{T_c}$ and SFN output vector $\boldsymbol{z}$ as the error measure. The error value of the k^{th} epoch is given by:

$$E^k = \frac{1}{2}\sum_{o=1}^{O}(e_o^k)^2 = \frac{1}{2}\sum_{o=1}^{O}(\boldsymbol{\phi}_o^k - \boldsymbol{D}_o^k)^2 \quad , 1 \leq k \leq K \tag{3.30}$$

where k is the current epoch number presented to the SFN, $\boldsymbol{\phi}_o^k$ is the o^{th} output of the SFN, $\boldsymbol{D}_o^k$ is the o^{th} element of the target vector when the k^{th} epoch is given to the network, and K is the total number of epochs in the training set. SFN is trained iteratively such that the total error of the network for the training class should be minimized.

3.5.2.2 Backpropagation method

The Backpropagation (BP) method adapts all weights of a neural network to minimize the error on a set of vectors belonging to a pattern recognition problem [156]. The BP learning rule is based on gradient descent. The weights are initialized with random values and changed in a direction to reduce the error [155]. In this study, the BP method is used to optimize both the spatial filter layer and the classifier layer. In a classical pattern recognition problem, a feature vector is given to the feed-forward neural network and the weights of the network are updated according to the feature's class and the network's output. However, the proposed training method accepts all of the samples in an epoch and updates the weights when an epoch is completely given to the network. For each layer, the learning rule of SFN is given by

$$W_{nm} = W_{nm} - \mu \frac{\partial E}{\partial W_{nm}}$$

$$V_{mo} = V_{mo} - \mu \frac{\partial E}{\partial V_{mo}} \tag{3.31}$$

$$\boldsymbol{b}_o = \boldsymbol{b}_o - \mu \frac{\partial E}{\partial \boldsymbol{b}_o}$$

where V_{mo} and W_{nm} are the weights of the classifier and spatial filter layers, respectively, b_o is the bias value for the second layer of the network, and μ is the learning rate parameter. The partial derivatives for each layer are calculated using the chain rule:

$$\frac{\partial E}{\partial V_{m,o}} = \frac{\partial E}{\partial \boldsymbol{\phi}_o} \frac{\partial \boldsymbol{\phi}_o}{\partial \boldsymbol{z}_o} \frac{\partial \boldsymbol{z}_o}{\partial V_{m,o}} \tag{3.32}$$

$$\frac{\partial E}{\partial \boldsymbol{b}_o} = \frac{\partial E}{\partial \boldsymbol{\phi}_o} \frac{\partial \boldsymbol{\phi}_o}{\partial \boldsymbol{z}_o} \frac{\partial \boldsymbol{z}_o}{\partial \boldsymbol{b}_o} \tag{3.33}$$

$$\frac{\partial E}{\partial W_{n,m}} = \sum_{o=1}^{O} \frac{\partial E}{\partial \boldsymbol{\phi}_o} \frac{\partial \boldsymbol{\phi}_o}{\partial \boldsymbol{z}_o} \frac{\partial \boldsymbol{z}_o}{\partial \boldsymbol{f}_m} \sum_{t=1}^{T} \frac{\partial \boldsymbol{f}_m}{\partial \boldsymbol{y}_m(t)} \frac{\partial \boldsymbol{y}_m(t)}{\partial W_{n,m}} \tag{3.34}$$

Note that the error of any output propagates to all weights in the spatial filter layer. Because the classifier layer output and error value are calculated after all of the samples in one epoch are counted, backpropagation should be calculated after the epoch is fully presented to the SFN. The required backward equations (derivatives) are given in equations (3.35) – (3.39):

$$\frac{\partial E}{\partial \boldsymbol{\phi}_o} = \boldsymbol{\phi}_o - \boldsymbol{D} \tag{3.35}$$

$$\frac{\partial \boldsymbol{\phi}_o}{\partial \boldsymbol{z}_o} = 1 - (\boldsymbol{\phi}_o)^2 \tag{3.36}$$

$$\frac{\partial \boldsymbol{z}_o}{\partial V_{m,o}} = \boldsymbol{f}_m \quad \frac{\partial \boldsymbol{z}_o}{\partial \boldsymbol{f}_m} = V_{m,o} \quad \frac{\partial \boldsymbol{z}_o}{\partial \boldsymbol{b}_o} = 1 \tag{3.37}$$

$$\frac{\partial \boldsymbol{f}_m}{\partial \boldsymbol{y}_m(t)} = \frac{1}{\frac{1}{T}\sum_{t=1}^{T} (\boldsymbol{y}_m(t))^2} \frac{2}{T} \boldsymbol{y}_m(t) \tag{3.38}$$

$$\frac{\partial \boldsymbol{y}_m(t)}{\partial W_{nm}} = \frac{\|\boldsymbol{w}_m\|\boldsymbol{x}_n(t) - W_{nm}\boldsymbol{y}_m(t)}{\|\boldsymbol{w}_m\|^2} \tag{3.39}$$

With the BP method, the weights (W, V and $\boldsymbol{b}$) are updated after each epoch. Training SFN with backpropagation is listed in Algorithm in Table 3.1. Here, itr refers to the iteration number, $MAXITR$ is the maximum number of iterations, $EMIN$ is the minimum error value to continue iterations, and W^0, V^0 and $\boldsymbol{b}^0$ are the initial values for W, V and $\boldsymbol{b}$, respectively.

Table 3.1 : Pseudo-code for BP algorithm

$itr \leftarrow 0$,$W \leftarrow W^0$, $V \leftarrow V^0$, $\boldsymbol{b} \leftarrow \boldsymbol{b}^0$
while $itr++ < MAXITR$ **and** $E > EMIN$ **do**
  Select an epoch from training set
  Calculate network output (eqs. 3.25– 3.28)
  Calculate error function E (eq. 3.30)
  Calculate $\partial E/\partial V_{mo}$,$\partial E/\partial \boldsymbol{b}_o$ and $\partial E/\partial W_{nm}$ (eqs. 3.32 – 3.34)
  Update network weights (eq. 3.31)
end while

3.5.2.3 Levenberg–Marquardt method

The Levenberg–Marquardt (LM) algorithm [186, 187] iteratively generates a solution for minimizing a non-linear problem. It is fast and has stable convergence [188]. Unlike the steepest descent method that the back-propagation algorithm uses, the LM method is an approximation to Newtons Method [189]. Let $E(\boldsymbol{q})$ be the total error that is iteratively minimized by the proposed method:

$$E(\boldsymbol{q}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{o=1}^{O}(e_o^k)^2 \tag{3.40}$$

where e_o^k is the error at the output o for the epoch number k and $\boldsymbol{q} \in \mathbb{R}^{NM+O(M+1)}$ is the vector of all weights forming the SFN:

$$e_o^k = D_o^k - \phi_o^k \tag{3.41}$$

$$\boldsymbol{q} = [W_{11}, W_{12}...W_{NM}, V_{11}, V_{12}, ..., V_{MO}, b_1, b_2, ...b_O] \tag{3.42}$$

The Levenberg–Marquardt method aims to minimize $E(\boldsymbol{q})$ according to the following update rule:

$$\boldsymbol{q}_{i+1} = \boldsymbol{q}_i - (J_i^T J_i + \mu I)^{-1} J_i \boldsymbol{e}_i \tag{3.43}$$

where i is the iteration number, $\boldsymbol{e} \in \mathbb{R}^{KO}$ is a vector that holds the errors of all outputs, $J \in \mathbb{R}^{(NM+O(M+1))x(KO)}$ is the Jacobian matrix, and μ is called the combination coefficient. When μ is very small, the algorithm works as the Gauss-Newton method, but when μ is large, the algorithm turns to the steepest descend [189] method. $\boldsymbol{e}$ and J are introduced below:

$$e = [e_1^1, e_2^1, ..., e_O^1 ..., e_1^K, e_2^K, ..., e_O^K]^T \tag{3.44}$$

$$J = \begin{pmatrix} \frac{\partial e_1^1}{\partial \boldsymbol{q}_1} & \frac{\partial e_1^1}{\partial \boldsymbol{q}_2} & \cdots & \frac{\partial e_1^1}{\partial \boldsymbol{q}_{NM+O(M+1)}} \\ \frac{\partial e_2^1}{\partial \boldsymbol{q}_1} & \frac{\partial e_2^1}{\partial \boldsymbol{q}_2} & \cdots & \frac{\partial e_2^1}{\partial \boldsymbol{q}_{NM+O(M+1)}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_O^1}{\partial \boldsymbol{q}_1} & \frac{\partial e_O^1}{\partial \boldsymbol{q}_2} & \cdots & \frac{\partial e_O^1}{\partial \boldsymbol{q}_{NM+O(M+1)}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_1^K}{\partial \boldsymbol{q}_1} & \frac{\partial e_1^K}{\partial \boldsymbol{q}_2} & \cdots & \frac{\partial e_1^K}{\partial \boldsymbol{q}_{NM+O(M+1)}} \\ \frac{\partial e_2^K}{\partial \boldsymbol{q}_1} & \frac{\partial e_2^K}{\partial \boldsymbol{q}_2} & \cdots & \frac{\partial e_2^K}{\partial \boldsymbol{q}_{NM+O(M+1)}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_O^K}{\partial \boldsymbol{q}_1} & \frac{\partial e_O^K}{\partial \boldsymbol{q}_2} & \cdots & \frac{\partial e_O^K}{\partial \boldsymbol{q}_{NM+O(M+1)}} \end{pmatrix} \tag{3.45}$$

When calculating the Jacobian matrix, the chain rule is applied to $\partial e / \partial \boldsymbol{q}$, and the following equations are obtained:

$$\frac{\partial e_o}{\partial V_{mo}} = \frac{\partial e_o}{\partial \boldsymbol{\phi}_o} \frac{\partial \boldsymbol{\phi}_o}{\partial \boldsymbol{z}_o} \frac{\partial \boldsymbol{z}_o}{\partial V_{mo}} \tag{3.46}$$

$$\frac{\partial e_o}{\partial \boldsymbol{b}_o} = \frac{\partial e_o}{\partial \boldsymbol{\phi}_o} \frac{\partial \boldsymbol{\phi}_o}{\partial \boldsymbol{z}_o} \frac{\partial \boldsymbol{z}_o}{\partial \boldsymbol{b}_o} \tag{3.47}$$

$$\frac{\partial e_o}{\partial W_{nm}} = \frac{\partial e_o}{\partial \boldsymbol{\phi}_o} \frac{\partial \boldsymbol{\phi}_o}{\partial \boldsymbol{z}_o} \frac{\partial \boldsymbol{z}_o}{\partial \boldsymbol{f}_m} \sum_{t=1}^{T} \frac{\partial \boldsymbol{f}_m}{\partial \boldsymbol{y}_m(t)} \frac{\partial \boldsymbol{y}_m(t)}{\partial W_{nm}} \tag{3.48}$$

Note that from (3.41), $\partial e_o / \partial \phi_o$ equals -1. Other terms may be calculated using equations (3.35) – (3.39). At the end of each epoch, the network outputs and error values are calculated, and then the Jacobian matrix is constructed. The LM method updates the network weights after all of the epochs in the training set are presented to

the SFN. μ is increased or decreased with each iteration such that the convergence rate is adjusted. Additionally, if the total error of the network increases with new weights after updating, the LM algorithm sets the weights to their previous values and slows the convergence rate. Training of SFN with the LM method is demonstrated using the algorithm in Table 3.2. Here, μ_0 is the initial value for the combination coefficient μ and β is the multiplier (divider) of μ for increasing (decreasing) it. For other terms, please refer to the description of the BP algorithm in Table 3.1. Note that, depending on the convergence of the network, the algorithm slightly changes to the steepest-descend or Gauss-Newton method by adjusting the value of μ.

Table 3.2 : Pseudo-code for LM algorithm.

$itr \leftarrow 0$, $\mu \leftarrow \mu_0$, $W \leftarrow W^0$, $V \leftarrow V^0$, $\boldsymbol{b} \leftarrow \boldsymbol{b}^0$
while $itr++ < MAXITR$ **and** $E > EMIN$ **do**
  for all epoch in Training Set **do**
    Calculate network output (eqs. 3.25–3.28)
    Calculate error for all outputs e_o^k (eq. 3.41)
    Calculate Jacobian matrix rows for the current epoch (eq. 3.45)
  end for
  Calculate total error (eq. 3.40)
  if $E_{itr} < E_{itr-1}$ **then**
    $\mu \leftarrow \mu / \beta$
  else
    $\mu \leftarrow \mu * \beta$
    Revert to previous weights $q_{itr} \leftarrow q_{itr-1}$
  end if
  Calculate new weights vector q_{itr} (eq. 3.43)
  Update network weights W,V and $\boldsymbol{b}$ (eq. 3.42)
end while

3.5.3 Running SFN with toy data

In this section, we test the SFN using generated toy data for 2 and 4 classes. Each class in the generated data has a different covariance matrix. For visualization purposes, the dimensions of the toy data and the size of the output neuron of the spatial filter layer were set to 2 ($N = 2$, $M = 2$). Note that the class label of an epoch in the toy data was randomly selected. Therefore, the number of epochs for each class will be approximately identical. Additionally, epoch data were generated with zero mean, fixed covariance matrices specific to each class using the Matlab command *mvnrnd* [172]. For both data sets, the epoch dimension (NxT) was set to $2x100$ and the total

number of epochs (K) was set to 100. SFN was trained with the BP and LM algorithms. The network parameters are as follows: $\mu = 10^{-3}$ for the BP method and $\mu_0 = 100$, $\beta = 2$ for the LM method. Both methods successfully converged to the desired error value ($EMIN = 10^{-3}$). As expected, the LM algorithm converged faster than the BP algorithm. The convergence of the two methods for the 2-classes case is shown in Fig. 3.17.

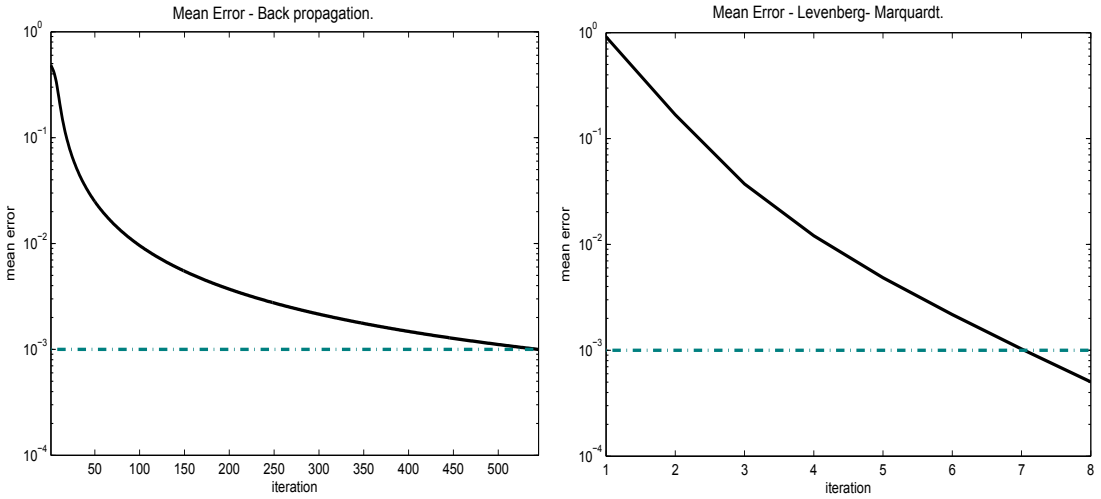

Figure 3.17 : Convergence of Backpropagation and Levenberg–Marquardt algorithms to the desired error ($EMIN$, dashed line) for 2 classes toy data example.

Fig. 3.18 presents the input data and the SFN output data for the 2-classes case. In this Figure, (a) illustrates the log-variance feature of the input data:

$$\begin{aligned} f_1(X^k) &= log(var([X^k_{11}, X^k_{12}, ...X^k_{1T}])) \\ f_2(X^k) &= log(var([X^k_{21}, X^k_{22}, ...X^k_{2T}])) \end{aligned} \tag{3.49}$$

where X^k is the k^{th} epoch. Each red (circle) and blue (plus) point represent the epoch with class 1 or 2, respectively. In (b), the input data calculated by principle components are enclosed with ellipses. Note that the input data have the same variance in each dimension. (c) Shows the effect of the SFN spatial filter layer, i.e., scattering of spatially filtered data (f^k_m). The spatial filter layer successfully separates the two classes. Here, the black dashed line shows the between-class border created by SFN classifier layer. In (d), the effect of SFN spatial filtering is shown with enclosing ellipses. Note that spatial filtering manipulates the data, in which the features that belong to any class have maximum variance in one dimension, and they have minimum variance in the other dimension.

The SFN output for the 4-classes toy data is shown in Fig. 3.19. In (a), the log-variance features of the 4 classes are shown. (b) Illustrates the enclosing ellipses for each class. The spatial filter output is given in (c). As for the 2-classes case, the spatial filter layer of SFN has two outputs ($M = 2$) such that we are able to visualize the output data. However, a larger output dimension may be used for data with more complex scattering. Because the SFN classifier has 4 outputs ($O = 4$), there are 4 class borders, which are drawn in (c). Note that, as a result of the selected target vectors in the one-versus-rest style, each borderline separates one class from the other classes. As shown in (d), the spatial filter layer manipulated the input data such that each class will have a different variance vector.

3.5.4 Convergence of network

Converge behavior of SFN was analyzed. Due to its superior convergence properties [190,191], the Levenberg-Marquardt (LM) method is used for training. SFN algorithm was run until converging to a desired error value. Fig. 3.20 displays the mean error at each iteration for multiple training experiments with subjects AA and K3B from the BCI competition datasets. SFN reaches the desired error at each experiment for both of the subjects. However, the average number of iterations until convergence is higher for subject K3B of the data set dataset IIIA, which is a 4 classes data set.

3.5.5 Training time

Because SFN learns the training set iteratively, the training time is longer than that for any direct method, such as CSP. Fig. 3.21 plots the average training times for subject *aa* in the BCIC-III-IVA data set versus the number of spatial filters (M). This figure also plots the training time across each subject, where each subject has a different number of training sets. As shown in this figure, the required time for training the network increases linearly with the size of the SFN weight matrices and the number of epochs in the training set. The algorithm was run on a single-core Intel® Xeon® CPU operating at 2 GHz.

Although the training of SFN takes a long time, once the network is trained, the classification time for an epoch is very small. Therefore, a longer training time is

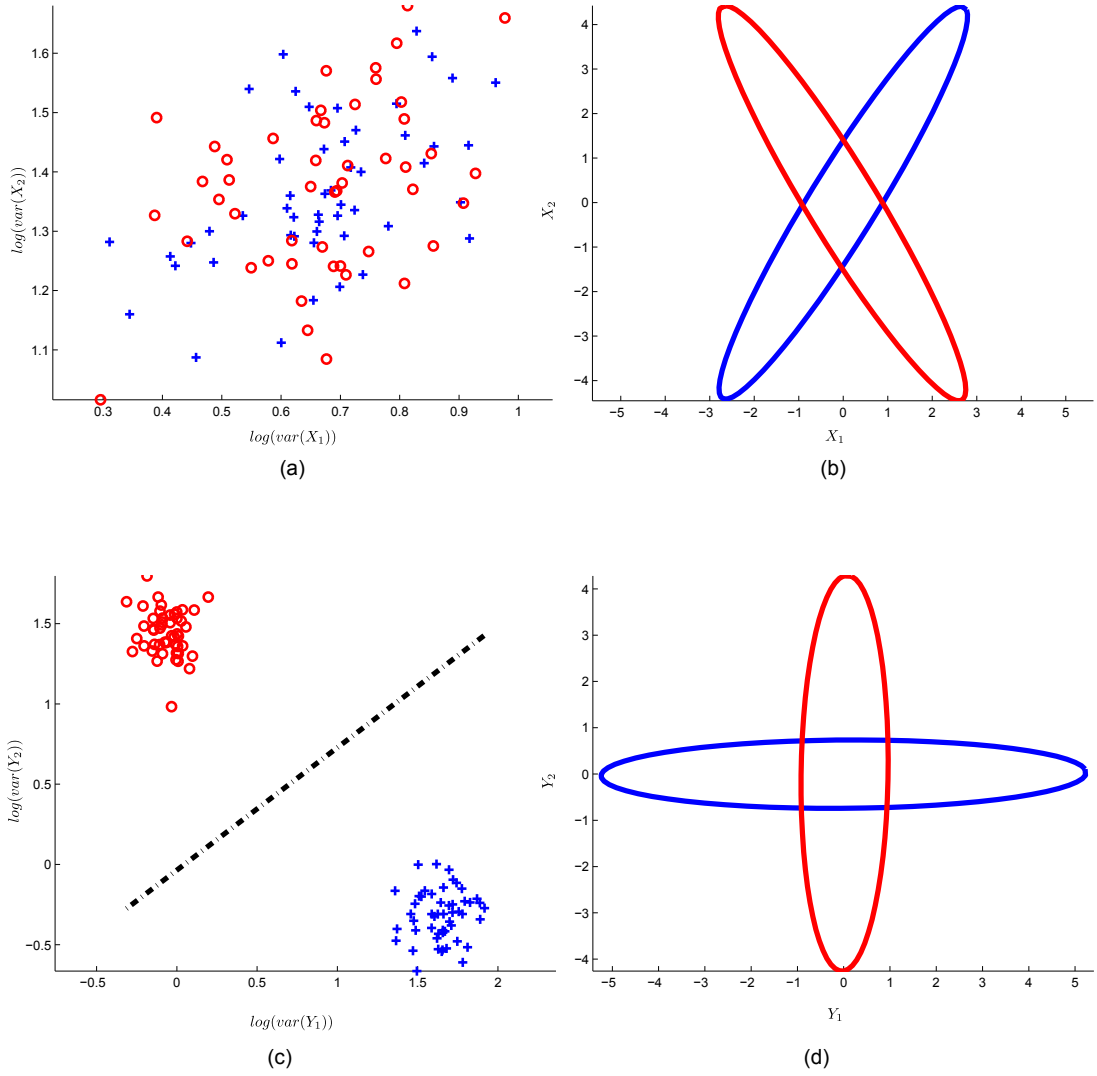

Figure 3.18 : Input data and SFN output data for toy data with 2 classes. (a) log-variance feature for 2-dimensional input data. Note that each point represents an epoch that belongs to class 1 (red circle) or class 2 (blue plus). (b) Enclosing ellipses represent the input data. (c) SFN spatial filter layer output (f) with generated class border (black dashed line) of the classifier layer. (d) Enclosing ellipses represent the spatially filtered input data (y).

not an obstacle for such a BCI system because training is performed off-line most of the time.

3.6 Spatial - Spectral Filtering Methods

Despite the popularity of the introduced spatial filtering algorithms, optimization of the spatial filters in a fixed spectral band lacks of automatic selection of the frequency bands which vary from subject to subject. In order to avoid incorrect spectral filter selection, Generally, the cut-off frequencies of the band pass filter are either selected

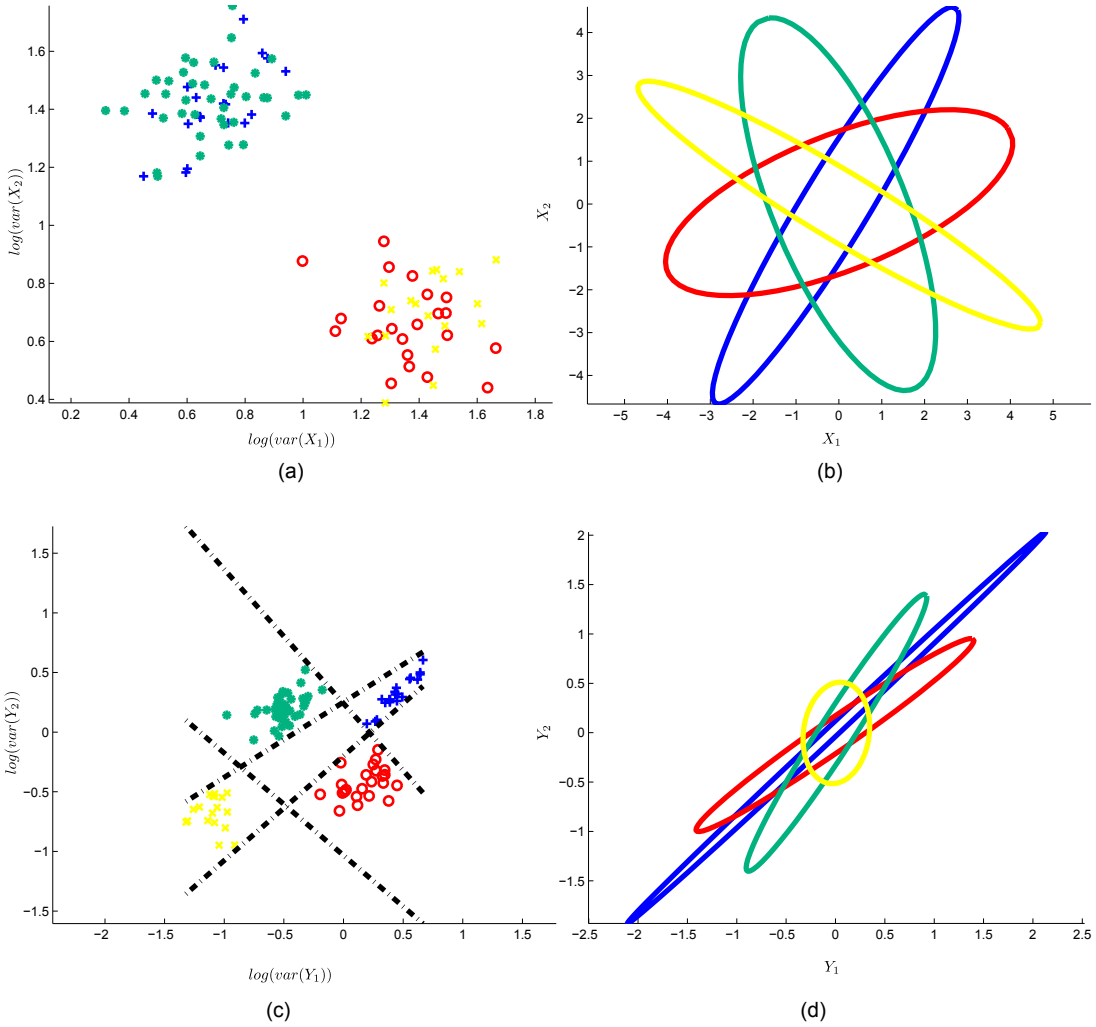

Figure 3.19 : Input data and SFN output data for toy data with 4-classes. (a) log-variance feature for 2-dimensional input data. Note that each point represents an epoch that belongs to class 1 (red circle), class 2 (blue plus), class 3 (green asterisk) or class 4 (yellow cross). (b) Enclosing ellipses represent the input data. (c) SFN spatial filter layer output (f) with generated class borders (black dashed lines) of the classifier layer. (d) Enclosing ellipses represent the spatially filtered input data (y).

manually or unspecifically set to a broad band filter [13], which results in poor classification performance. Manual searching of the best frequency band through the training set is laborious and time consuming [14]. Therefore, optimizing a spectral filter along with the spatial filter is highly desirable [13]. Recently, there are some methods which modifies the spectral characteristic of the filters along with its spatial filter coefficients, named *called spatio-spectral filters*. In this section, some important spatio-spectral filtering algorithms and proposed spatio-spectral filtering method of this study will be introduced.

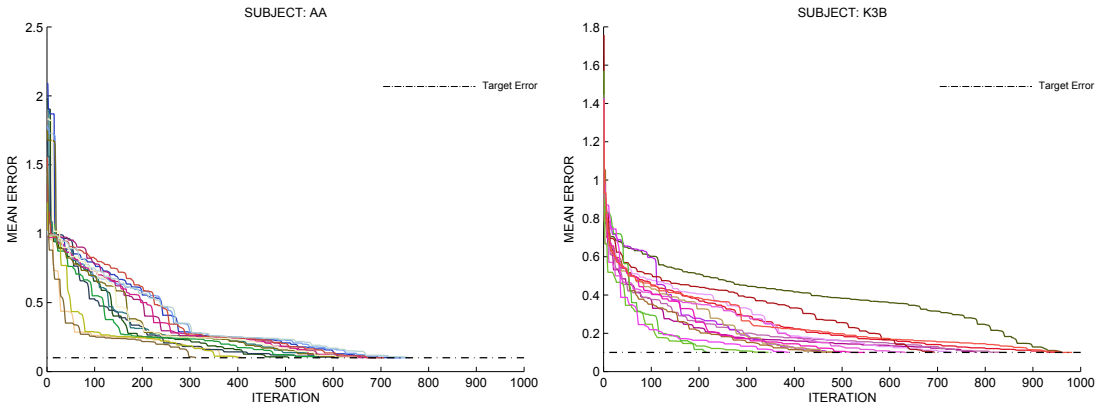

Figure 3.20 : Converge behavior of the SFN. Network had been trained 15 times for subject AA (on the left) and for subject K3B (on the right). Resulting mean error value at each iteration until convergence for each training experiment is displayed in different colors.

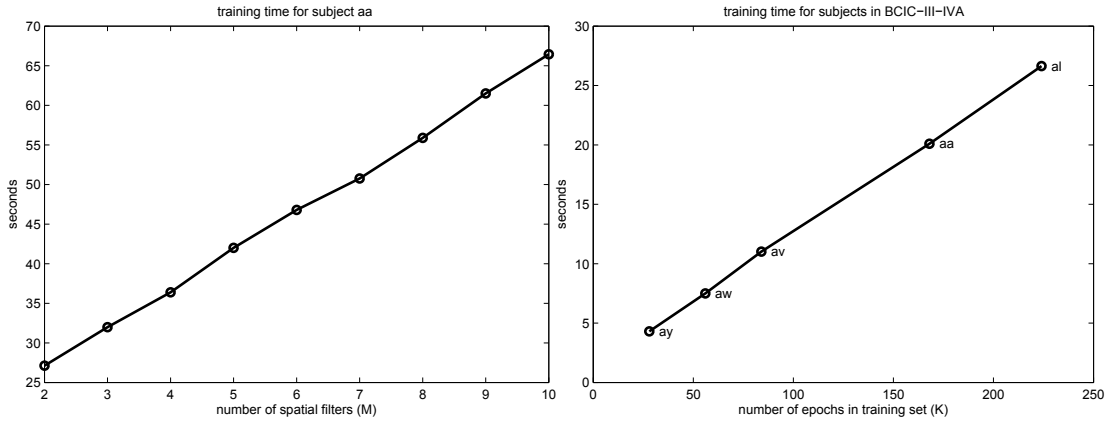

Figure 3.21 : Training time of SFN. Left: number of spatial filters (M) versus training time for subject *aa*. Right: number of epochs versus training time for the subjects of BCIC-III-IVA when $M = 2$. SFN was configured with $EMIN = 0$, $ITRMAX = 1000$, $\mu = 100$ and $B = 2$.

3.6.1 Common spatio-spectral patterns (CSSP)

In CSSP method [138], EEG channels and their time delayed versions are concatenated in order to create a new epoch with higher number of channels two times so that a second order FIR filter is created for each channel. Let $X \in \mathbb{R}^{NxT}$ be an input epoch with N channels and T samples. In CSSP, input signal is formed by sub signals namely X_1 and X_2 where $X_1 \in \mathbb{R}^{NxT-\tau}$, $[X_1]_{n,t} = X_{n,t}$ and $X_2 \in \mathbb{R}^{NxT-\tau}$, $[X_2]_{n,t} = X_{n,t+\tau}$. New epoch will be $\widehat{X} = [X_1^\top, X_1^\top]^\top$ where, $\widehat{X} \in \mathbb{R}^{2Nx(T-\tau)}$. $\widehat{X}$ is given to CSP block and best linear combination of each channel and their time delayed versions are calculated. A block diagram regarding CSSP method is given in Figure 3.22.

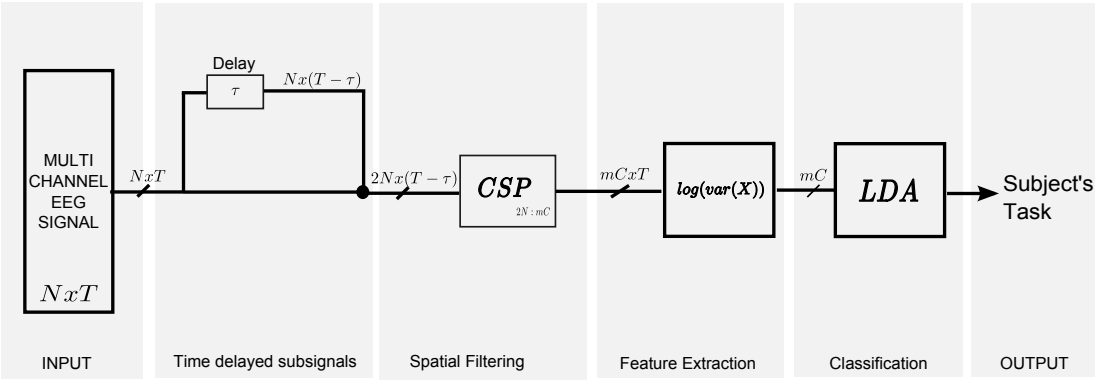

Figure 3.22 : A block diagram representing the common spatio-spectral patterns algorithm.

3.6.2 Common sparse spectral spatial patterns (CSSSP)

Different from CSSP method, one FIR filter and corresponding spatial filter is designed for each class in CSSSP method [113]. In CSSSP, weights of spectral and spatial filters are optimized iteratively. Let the FIR filter degree be T. CSSSP proposes below fitness function for the optimization of FIR filter weights $(b_0, b_1, ..., b_{T-1})$ and spatial filter weights $\boldsymbol{w}$:

$$(\boldsymbol{b}, \boldsymbol{w}) = \max_{b_2,\ldots,b_T} \max_{\boldsymbol{w}} \boldsymbol{w}^\top \left(\sum_{\tau=0}^{T-1} \left(\sum_{j=1}^{T-\tau} b_j b_{j+\tau} \right) \Sigma_c^\tau \right) \boldsymbol{w} + \frac{B}{T} b_1$$

$$\max_{b_2,\ldots,b_T} \max_{\boldsymbol{w}} \boldsymbol{w}^\top \left(\sum_{\tau=0}^{T-1} \left(\sum_{j=1}^{T-\tau} b_j b_{j+\tau} \right) \left(\Sigma_1^\tau + \Sigma_2^\tau \right) \right) \boldsymbol{w} = 1 \tag{3.50}$$

In the above equation,Σ_c^τ the correlation matrix between the input signal X and the delayed signal X_τ, which is,

$$\Sigma_c^\tau = X X_\tau^\top + X_\tau X^\top \tag{3.51}$$

and B is the coefficient which controls the sparsification of $\boldsymbol{b}$. Optimization problem is solved with gradient search methods and CSP method. Figure 3.23 displays a block diagram of the CSSSP algorithm.

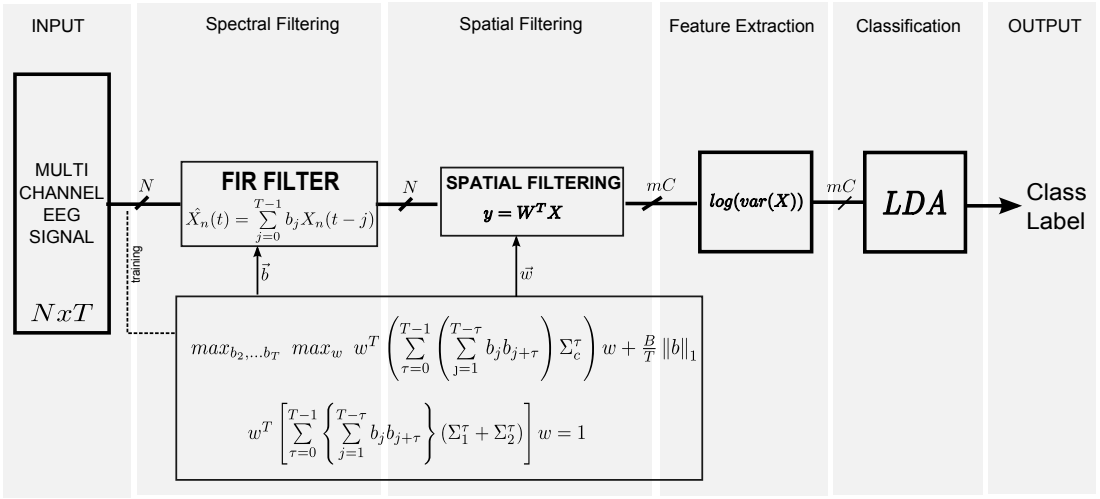

Figure 3.23 : A block diagram representing the common sparse spatio-spectral patterns algorithm.

3.6.3 Filter bank common spatial patterns (FBCSP)

FBCSP [192] algorithm analyzes input EEG signal within various frequency bands. FBCSP incorporates 4 stages. In the first stage, multi channel EEG signal is fed into a filter bank with zero-phase Chebyshev Type II filters. The bands of these filters are pre defined thus; no filter update during runtime is needed. In the second stage, outputs of each filter of the previous stage are given to the separate CSP blocks in order to perform the spatial filtering. Then, band specific feature vectors are created by using the band power. The third stage performs feature selection and employs several feature selection algorithms based on mutual information such as Mutual Information based Best Individual Feature (MIBIF) and Mutual Information-based Naïve Bayesian Parzen Window(MINBPW) [193] algorithms. The selected features is classified in the last stage where many classification methods such as Naïve Bayesian Parzen Window (NBPW), Fisher Linear Discriminant(FLD) and Support Vector Machine (SVM) algorithms may be employed. Figure 3.24 displays a block diagram of the FBCSP algorithm.

3.6.4 Spectrally weighted common spatial patterns (Spec-CSP)

Spec-CSP [139] method uses an iterative algorithm in order to achieve the optimum spectral and spatial filter coefficients. Spec-CSP defines a feature vector with *J* column as:

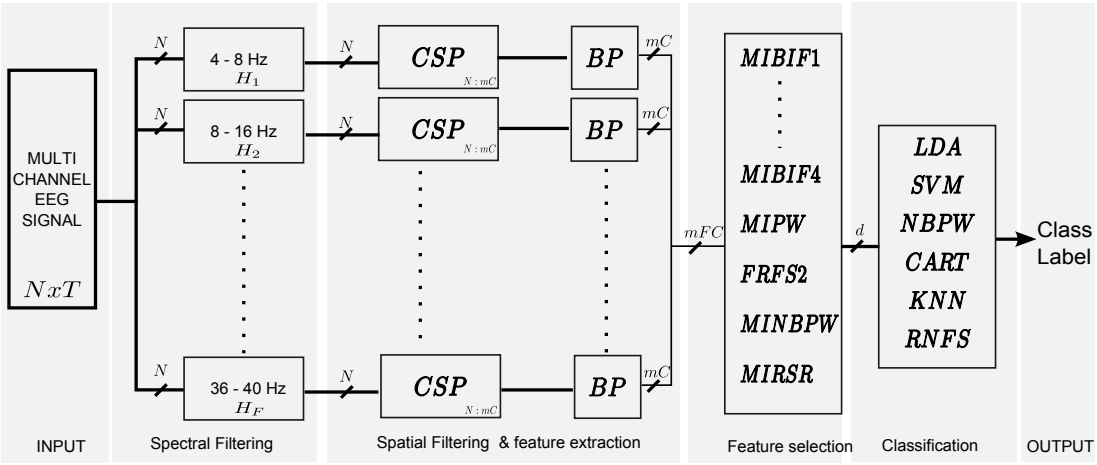

Figure 3.24 : A block diagram representing the filter bank common spatial patterns algorithm.

$$\phi_j(X, \boldsymbol{w}_j, B_j) = \log(\boldsymbol{w}_j^\top X B_j B_j^\top X^\top \boldsymbol{w}_j) \tag{3.52}$$

where, $\boldsymbol{w}_j \in \mathbb{R}^N$ is a spatial filter vector and $B_j \in \mathbb{R}^{TxT}$ is a linear time invariant temporal filter which includes the spectral filter coefficients and localizes the motor imagery signals in frequency domain. Optimization of the spatial filter ($\boldsymbol{w}_j$) is based on the CSP method. Let U is the Fourier transformation matrix where $U = \left\{1/\sqrt{T}e^{-\frac{2\pi ikl}{T}}\right\} \in \mathbb{C}^{TxT}$ and thus, $UU^\top = I_{TxT}$. Embedding $UU^\top$ into the equation gives the sensor covariance matrix:

$$\Sigma^c(B) = \left\langle XUU^\top B_j B_j^\top UU^\top X^\top \right\rangle^c \tag{3.53}$$

Here, $U^\top B_j B_j^\top U = diag(\alpha_1, \alpha_2, ...\alpha_T)$ is a diagonal matrix and includes the spectral weights (α_t) and $\langle\bullet\rangle^c$ denotes for the expected value within class c. Let $\widehat{X} = XU \in \mathbb{C}^{NxT}$ is Fourier transformed input signal and $\hat{x}_t \in \mathbb{C}^N$ is the t^{th} frequency component. Cross spectrum matrix based on α_t is given as follows:

$$\Sigma(\alpha) = \sum_{t=1}^{T} \boldsymbol{\alpha}_t V_k^c \in \mathbb{R}^{NxN} \tag{3.54}$$

where, $V_k \in \mathbb{R}^{NxN}$ is the real part of the k^{th} frequency component in the cross spectrum. In order to optimize α coefficients, the following optimization function is solved with the Fisher Discriminant Analysis (FDA):

$$\boldsymbol{\alpha}_{opt} = \max \frac{s(\boldsymbol{w},\alpha)^1 - s(\boldsymbol{w},\alpha)^2}{var\,(s(\boldsymbol{w},\alpha)^1) + \mathrm{var}\,(s(\boldsymbol{w},\alpha)^2)} \qquad s.t. \quad \boldsymbol{\alpha}_t \geq 0 \tag{3.55}$$

in the above equation, $s(\boldsymbol{w},\alpha)^c = \sum_{t=1}^{T} \boldsymbol{\alpha}_t \boldsymbol{w}^\top V_k \boldsymbol{w}$ and c is the class label. Spatial or spectral coefficients are updated alternatively at each optimization step.

3.6.5 Discriminative filter bank common spatial patterns (DFBCSP)

DFBCSP [140] optimizes spectral and spatial filters alternatively by using an iterative algorithm similar to Spec-CSP. However, while Spec-CSP seeks for a single spectral and a associated spatial filter, DFBCSP designs a set of multiple spectral and associated spatial filters. The type of the spectral filter in DFBCSP is FIR filter which can be defined with a matrix multiply equation. Let $X \in \mathbb{R}^{NxT}$ be the input epoch with N channels and T samples. Filtering the input signal with a P^{th} order FIR filter $\boldsymbol{h}_i \in \mathbb{R}^P$ can be defined as a matrix multiplication:

$$\hat{\boldsymbol{x}}_k = \sum_{p=1}^{P} \boldsymbol{h}_p X_{k+P-p} = \boldsymbol{h}_p A_k \tag{3.56}$$

where, $p = (1,2,\cdots,P)$; $k = (1,2,\cdots,K)$; $K = T - P + 1$; $A_k \in \mathbb{R}^{NxP}$ and $[A_k]_{n,p} = [X]_{n,k+p-P}$. Therefore, obtained variance after spectra-spatial filtering of an input epoch is given:

$$\alpha(X,\boldsymbol{w},\boldsymbol{h}) = \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{w}^\top \hat{A}_k \boldsymbol{h}\boldsymbol{h}^\top \hat{A}_k^\top \boldsymbol{w} \tag{3.57}$$

where, $\boldsymbol{w} \in \mathbb{R}^N$ is the spatial filter and $\hat{A}_k$ is the zero mean input signal defined as: $\hat{A}_k = A_k - \overline{A}_k$. In order to optimize the filters, DFBCSP defines optimization functions as generalized eigen value problems for both spatial and spectral filters. At iteration i, when $\boldsymbol{h}_i$ is known, $\boldsymbol{w}_i$ is solved by the following function:

$$\max_{\boldsymbol{w}_i} \hat{J}_1(\boldsymbol{w}_i|\boldsymbol{h}_i) = \frac{\boldsymbol{w}_i^\top R_c(\boldsymbol{h}_i) \boldsymbol{w}_i}{\boldsymbol{w}_i^\top \left(R_1(\boldsymbol{h}_i) + R_2(\boldsymbol{h}_i)\right) \boldsymbol{w}_i} \tag{3.58}$$

where, $R_c(\boldsymbol{h}_i) = E_{X \in c}\left[\frac{1}{K} \sum_{k=1}^{K} \hat{A}_k \boldsymbol{h}\boldsymbol{h}^\top \hat{A}_k^\top\right] \in \mathbb{R}^{NxN}$. Above optimization problem is solved with classical CSP approach. Then, new spatial filter vector $\boldsymbol{w}_i$ is calculated. Then, since $\boldsymbol{w}_i$ is known, spectral filter vector $\boldsymbol{h}_i$ is found by the following equation:

$$\max_{\boldsymbol{h}_i} \hat{J}_2(\boldsymbol{h}_i|\boldsymbol{w}_i) = \frac{\boldsymbol{h}_i^\top Q_c(\boldsymbol{w}_i)\boldsymbol{h}_i}{\boldsymbol{h}_i^\top \left(Q_1(\boldsymbol{w}_i)+Q_2(\boldsymbol{w}_i)\right)\boldsymbol{h}_i} \tag{3.59}$$

where, $Q_c(\boldsymbol{w}_i) = E_{X\in c}\left[\frac{1}{K}\sum_{k=1}^{K}\hat{A}_k \boldsymbol{w}\boldsymbol{w}^\top \hat{A}_k^\top\right] \in \mathbb{R}^{PxP}$. At each iteration, newly found $\boldsymbol{h}_i$ or $\boldsymbol{w}_i$ should increase or keep unchanged the value of the fitness function $\hat{J}$. Thus, alternating optimization guarantees the monotonic increase so that the convergence. The feature vector is constructed with constructed with F spectral filters $(\boldsymbol{h}_1 \cdots \boldsymbol{h}_F)$ and $2r$ spatial filters $(\boldsymbol{w}_i^1 \cdots \boldsymbol{w}_i^{2r})$ associated with each spectral filter $\boldsymbol{h}_i$:

$$y = \left[\alpha\left(X, \boldsymbol{w}_1^1, \boldsymbol{h}_1\right), \cdots \alpha\left(X, \boldsymbol{w}_1^{2r}, \boldsymbol{h}_1\right), \cdots \alpha\left(X, \boldsymbol{w}_F^1, \boldsymbol{h}_F\right), \cdots \alpha\left(X, \boldsymbol{w}_F^{2r}, \boldsymbol{h}_F\right)\right] \tag{3.60}$$

3.6.6 Filter bank common spatio-spectral patterns (FBCSSP)

The proposed method of this study called "filter bank common spatio - spectral patterns (FBCSSP)" method consists of two CSP layers. Figure 3.25 displays a block diagram of the FBCSSP algorithm. At the first layer, EEG signal is filtered with a couple of FIR band pass filters. Then, each band passed EEG signal with N channels are spatially filtered in the CSP-1 layer so that, best spatial patterns for each frequency bands are determined in this layer. At this point, proposed method differs from FBCSP and SBCSP methods. These methods finalize preprocessing and extract features at the end of the first layer. However, FBCSSP method continues signal preprocessing operation. Obtained CSP-1 outputs are directly given to a second CSP filter, CSP-2. The purpose of CSP-2 is linearly combining the outputs of the first spatial filter layer so that maximum divergence could be obtained.

Let $X \in \mathbb{R}^{NxT}$ be an input EEG signal matrix with T samples and N channels, called as epoch. Firstly, all epochs in the training set are filtered with FIR band pass filters at desired frequencies with degree P,

$$\hat{X}_{f,n}(t) = \sum_{p=0}^{P} \boldsymbol{h}_{f,p} X_n(t-p) \qquad \hat{X}_f \in \mathbb{R}^{NxT} \quad f = (1,2,\cdots F) \tag{3.61}$$

where, $\boldsymbol{h}_{f,p}$ is the p^{th} weight of f^{th} FIR filter. For each FIR filter output, a CSP filter is created. Let the average covariance matrices at the output of the f^{th} filter be R_f^c where, c is the class label,

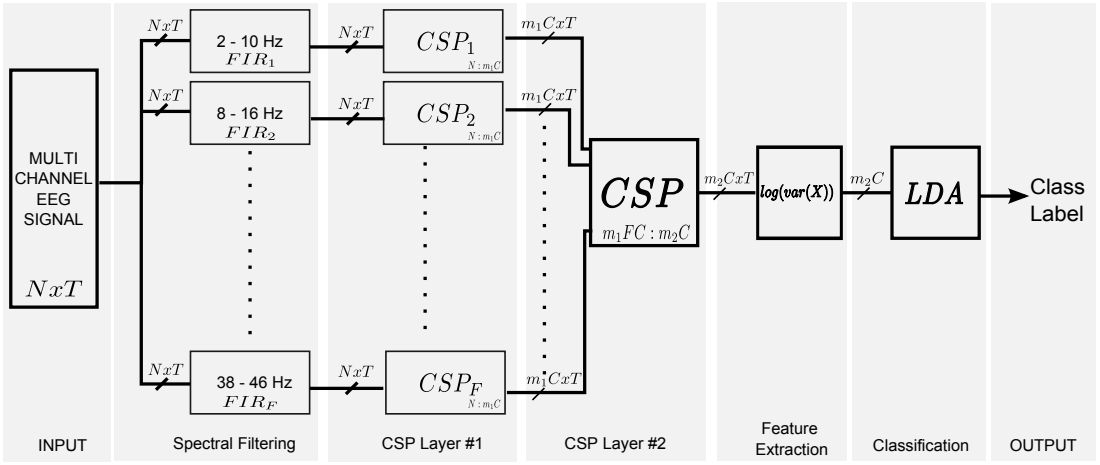

Figure 3.25 : A flowchart regarding filter bank common spatio - spectral patterns method.

$$R_f^c = \frac{1}{K_c} \sum_{k \in c} \hat{X}_f^k \left(\hat{X}_f^k \right)^\top \tag{3.62}$$

Where K_c is the number of epochs which belong to the class c. For two classes, classical CSP approach may be applied. However, in order to find the spatial filters in a multiclass BCI experiment, one versus rest (OVR) CSP method may be applied [173]. Let m_1 o denote the number of spatial filters for one class at the first layer and M_1 to denote the total number of spatial filters where $M_1 = m_1 C$ and C is the total number of classes. Let the obtained spatial filter to be denoted with $U_f \in \mathbb{R}^{NxM_1}$. Then, the output of this spatial filter will be,

$$Y_f = U_f^\top \hat{X}_f \qquad Y_f \in \mathbb{R}^{M_1 xT} \tag{3.63}$$

For the next layer, all outputs of first spatial layer are concatenated row by row and a new epoch is created. Let Y be the new epoch matrix which is defined as,

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_F \end{bmatrix} \qquad Y \in \mathbb{R}^{FM_1 xT} \tag{3.64}$$

The second CSP layer works as a frequency selection. Let the average covariance matrices for this layer be $R^c \in \mathbb{R}^{FM_1 xFM_1}$. R^c is calculated as,

$$R^c = \frac{1}{K_c} \sum_{k \in c} Y^k \left(Y^k\right) \top \tag{3.65}$$

Again, classical CSP or OVR CSP methods may be used for calculation of the spatial filter matrix. Let m_2 to be the number of spatial filters for each class at the second layer. Then, the total number of spatial filters in this layer will be $M_2 = m_2 C$. If we denote $W \in \mathbb{R}^{FM_1 x M_2}$ as the spatial filter matrix of the second label, the output of this layer will be,

$$Z = W^\top Y \qquad\qquad Z \in \mathbb{R}^{M_2 x T} \tag{3.66}$$

The following feature extraction and classification blocks are applied just as in the classical CSP method. In order to create a feature set, band power (BP) is used given as,

$$\boldsymbol{f}_j = \log\left(\frac{\text{var}(y^j)}{\sum_{l=1}^{M_2} \text{var}(y^l)}\right) \qquad j = 1, 2, \cdots M_2 \tag{3.67}$$

where represents the column number in feature vector . In the above equation, logarithm function is used for approximating the feature distribution to a normal distribution [132].

3.6.6.1 Filter bank selection

The filter bank used in FBCSSP may be configured according to the requirements and prior information related with the processed signal. For motor imagery signals, a filter bank covering 8 Hz to 36 Hz is reasonable. However, FBCSSP has the capability to combine the output filters and finally generate an optimized spectral filter. Therefore, it is better to choose a filter bank which cover a wide frequency band in which the banks overlaps.

While creating the filter bank structure, having linear phase response is the most important point because in the second CSP layer, a spectral combination operation is done. FIR filters are appropriate option for being linear phase response.

Here, the effect of linear combining the filter bank outputs will be analyzed. Embedding (3.61) into (3.63) gives the output of the first layer in terms of the input and the spectral filter parameters,

$$Y_f(t) = U_f^\top \sum_{p=0}^{P} \boldsymbol{h}_{f,p} X(t-p) \tag{3.68}$$

By using (3.66), it is possible to write the overall filter within one equation,

$$Z(t) = \sum_{f=1}^{F} W_f^T U_f^\top \sum_{p=0}^{P} \boldsymbol{h}_{f,p} X(t-p) \tag{3.69}$$

where, $W_f \in \mathbb{R}^{M_1 x M_2}$ is the spatial filter matrix in the second layer, associated with the f^{th} output of the first layer. By organizing this equation, we reach the equation,

$$Z(t) = \sum_{p=0}^{P} \delta_p^\top X(t-p) \tag{3.70}$$

which is the characteristic equation of the proposed spatio - spectral filter. $\delta_p \in \mathbb{R}^{NxM_2}$ holds the spatial and spectral characteristics of the FBCSSP filter and it is defined as the following equation.

$$\delta_p = \sum_{f=1}^{F} U_f W_f \tag{3.71}$$

Above result yields, linear combination of the outputs of linear FIR filters means linear combination of the filter coefficients. So, the both CSP layers together bring up a FIR filter which is a combination of the banks in the filter bank. Therefore, using overlapped and numerous filters should give more flexible FIR filters. An example filter bank frequency response is given in Figure 3.26. Here, there are 7 FIR filters that cover the entire frequency band and their linear combination

Figure 3.27 demonstrates the frequency selection capability of the FBCSSP filter. In this figure, there are three channels and two classes, which are plotted in time domain on the top sub figure. Formulas for each channel for the both classes are also given on the right. Here, $\mathcal{N}(0,1)$ means gaussian distribution with zero mean and unit variance. The only difference between two classes are on the second channel at 24Hz frequency. Therefore, FBCSSP should focus to this frequency. Bottom left sub figure displays the average spectral density of the three channels varying over time. Here, y axis is the frequency, x axis is the time and the power amplitude is coded with gray tones (black:low white:high). In order to train the FBCSSP, it was feed with the input

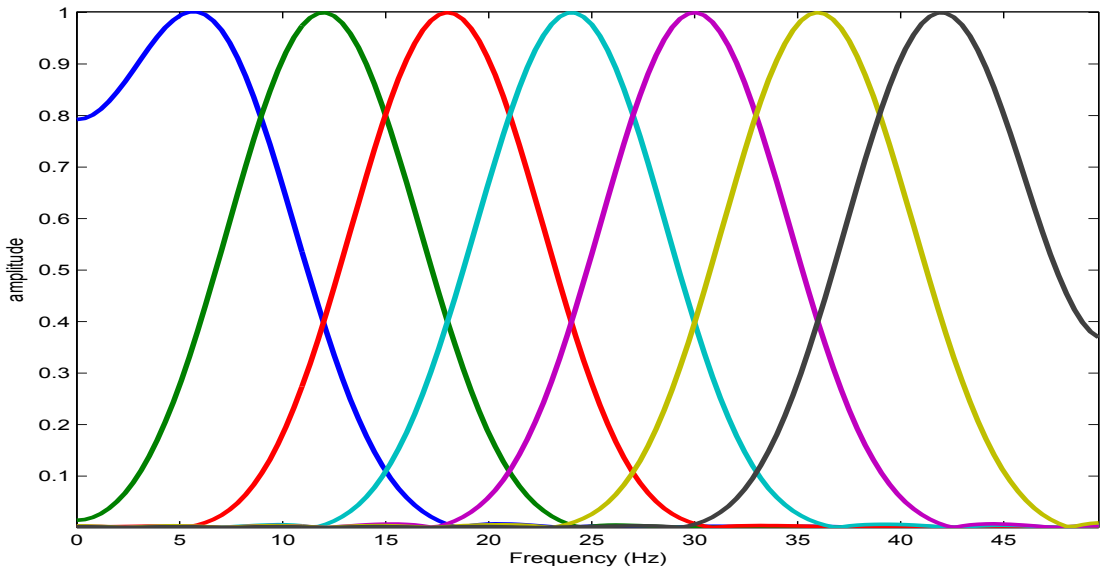

Figure 3.26 : Frequency responses of the selected FIR filters used in the study.

signal. The filter bank selected as given in Figure 3.26. Also note that, $m_1 = 1$ and $m_2 = 1$ for the FBCSSP layers 1 and 2, respectively. On the bottom right sub figure, the spectral characteristic of the trained FBCSSP filter is displayed. This plot was obtained by scanning the FBCSSP structure with varying frequency signals and measuring the average output power. This figure shows that FBCSSP is successful at finding the spectral properties of the input signal.

3.6.6.2 Conclusion

Normally, CSP filter extracts independent components while simultaneously diagonalizing of two covariance matrices. So, applying the CSP method to the output of another CSP filter will not improve the divergence of the signal since the output of the first CSP filter is linearly independent. However in the proposed method, when joined altogether, the outputs of the first layer will become a non-independent multi channel signal thus, CSP of second layer should increase the overall divergence of the incoming signal. In fact, second CSP makes a spectral weighting while linearly combining the outputs of the first layer.

Since they are linear filters of the same type, instead of cascading CSP filters one after another, a single CSP could be used. Indeed, this should give a higher Rayleigh ratio then the FBCSSP. However, this CSP matrix filter would have inputs and outputs and classifying performance will not be higher as expected.

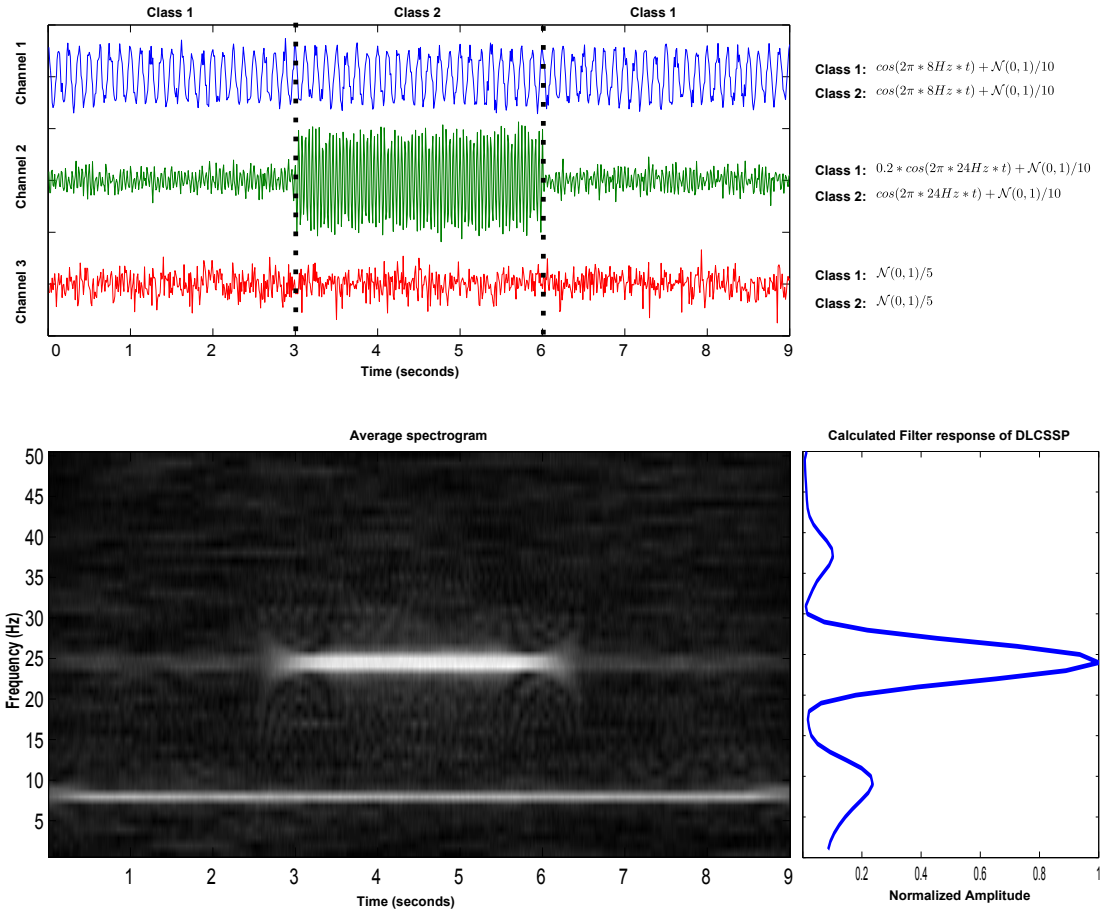

Figure 3.27 : Toy data example for the FBCSSP method. Top figure: time plot of the input signal with equations. Bottom left: Spectral density of the input signal. Bottom right: output filter obtained with FBCSSP algorithm.

Different to various spectral filter optimization methods reported in the literature, FBCSSP searches for the linear combinations of some predefined FIR filters. FBCSSP method has some advantages over these methods. Firstly, computational complexity of the algorithm will not increase with the degree of the FIR filters. Because only the outputs of the filters are being used, algorithm doesn't try to manipulate the filter weights. Whereas, those methods face with increasing computational complexity with the increasing filter degree.

Secondly, FBCSSP method gives the flexibility of defining various FIR filters. This makes the algorithm to embed the existing prior knowledge into the spectra-spatial filters. For example, one can design FIR filters especially at the spectral region of and waves and ignore the other frequencies. Third advantage of the FBCSSP is its non-iterative structure. The methods in [120, 139, 140] use an alternating optimization strategy which iteratively increases the fitness function by updating spatial and spectral filter parameters, respectively. However, FBCSSP calculates spectral and spatial filters

within one pass. Therefore, training time of the proposed method should take much less time then the iterated algorithms. Above mentioned methods search for a spatial and an associated spectral filter set. However, different spatial locations at different frequency bands may be activated in execution of motor imagery. This leads to spatial patterns specific to frequency band. FBCSSP method does not ignore this assumption and produces spatial filters for each defined frequency bands. This enables us to investigate the obtained spatial-spectral filters at a specific frequency band.

4. COMPUTER SIMULATIONS

In this chapter, outcomes of the methods described in the previous chapter will be analyzed. The results of the methods will not only be presented in terms of numerical classification performance, but also obtained spatial and spectral filters of the proposed methods will be illustrated so that physiological plausibility of them will be demonstrated. Prior to the results, the data used in the study will be introduced with applied preprocessing steps. Then, outcomes of each method will be presented with classification accuracies compared with the similar methods in the literature.

4.1 Data Description

In this study, we preferred to use publicly available motor imagery data sets rather than designing a custom test setup and acquiring our own data for many reasons. Firstly, we need verification of the algorithms that are described in the literature and rewritten in the scope of this thesis. Obtaining similar results with the original papers encouraged us to make contribution with novel methods. Secondly, comparing a new algorithm with the fellow methods in the literature would be more suitable if the same data sets were used. Thirdly, upon the publishing of the method, one can reproduce the algorithm and obtain the same results if the data set is available online. Also, there are uncountable number of methods in the literature using the same data sets for similar reasons. Besides, our scope in this thesis was contribution to the literature with newly developed algorithms rather then acquiring motor imagery data.

The motor imagery data used in this study is collected from online and freely available datasets, which are shared with researches in a set of competitions named *BCI competitions* organized by Wolpaw, Blankertz, Pfurtscheller and Birbaumer from 2001 to 2008 [128, 129, 194]. BCI Competitions are organized in order to foster the development of improved BCI technology. In each competition a variety of data sets was made publicly available online. Each data set is a record of brain signals from BCI experiments of leading laboratories in BCI technology split into two parts: one part of

labeled data ('training set') and another part of unlabeled data ('test set'). Researchers worldwide could tune their methods to the training data and submit the output of their translation algorithms for the test data. The truth about the test data was kept secret until, after the deadline, it was used to evaluate the submissions. The name of the data sets used in this study are *BCI competition III Data Set IVa* and *BCI competition III Data Set IIIa*, which are two and four class data sets, respectively.

4.1.1 BCI competition III data set IVa

This dataset including two classes, (available from [195]) consists EEG recordings of 5 subjects who performed motor imagery of *right hand* and *foot*. The recording was made using Brain Amp amplifiers and a 128 channel Ag/AgCl electrode cap from ECI [196] . 118 EEG channels were measured at positions of the extended international 10/20-system. Locations of EEG channels used in the experiment are displayed on Figure 4.1. Signals were band-pass filtered between 0.05 and 200 Hz and then digitized at 1000 Hz with 16 bit (0.1 μV) accuracy. They provides also a version of the data that is down sampled at 100 Hz (by picking each 10th sample) that were typically used for analysis.

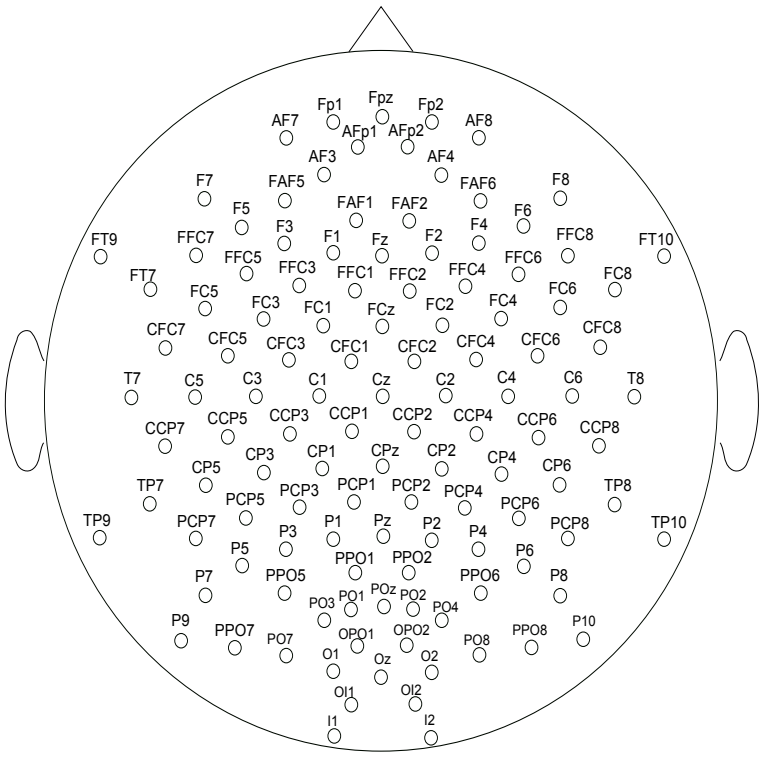

Figure 4.1 : EEG channels used in BCI competition III Data Set IVa. EEG was captured with 118 electrodes according to the extended international 10/20-system.

4.1.1.1 Experiment procedure

This data set was recorded from five healthy subjects. Subjects sat in a comfortable chair with arms resting on armrests. Visual cues indicated for 3.5 s which of the following 3 motor imageries the subject should perform: (L) *left hand*, (R) *right hand*, (F) *right foot*. However, only cues for the classes *right* and *foot* are provided for the competition data set. The presentation of target cues were intermitted by periods of random length, 1.75 to 2.25 s, in which the subject could relax. Figure 4.2 shows timing diagram for the dataset.

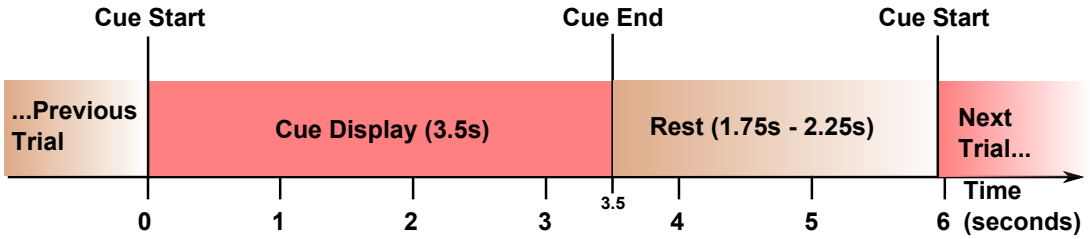

Figure 4.2 : Timing diagram for the BCI competition III Data Set IVa.

There are 280 trials for each subject. However, the numbers of training and test sets differ for each subject: Training and test sizes are given in Table 4.1.

Table 4.1 : Subject labels, train and test sizes for BCI competition III Data Set IVa.

Subject Name	Train Size	Test Size
AA	168	112
AL	224	56
AV	84	196
AW	56	224
AY	28	252

4.1.1.2 Data format

The data are provided in Matlab format (*.mat) conaining these variables:

- `cnt`: the continuous EEG signals, size [time x channels] . The array is stored in data type INT16 . To convert it to μV values, use `cnt=0.1*double(cnt)` in Matlab.
- `mrk`: structure of target cue information with fields

  - `pos`: vector of positions of the cue in the EEG signals given in unit sample, length=number of cues.
  - `y`: vector of target classes (1,2 or `NAN`), length=number of cues.
  - `className`: cell aray of class names

- `info`: structure providing additional information with fields
  - `name`: name of the data set
  - `fs`: sampling rate
  - `clab`: cell array of channel labels
  - `xpos`: x-position of electrodes in a 2d-projection
  - `ypos`: y-position of electrodes in a 2d-projection

4.1.2 BCI competition III data set IIIa

Data Set IIIa is a multi class cued data set with a 60-channel EEG signal sampled at 250 Hz and recorded from 3 subjects. The data set includes four classes named *left hand*, *right hand*, *foot*, and *tongue*. For each subject, there are minimum 60 trials per class; however, some of the trials are marked with *rejected trial*.

The recording was made with a 64-channel EEG amplifier from Neuroscan [197], using the left mastoid for reference and the right mastoid as ground. Locations of the EEG channels are displayed in Figure 4.3. The EEG was sampled with 250 Hz, it was filtered between 1 and 50Hz with Notchfilter on. The data of all runs was concatenated and converted into the GDF format [198].

4.1.2.1 Experiment procedure

The subject sat in a relaxing chair with armrests. The task was to perform imagery left hand, right hand, foot or tongue movements according to a cue. The order of cues was random. The experiment consists of several runs (at least 6) with 40 trials each each; after trial begin, the first 2s were quite, at t=2s an acoustic stimulus indicated the beginning of the trial, and a cross (+) is displayed; then from t=3s an arrow to the left, right, up or down was displayed for 1 s; at the same time the subject was asked to imagine a left hand, right hand, tongue or foot movement, respectively, until the cross

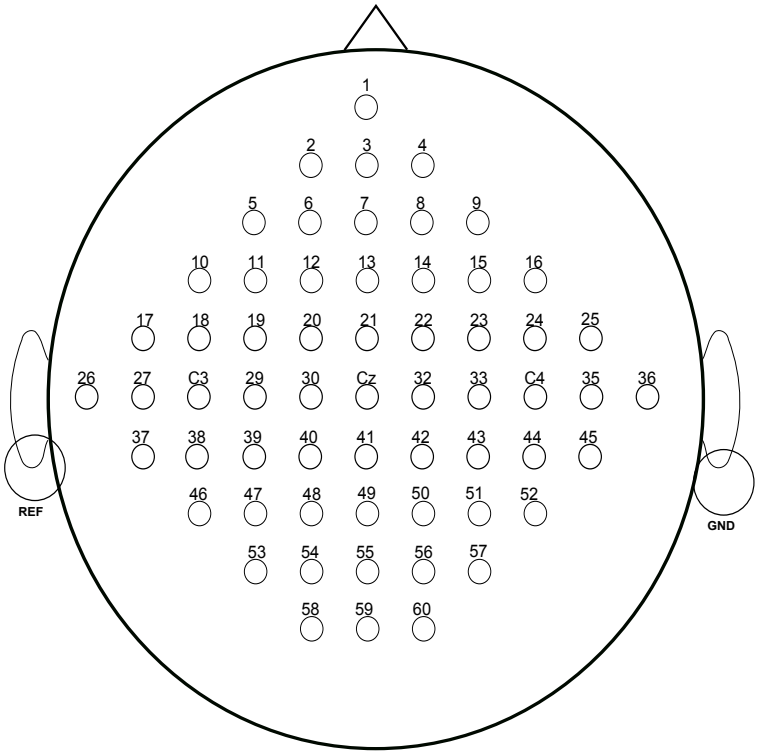

Figure 4.3 : EEG channels used in BCI competition III Data Set IIIa. EEG was captured with 60 channel EEG.

disappeared at t=7s. Each of the 4 cues was displayed 10 times within each run in a randomized order. Timing diagram for a trial is shown in Figure 4.4.

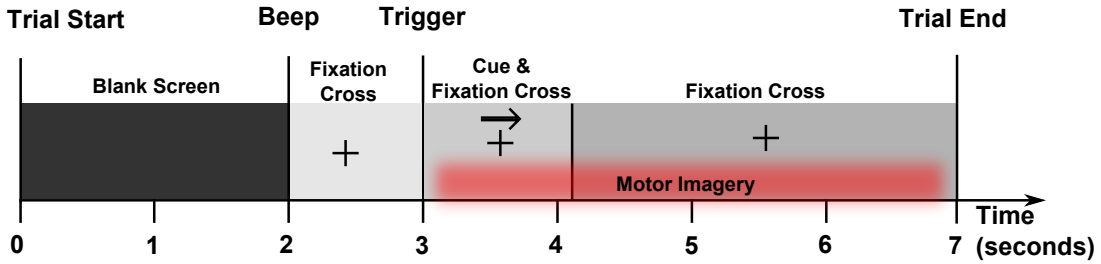

Figure 4.4 : Timing diagram for the BCI competition III Data Set IIIa.

4.1.2.2 Data format

The data is stored in the GDF format [198] and can be loaded into Matlab or Octave with Biosig-toolbox [199] (version 0.81 or higher) using the command `[s,HDR] = sload(filename)`. The data s can contains `NaN`'s; these `NaN`'s indicate the breaks in between the runs or saturation of the analog-to-digital converter. Note that, `'OVERFLOWDETECTION:OFF'` option turns off automatic overflow detection. The data in `s` and `HDR` are explained below.

- `s`: the continuous EEG signals, size [samples x channels] .

- HDR: structured data about the signal. Important elements of HDR structure is listed as follows:
  - NS: number of channels in the signal
  - NRec: number of samples in the signal
  - SampleRate: sample rate in Hertz
  - EVENT: structured data about all events. Event data includes the following items:
    - TYP: event type. For a list of event types, see [200].
    - POS: start time of an event (in samples)
    - DUR: duration of an event (in samples)
  - Label: associated electrode name of each column in s
  - Classlabel: class labels of each trial. (1,2,3,4 or NAN). Values '1','2','3','4' indicate the labels of the training set, NAN indicates the trials of the test set.
  - TRIG: The beginning of each trial (t = 0s according to Figure 4.4).
  - ArtifactSelection: indicates trials with artifacts which were visually identified

4.2 ERD Signal Demonstration

The goal of this section is to analyze the real data and validate it by extracting the event related desynchronisation (ERD) signal. Figure 4.5 shows the block diagram for the demo application.In this example, epoch data is loaded by parsing, filtering and cropping functions respectively. The parameters for these functions are given in the same figure. Then, average signal for both classes' power is obtained by using the square operation. Finally, the signal is smoothed with moving average function and plotted. The Matlab code for the given example may be found in Appendix A.1.

Figure 4.6 displays ERD signal for the classes right hand and foot. This image obtained by calculating the average power variation of all epochs in the given data set. The selected data set is dataset IVA and the selected subject is *AY* since output obtained from this subject reveals the ERD signal more clear even if no spatial filter were used.

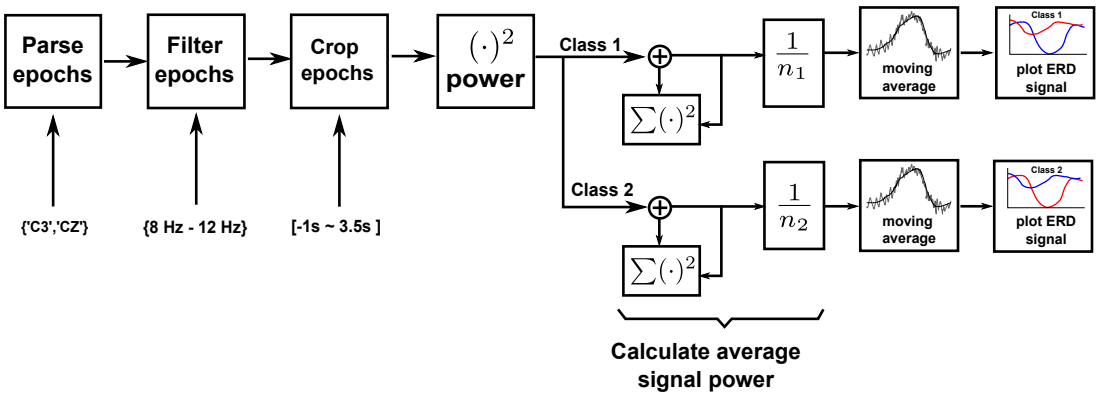

Figure 4.5 : Flow diagram of a sample application for ERD signal demonstration.

Blue and red plots on the figure represents channels `C3` and `CZ`, respectively. The reason for selecting these two channels is because of their spatial locations under the scalp correspond to the areas of *right hand* and *foot* in te motor cortex. Dashed vertical line at time $t = 0$ indicates the instant when the cue was shown to the subject. Note that, filter pass band was selected as 8 to 12 Hz in which the motor imagery ERD is most likely to happen.

Looking at the Figure 4.6, ERD signal is clearly distinguished for the both classes at about 1 to 3 seconds after the trigger with a significant power decrease at both channels. Note that, ERD is more dominant at channel `C3` for class *right hand* on the top plot while it is channel `CZ` for class *foot* on the bottom plot which is an evidence of relation with the corresponding motor imagery tasks.

4.3 Data Preprocessing

In this section, the process of loading datasets and preparing them for classification operation will be described in details. It is possible to separate the preprocessing phase into sub processes as given in Figure 4.7. In this figure, each block represents a subprocess inheriting the data from the previous block. Small blocks at the bottom represents the required parameter set for the given processing block. At the end of the preprocessing, train and test sets are obtained for the classification phase.

The preprocessing starts with parsing the input EEG data into epochs according to the data description specific to the given data set. EEG in both of the data sets are a single chunk of multi dimensional signal containing huge amount of data. In order to prepare this data for classification, it is required to parse the input signal into smaller chunks

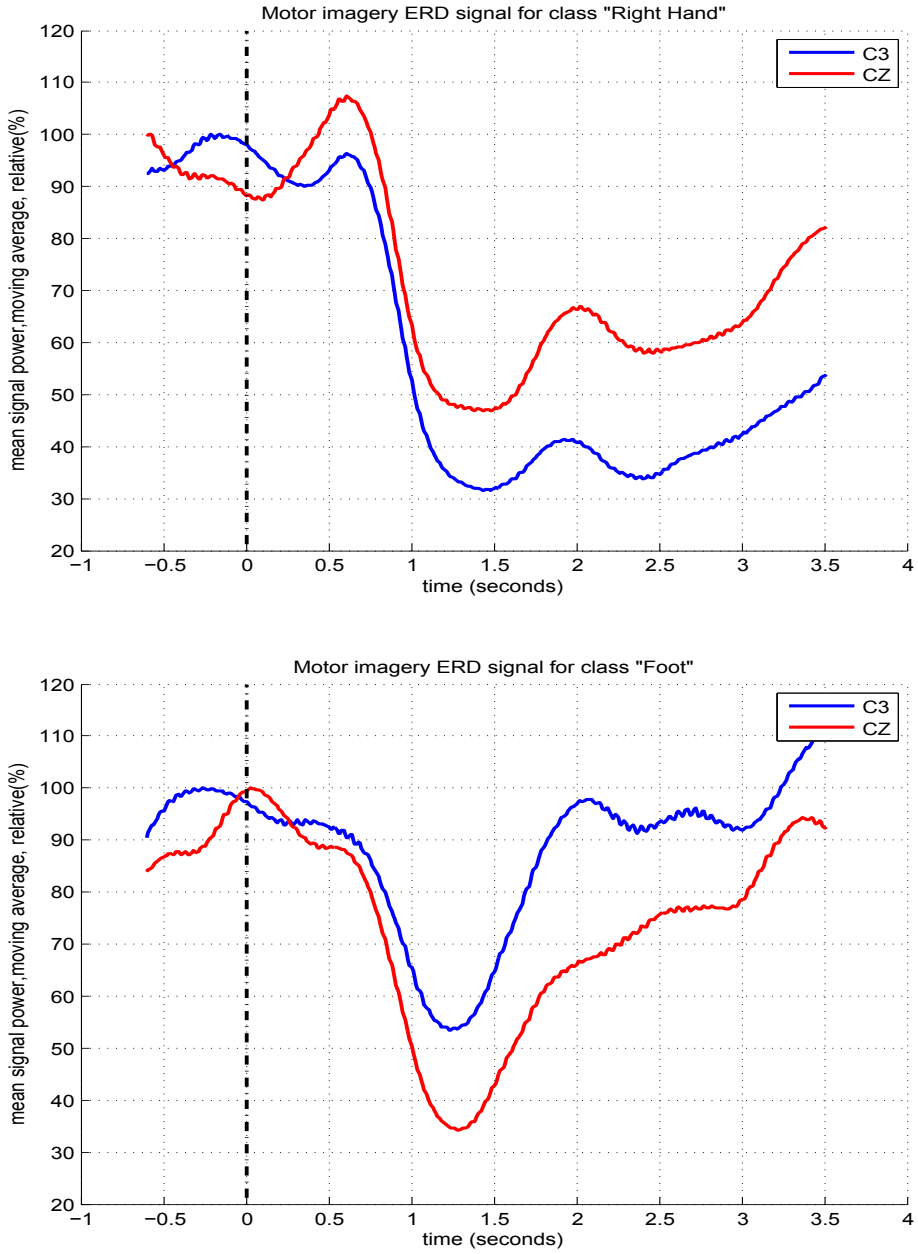

Figure 4.6 : ERD signal obtained from subject *ay*. Dashed vertical line indicates start of trial.

of data which is called *epoch*. The procedure for extracting the epochs from this data are defined in the related data description. Also, one should decide the set of channels that will be analyzed during the classification procedure. Therefore, epoch parsing step contains channel selection.

Matlab functions implementing the epoch parsing process are listed in Appendices A.2 and A.3. With the epoch parser functions, the channel names should be given. The used channels for both data sets were selected so that they roughly cover all over the motor cortex. The selected channels for BCI competition III Data Set IVa and IIIA

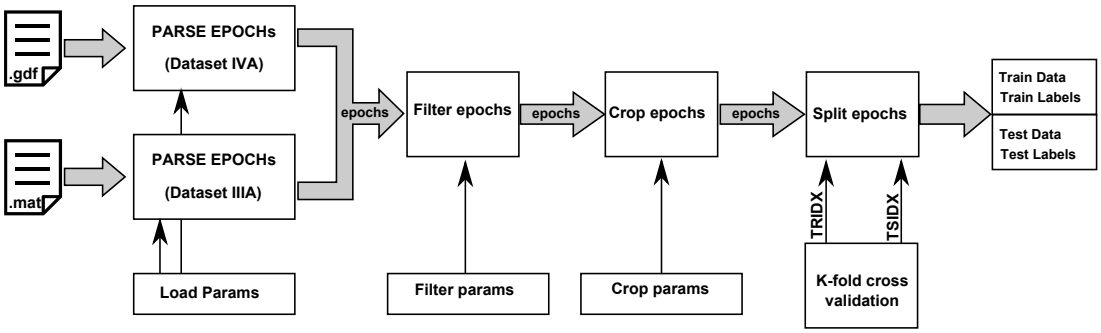

Figure 4.7 : Preprocessing flow diagram.

data sets are given in Figure 4.8. The number of channels used were 68 for Data Set IVa and 43 for Data Set IIIa.

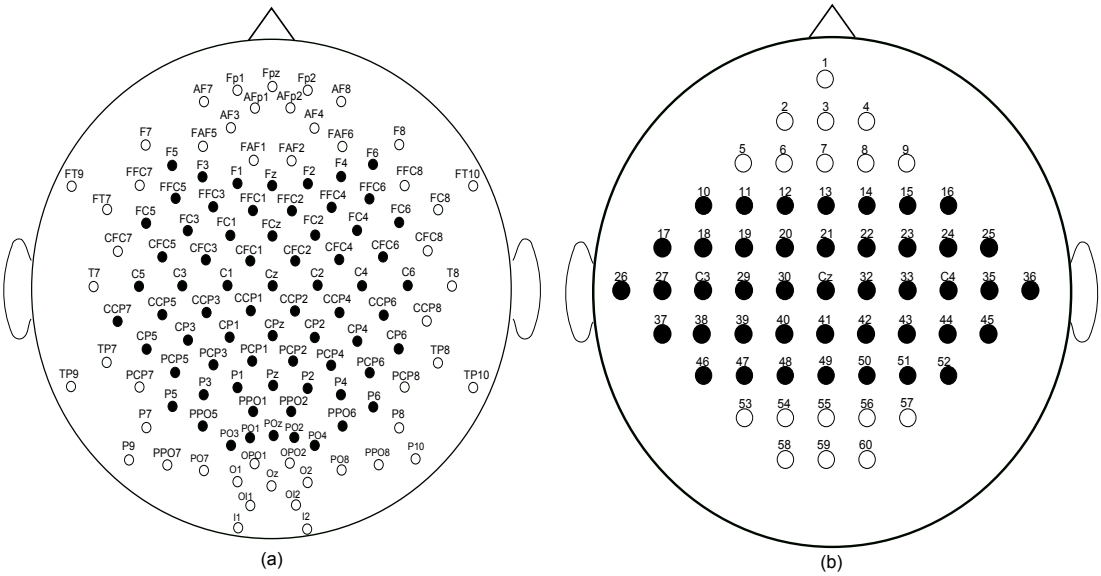

Figure 4.8 : Selected EEG channels for BCI competition III Data Set IVa (a) and BCI competition III Data Set IIIa (b).

For the both data sets, and for all methods except spatio-spectral filtering methods, the filter used was fifth order Butterworth band pass filter with cut off frequencies 8 and 30 Hz. However, for spatio spectral filtering methods, cut off frequencies were selected as 1 and 49 Hz since those methods should optimize the spectral response by themselves. Frequency response of the designed filters are displayed in Figure 4.9. Matlab function for epoch filtering algorithm is given in Appendix A.4.

In order to focus on motor imagery related signals, epoch cropping function was utilized. The timing parameters of this function is 0.5 s to 2.5 s after the trigger. Therefore, total duration of an epoch after epoch cropping function should be 2 seconds. Matlab function for epoch filtering algorithm is given in Appendix A.5.

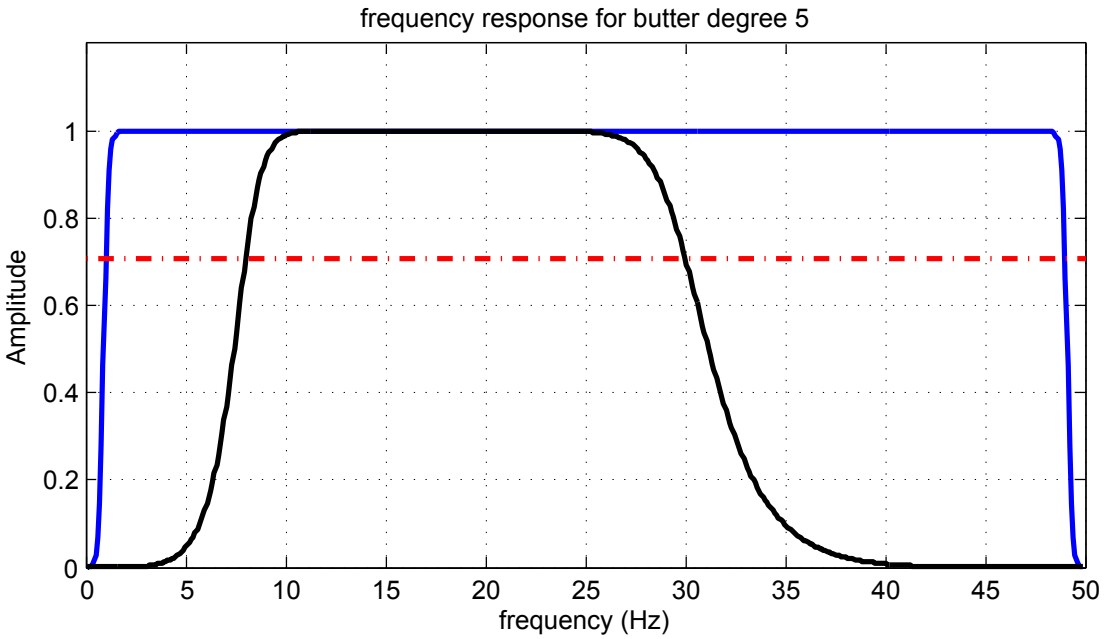

Figure 4.9 : Frequency response of the selected filters for preprocessing. Blue line is 5th order 1-49 Hz Butterworth filter. Black line is 5th order 8-30 Hz Butterworth filter. Red dashed line indicates the -3dB $(1/\sqrt{2})$ cut off value.

When partitioning the epochs into training and test sets, when using the conventional method (e.g. partitioning the data set into two sets of 70% for training and 30% for test), the problem is that the error (e.g. Root Mean Square Error) on the training set in the conventional validation is not a useful estimator of model performance and thus the error on the test data set does not properly represent the assessment of model performance.This may be because there is not enough data available or there is not a good distribution and spread of data to partition it into separate training and test sets in the conventional validation method. In these cases, a fair way to properly estimate model prediction performance is to use cross-validation as a powerful technique [201].

Cross-validation, sometimes called rotation estimation, is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.

In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k-1$ subsamples are used as training

data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data.

Figure 4.10 illustrates this statement. Black rectangles on this figure represents the samples selected as test set and white ones as training set. Each horizontal lane represents one of the folds with 20/80 test/train size.

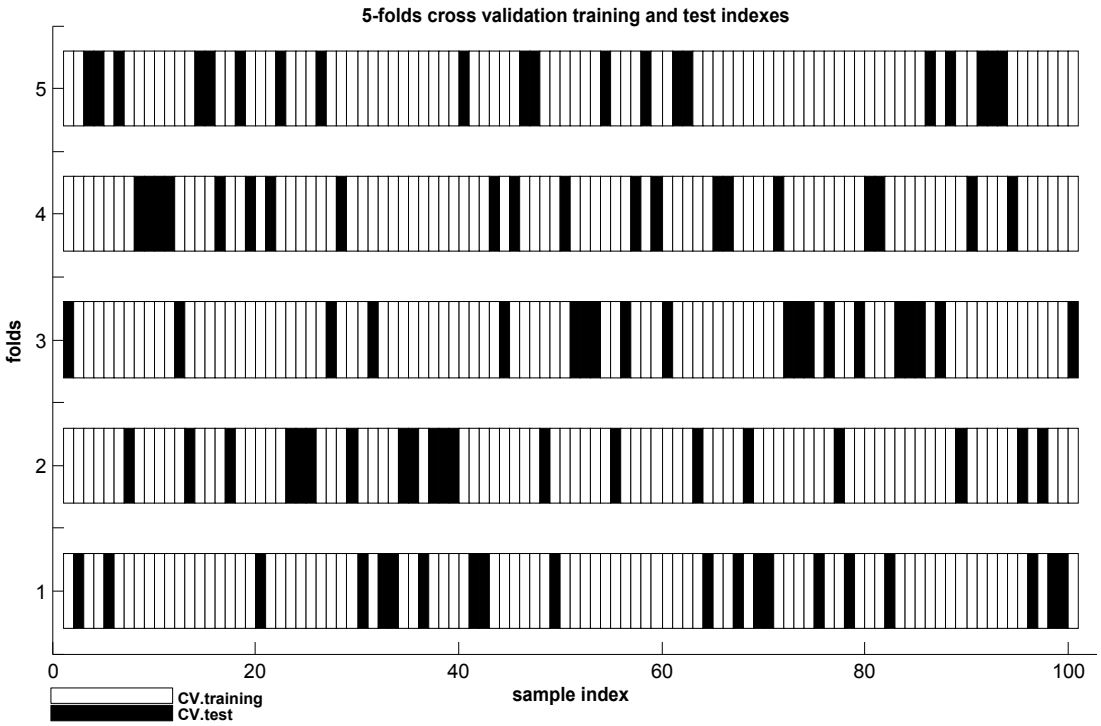

Figure 4.10 : An example for displaying train/test sets in a five-fold cross validation.

For splitting the epochs and creating the training and test sets, K-fold cross validation was used with $K = 10$. Therefore 90% of the whole data set were used as training set while remaining 10% used for evaluating. K-fold cross validation ensures that all epochs in the data set were used in test set. Since the training and test data sets varies for all folds, the subsequent data processing steps (training, testing and performance evaluation) were repeated as many times as the number of folds. Matlab function for epoch splitting is given in Appendix A.6.

4.4 Data Processing Phase

In this section, the procedure for evaluating the classification methods that were presented in the previous chapters will be introduced. A block diagram summarizing the data processing phase is given in Figure 4.11.

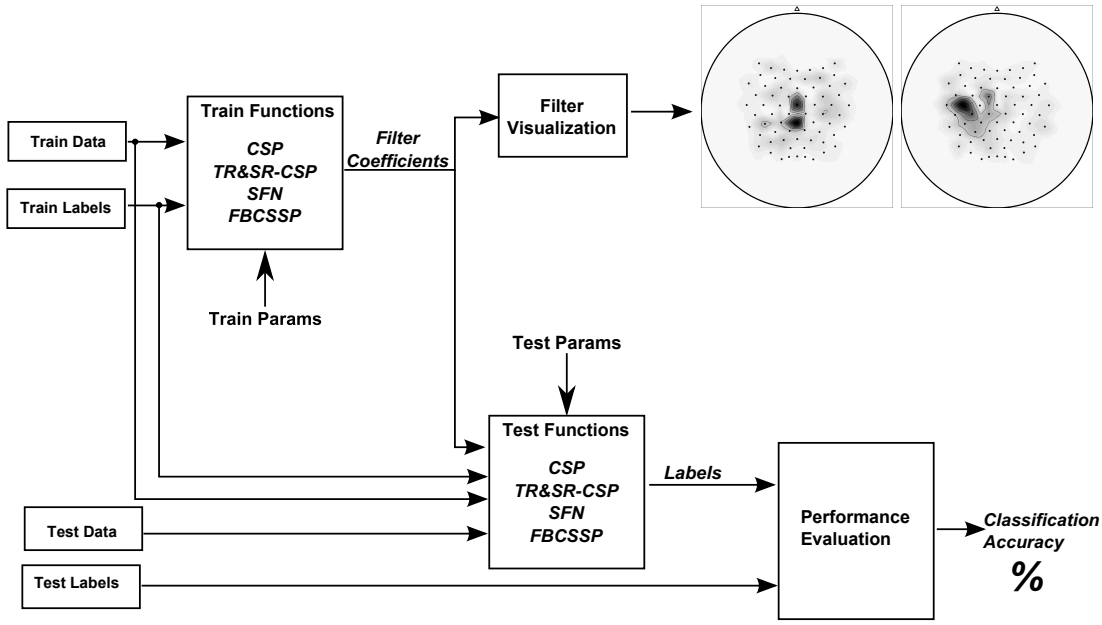

Figure 4.11 : Block diagram for data processing.

Data processing starts with training functions which optimizes the spatial and spectral filters in accordance with the given training data. Training functions output optimized filter coefficients. Then, this coefficients are given to test functions along with test data without test labels in order to estimate the labels of the test data. Note that, Matlab code implementing the proposed methods are given in Appendices A.7 to A.9. At the end of the testing process, performance of the system is calculated as classification accuracy by using estimated and true labels of the test data. Also, output of the training functions may be used for visualizing the spatial and spectral filters in order to demonstrate the physiological plausibility of the filter coefficients.

4.4.1 Configuration of evaluated methods

In the following paragraphs, the methods that were evaluated will be listed with the configuration specific to the method itself. As described in the Methods section, presented classification algorithms have at least one setting values that is called *hyper-parameter*. For each subjects and each method, hyper parameters are selected from a list and the best combination of the parameter values that gives the highest average performance within a K-fold cross validation is reported under the obtained classification accuracy. An example figure demonstrating the parameter selection is given in Table 4.2. In this figure, there are two parameters called m and α where, the possible values of m and α are m_1 to m_M and α_1 to α_A. Therefore, there are MxA

combinations of the two parameters. The *best* parameter configuration is indicated by the highlighted row since the mean performance value of the selected row is the highest one.

Table 4.2 : An example table for selecting the best hype parameter combination. The highlighted row has the maximum average performance.

	K-fold cross validation						
Configuration	$k=1$	$k=2$	$k=3$	.....	$k=K$	Mean	Std
$m=m_1;\alpha=\alpha_1$	67%	82%	76%	.....	83%	67%	3.25
$m=m_1;\alpha=\alpha_2^*$	88%	92%	78%	.....	86%	84%	6.42
⋮	⋮	⋮	⋮	.....	⋮	⋮	⋮
$m=m_1;\alpha=\alpha_A$	63%	66%	71%	.....	70%	75%	7.54
$m=m_2;\alpha=\alpha_1$	69%	72%	67%	.....	74%	79%	4.45
⋮	⋮	⋮	⋮	.....	⋮	⋮	⋮
$m=m_M;\alpha=\alpha_A$	72%	71%	69%	.....	62%	70%	5.37

4.4.1.1 CSP

CSP method uses *m* as a hyper parameter which stands for the number of spatial filters used for constructing the spatial filter. Possible values for *m* was selected from $\{1,2,3,4,5\}$. Then for each subject, all possible *m* value was tried and the best *m* that gave the highest mean performance is selected.

4.4.1.2 TRCSP

As in CSP method, Tikhonov regularized CSP uses *m* as a hyper parameter which stands for the number of spatial filters used for constructing the spatial filter. Also, with the regularized CSP methods, α parameter, which is defined in (3.15), is a multiplier determining the effect of the penalty term . *m* was selected from the set $\{1,2,3,4,5\}$ and α was selected from the set $\{0.01,0.025,0.05,0.075,0.1,0.25,0.5,0.75\}$.

4.4.1.3 SRCSP

The hyper parameters used with spatially regularized CSP method are *m*, α and *r* where *r* is an hyperparameter which defines how far two electrodes can be to be still considered as close to each other [134]. The set of possible values for *m*

is $\{1,2,3,4,5\}$, for α is $\{0.01,0.025,0.05,0.075,0.1,0.25,0.5,0.75\}$ and for r is $\{0.1,0.25,0.5,0.75,1,1.25,1.5\}$ so that, total 280 combinations were tried for each subject (and for each fold).

4.4.1.4 sCSP

The hyper parameters used with stationary CSP method are m and α. The set of possible values for m is $\{1,2,3,4,5\}$ and for α is $\{0.01,0.025,0.05,0.075,0.1,0.25,0.5,0.75\}$. Also, chunk size ($v$) for the sCSP method was set to 1 so that covariances were applied for a period of a single trial.

4.4.1.5 TR&SR-CSP

TR&SR-CSP method uses m for number of spatial filters per class, α for multiplier of the penalty term and r for a scale parameter for electrode distance measure as hyper parameters. The possible values for r were chosen out of $\{1,2,3,4,5\}$, for α it was $\{0.01,0.025,0.05,0.075,0.1,0.25,0.5,0.75\}$ and for r, $\{0.1,0.25,0.5,0.75,1,1.25,1.5\}$. Also, center electrodes for subjects were selected as, {'C3' and 'CZ'} for 2 class data set BCI competition III Data Set IVa and {'C3', 'C4','CZ','Ch26'} for BCI competition III Data Set IIIa, which is a 4 classes data set.

4.4.1.6 SFN

SFN method uses M for the number of neurons in hidden layer which is the total number of spatial filters. For compatibility, m parameter where $M = mC$ where C is the number of classes and m is the number of spatial filters per class. m was chosen out of $\{1,2,3,4,5\}$. Other parameter settings related with SFN are listed below:

- `sfnparams.EMIN`: 0.1
- `sfnparams.ITRMAX`: 1000
- `sfnparams.trainMethod`: 'LM'
- `sfnparams.mu`: 100
- `sfnparams.beta`: 2

4.4.1.7 CSSP

For common spatio spectral pattern algorithm, the best combination for m and τ are searched where, τ is the delay parameter. The possible values for m were chosen out of $\{1,2,3,4,5\}$ and for τ, possible values are chosen out of $\{1,...,10\}$.

4.4.1.8 CSSSP

With common sparse spatio spectral algorithm, the order of the filter was 20. m was chosen out of $\{1,2,3,4,5\}$ and B, which is the regularization parameter that affects the sparsity of $\boldsymbol{b}$ the set of possible values is $\{0,0.01,0.1,0.25,0.5,1,2,5\}$. For optimizing the filter coefficients, gradient descent method was used.

4.4.1.9 DFBCSP

with DFBCSP method, the order of the filter was 20. m was chosen out of $\{1,2,3,4,5\}$ and F, which is the number of filters were chosen out of $\{1,...10\}$.

4.4.1.10 FBCSP

For FBCSP method, the filter bank used was a FIR filter bank including 7 FIR filters with cut off frequencies [2,10; 8,16;14,22; 20,28;26,34;32,40;38,46]. The degree of the filters was set to 20. The frequency responses of the filters in the filter bank configured for evaluation are given in Figure 3.26. The parameter m for FBCSP method was chosen out of $\{1,2,3,4,5\}$ and the number of features selected (d) was chosen out of $\{1,2,3\cdots 10\}$.

4.4.1.11 FBCSSP

In FBCSSP method, the hyper parameters that were tried for the best combination are $\{m_1,m_2\}$ which are the number of spatial filters for the first and the second CSP layers per class, respectively. the values of m_1 and m_2 are selected from the list $\{1,2,3,4,5\}$ so that there are 25 combinations for each subject.

The FIR filter bank used for FBCSSP method includes 7 FIR filters with cut off frequencies [2,10; 8,16;14,22; 20,28;26,34;32,40;38,46]. The degree of the filters

was set to 20. The frequency responses of the filters in the filter bank configured for evaluation are given in Figure 3.26.

4.4.2 Performance criteria

In a study based on data classification, the performance of the proposed classifier should be reported by some performance metrics that are generally accepted. By knowing the true labels and predicted labels, these performance metrics may be calculated. For a multi class classification system, the output of true labels and predicted labels may be tabulated as in Figure 4.12 which is called *confusion matrix*.

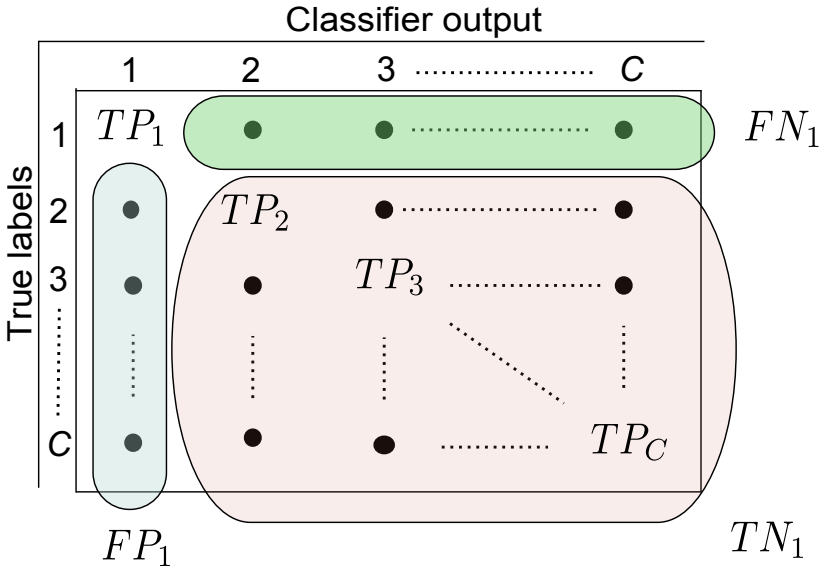

Figure 4.12 : Confusion matrix generated for performance evaluation. *C*=total number of classes, *TP*=true positive, *TN*=true negative, *FP*=false positive, *FN*=false negative.

The performance measure used in this study is overall accuracy in percentages which is calculated by the following equation,

$$ACC\% = 100 \sum_{c=1}^{C} \frac{TP_c}{TP_c + TN_c + FP_c + FN_c} \tag{4.1}$$

Where, C is the total number of classes in the data set. Detailed information about performance metrics for a multi class system may be found in [202]. The function implemented for performance evaluation called `perfCalc` which is listed in the Appendix A.10 evaluates accuracy and other performance measurements that are *precision*, *sensitivity*, *specificity* and *f-score*

4.4.3 Filter visualization technique

Most of the studies about motor imagery classification found in the literature demonstrates the physiological plausibility of calculated spatial filters by color coding spatial filter amplitudes and mapping corresponding channel locations of the spatial filter on a head image. Since the number of electrodes is not enough to cover all over the head area, interpolating methods are being used in order to get a smooth color transition between electrodes. Example spatial filter illustrations are given in Figure 4.13. Also, it is possible to insert electrode names or a contour plot with determined number of slices as given in the figure.

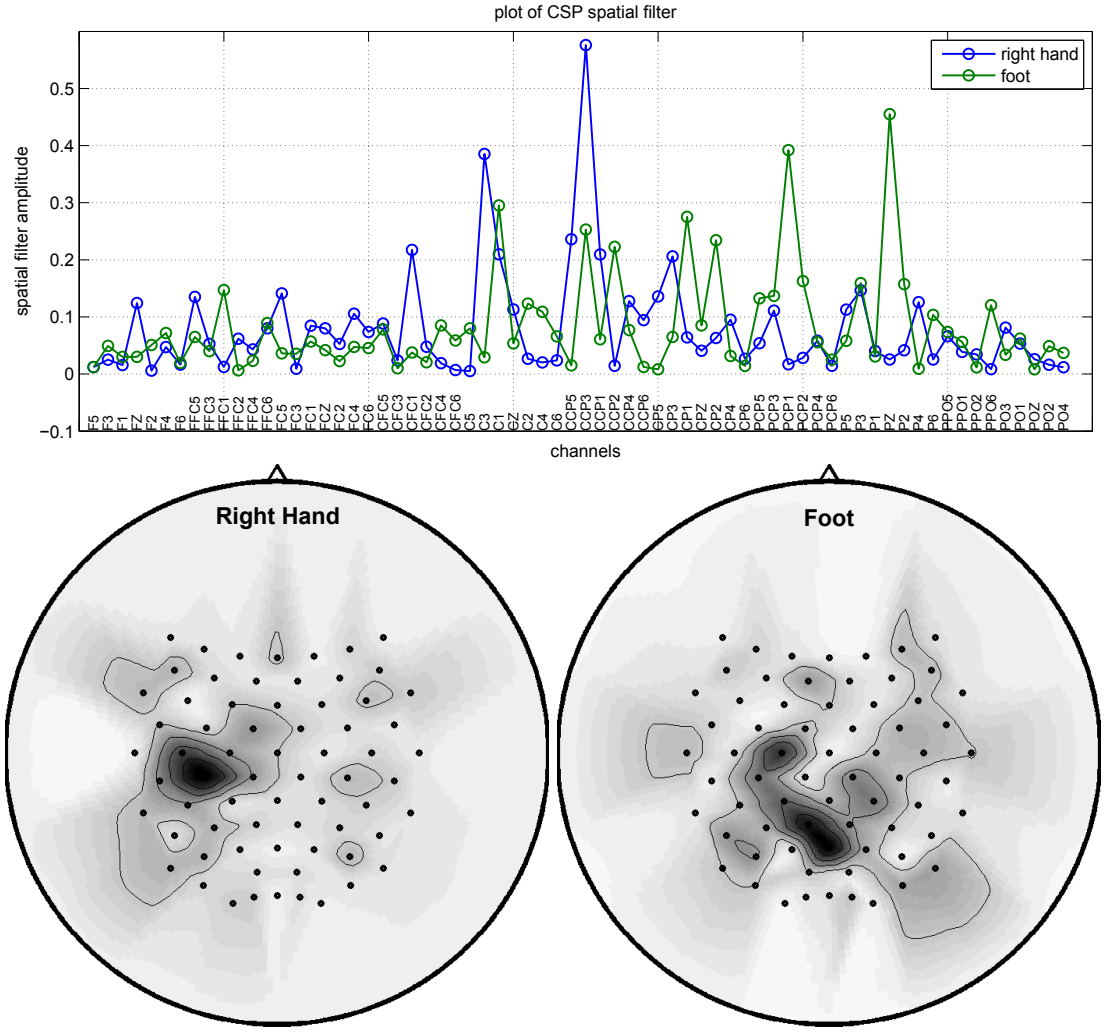

Figure 4.13 : Example figures for spatial filter visualization. Spatial filter coefficients are color coded according to amplitude values and interpolated on an imaginary head displaying electrode locations.

In order to visualize the spatial filters, Matlab function called `img_head` was coded. This function accepts spatial filter coefficient values and used channel names as input

and outputs a head figure according to the given input parameters. Matlab code for `img_head` function is listed in Appendix A.11.

4.5 Performance Evaluation Results

Tables 4.3 and 4.4 list the performance evaluation results for the given data sets. The first table contains the results for data set IVA which is a 2 class data set with 5 subjects whereas, second one displays the results for data set IIIA, which is the data set with 4 classes and 3 subjects.

Both tables report classification accuracies of each method for each subject. The formula of the percentage accuracy (*ACC%*) was given in equation (4.1). The tables also list the standard deviation (std) of any method for any subject. Since 10-fold cross validation was used for evaluating the classification accuracy, std. represents the standard deviation of all folds for the given subject and method.

The last column lists the overall accuracy and standard deviation for any method and all subjects which summarizes the corresponding method's classification performance.

The number(s) in parenthesizes in any cell notifies the selected values of parameters for the corresponding subject and the method. Also, the names of the parameters is given in the second column. Note that, the accuracy value in each cell is the outcome of the given parameter configuration. The procedure about selecting the optimal values for the parameters was described in subsection 4.4.1.

Since the supplied data is subjective, methods' performances highly vary along the subjects. Therefore, along with the quantitative performance summary supplied by the tables, graphical presentations of the performances which benchmark the listed methods with box plots subject by subject are given in Figures 4.14 and 4.15. The first figure shows the classification accuracies of the subjects belong to the data set IVA while the second one is for the data set IIIA.

In these figures, the horizontal axis is the methods that were evaluated and the vertical axis is the evaluated performance value. For any subject and any method, the box boundaries represent the upper and lower 25% quantiles of the input data which is the output of the 10 fold cross validation for the selected configuration. The red horizontal lines inside or on the boundary of the boxes represent the median values. The whiskers

Table 4.3 : Classification performances of the listed methods for the subjects in BCI competition III Data Set IVa.

METHOD		SUBJECTS					Average
		aa	***al***	***av***	***aw***	***ay***	
CSP	ACC (%)	83.57	97.50	72.85	92.5	94.28	88.14
	std.	±9.85	±3.39	±10.13	±7.23	±3.84	±9.99
	(m)	(2)	(3)	(5)	(2)	(4)	
TRCSP	ACC (%)	86.07	96.43	73.92	87.5	93.57	87.5
	std.	±4.60	±3.37	±6.53	±3.04	±4.39	±8.70
	(m,α)	(4;0.01)	(4;0.01)	(5;0.05)	(3;0.01)	(2;0.01)	
SRCSP	ACC (%)	85.71	96.43	75	86.42	95	87.71
	std.	±4.46	±3.37	±6.07	±6.90	±5.64	±8.61
	(m,α,r)	(3;0.01;0.10)	(3;0.01;0.10)	(3;0.01;0.10)	(3;0.01;0.10)	(3;0.01;0.10)	
sCSP	ACC (%)	86.07	97.86	79.64	94.64	96.07	90.85
	std.	±8.33	±3.45	±7.54	±5.89	±6.4	±7.73
	(m,α,r)	(5;0.07)	(3;0.05)	(5;0.03)	(3;0.01)	(4;0.03)	
TR&SR-CSP	ACC (%)	91.07	98.57	81.07	96.43	96.79	92.79
	std.	±5.39	±2.50	±8.43	±2.38	±3.55	±7.12
	(m,α,r)	(5;0.01;1.25)	(3;0.01;0.75)	(5;0.01;0.50)	(3;0.01;1.25)	(5;0.01;0.75)	
SFN	ACC (%)	85.00	96.07	76.79	91.43	93.22	88.50
	std.	±5.27	±5.18	±9.41	±4.82	±3.55	±7.71
	(m)	(1)	(1)	(2)	(3)	(2)	
CSSP	ACC (%)	87.85	95.21	75.71	94.64	96.07	90.5
	std.	±4.52	±1.88	±8.38	±7.76	±4.27	±9.13
	(m,τ)	(4;10)	(4;2)	(5;7)	(3;3)	(1;13)	
CSSSP	ACC (%)	89.13	96.73	74.43	95.85	94.27	90.09
	std.	±1.67	±3.87	±5.12	±2.64	±3.57	±9.23
	(m,B)	(1;0)	(2;0.1)	(2;1)	(2;0.1)	(2;0)	
FBCSP	ACC (%)	90.46	98.54	67.43	97.72	96.07	90.05
	std.	±4.31	±2.23	±6.63	±2.47	±3.63	±13.03
	(m,d)	(2;4)	(2;3)	(5;10)	(3;8)	(1;4)	
DFBCSP	ACC (%)	92.33	98.54	77.78	97.73	94.69	92.21
	std.	±3.25	±1.78	±4.27	±1.28	±3.54	±8.44
	(m,F)	(1;3)	(4;5)	(1;2)	(1;1)	(2;6)	
FBCSSP	ACC (%)	92.51	98.93	80.72	97.51	97.51	93.43
	std.	±3.12	±1.72	±9.55	±2.41	±2.41	±7.51
	(m_1,m_2)	(1;1)	(1;1)	(4;4)	(5;3)	(1;5)	

(dashed lines above and below the boxes) extend to the most extreme data points the box plot algorithm considers to be not outliers, and the outliers are plotted with red cross marks individually.

Despite the fact that classification accuracies of the methods highly varies from subject to subject because of the inter-subject variability, when these methods are compared to each other within one subject, there isn't a huge gap between them. However, there is a slight increase of performance from conventional CSP method to regularized CSP methods and spatio spectral methods. Also, it is obvious that spatio-spectral filtering algorithms generally achieve higher classification rates than the methods which merely works in spatial domain.

The methods that were proposed within the scope of this study (TR&SR-CSP, SFN and FBCSSP) output fairly good results when compared to similar methods. For example,

Table 4.4 : Classification performances of the listed methods for the subjects in BCI competition III Data Set IIIa.

METHOD		SUBJECTS			Average
		k3b	***k6b***	***l1b***	
CSP	ACC (%) std. (m)	89.92 ±5.73 (4)	58.92 ±12.73 (3)	79.04 ±14.24 (2)	75.96 ±17.15
TRCSP	ACC (%) std. (m,α)	87.24 ±5.87 (4;0.01)	62.45 ±13.85 (4;0.01)	77.06 ±13.59 (2;0.01)	75.59 ±15.32
SRCSP	ACC (%) std. (m,α,r)	88.92 ±6.20 (4;0.01;0.10)	63.40 ±10.16 (5;0.01;0.25)	77.87 ±10.01 (2;0.01;0.5)	76.73 ±13.71
sCSP	ACC (%) std. (m,α,r)	90.56 ±5.30 (4;0.01)	60.52 ±12.09 (2;0.025)	79.08 ±11.61 (3;0.01)	76.72 ±15.95
TR&SR-CSP	ACC (%) std. (m,α,r)	93.29 ±3.51 (4;0.01;1.25)	68.63 ±9.34 (5;0.01;0.1)	79.82 ±11.67 (5;0.01;0.50)	80.58 ±13.35
SFN	ACC (%) std. (m)	87.93 ±7.09 (1)	63.56 ±12.29 (1)	80.70 ±11.47 (2)	77.40 ±14.54
CSSP	ACC (%) std. (m,τ)	91.63 ±3.90 (5;4)	62.42 ±10.23 (5;2)	73.57 ±14.35 (3;8)	75.87 ±15.84
CSSSP	ACC (%) std. (m,B)	90.91 ±3.71 (2;0.1)	64.53 ±8.20 (6;0.1)	78.18 ±9.44 (8;0)	77.88 ±13.15
FBCSP	ACC (%) std. (m,d)	92.43 ±2.84 (2;6)	70.71 ±7.57 (4;10)	80.29 ±7.69 (3;8)	81.14 ±10.97
DFBCSP	ACC (%) std. (m,F)	93.75 ±3.54 (4;6)	73.20 ±6.44 (5;10)	84.75 ±8.44 (4;8)	83.90 ±10.59
FBCSSP	ACC (%) std. (m_1,m_2)	95.28 ±3.70 (3;5)	79.94 ±8.04 (5;3)	85.66 ±8.80 (2;1)	86.96 ±9.48

TR&SR-CSP method's performance is always higher than the other regularized CSP methods. Likewise, FBCSSP algorithm outperforms the other spatio-spectral methods most of the time. When compared to CSP method, SFN gives similar average performances however, the learning rule of the SFN algorithm makes it sensitive to the initial network weights. Therefore, SFN may produce higher results with better initial network weights. Because the initial weights of SFN are set randomly, the accuracy for any subject and M value vary across each running of the algorithm. This situation may be improved with better and more robust selection of the initial network weights, which provides constantly higher results. However, increasing the robustness of the initial weight selection is beyond the scope of this paper.

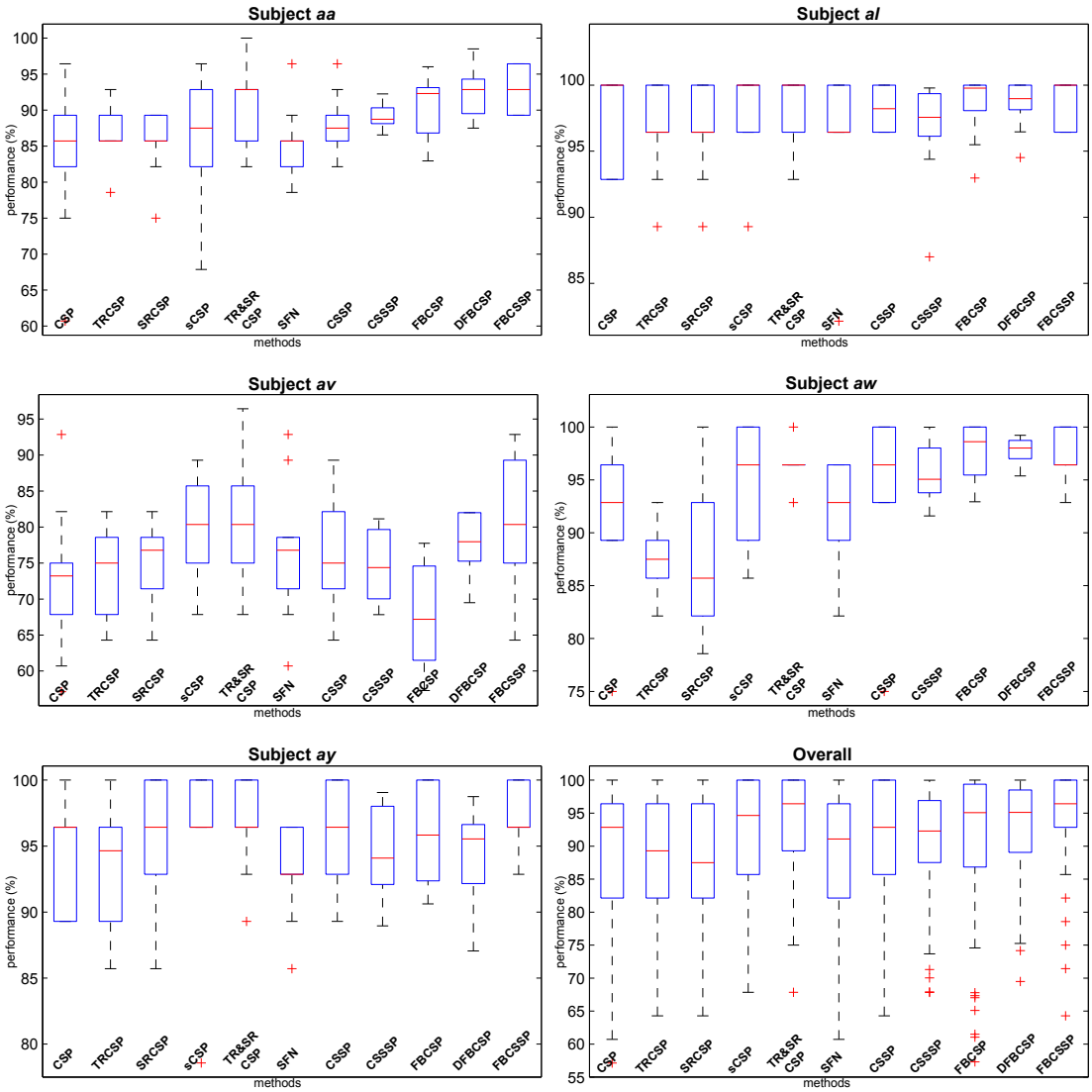

Figure 4.14 : Boxplots displaying the disturbance of classificaion performances for the subjects in dataset IVA.

The results implies that, FBCSSP method is a successful motor imagery classification algorithm with its simple structure, non-iterative and fast training method and increased classification accuracies. Being a spatio-spectral filtering algorithm, this method also produces physiologically agreeable filters not only in spatial domain, but also in spectral domain, which will be described in the next subsections.

4.5.1 Spatial filters visualizations

In this subsection, obtained spatial filters will be presented for demonstrating the physiological plausibility of the proposed methods. Figure 4.16 displays the primary motor and somatosensory areas of motor cortex on a human brain with approximate

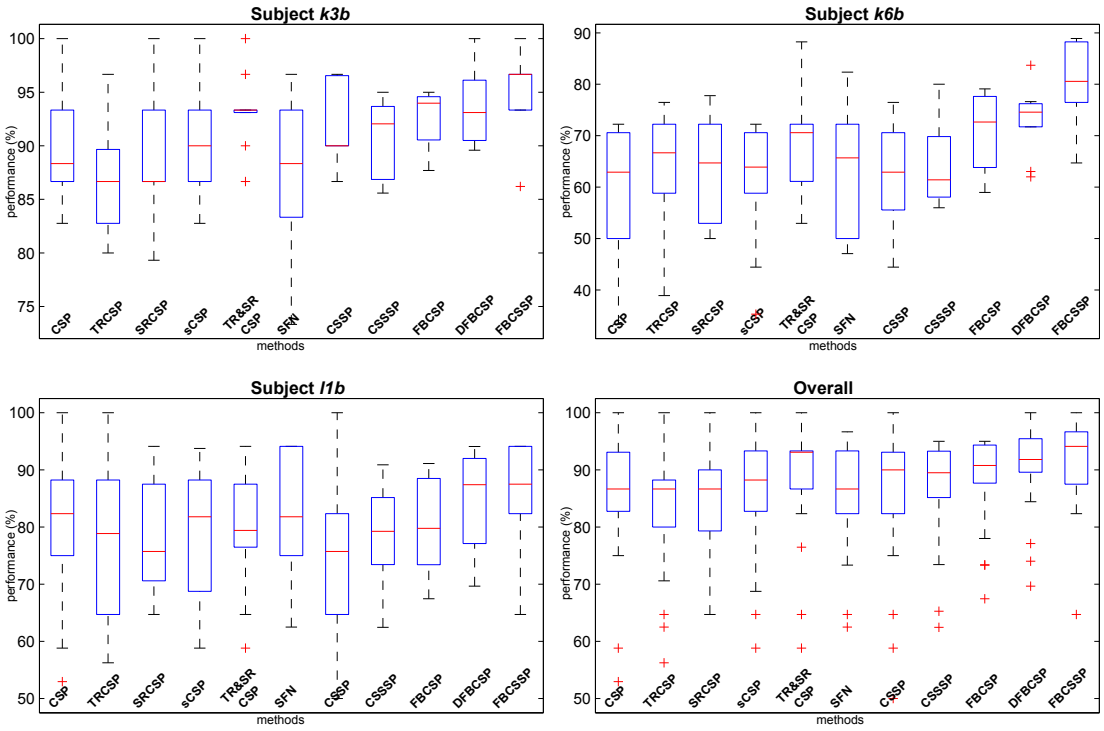

Figure 4.15 : Boxplots displaying the disturbance of classificaion performances for the subjects in dataset IIIA.

positions of EEG electrodes. Note that, the left hemisphere of brain controls the right side of the body and right hemisphere controls the left side.

On each figure below, spatial filters of one subject is displayed. Figure 4.17 to Figure 4.21 display the spatial filters from the subjects *aa* to *ay* and Figure 4.22 to Figure 4.24 display the spatial filters from the subjects *k3b* to *l1b*.

Figures map the spatial filters amplitudes to the EEG electrodes with color coding where white color means lower amplitude and black color means higher amplitude. A spatial filter coefficient with higher amplitude implies an activation related with the given motor imagery task at the location of the corresponding EEG channel. Since the activation areas of motor cortex is common to all people with small variations, obtained spatial filter visualizations should match the physiological motor cortex mapping.

Looking at the figures, we may observe that, spatial filters generally focused on the areas suitable with motor cortex mapping. Especially the spatial filters of TR&SR-CSP method clearly localized on the affiliated electrodes. The reason for that, this method steers the spatial filters to the given electrode areas by the given penalty matrix. However, the effect of th penalty matrix which is set by the α parameter, generally set to minimum which may be seen on the performance evaluation tables. Also, FBCSSP

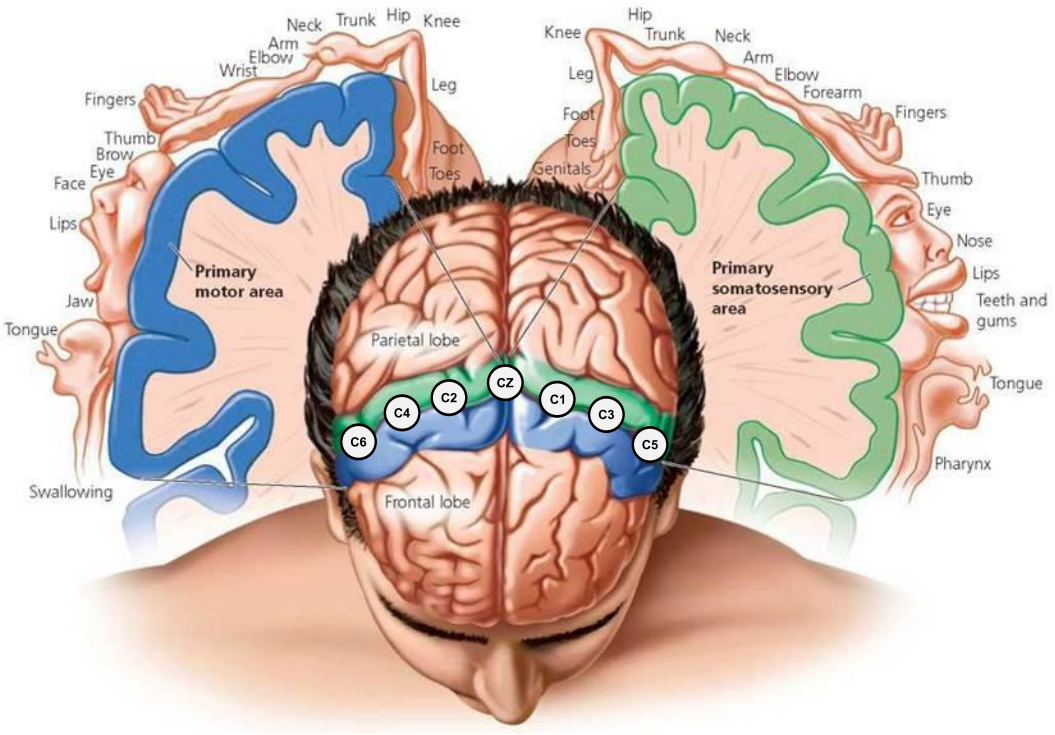

Figure 4.16 : Locations of primary motor and somatosensory areas over the brain.

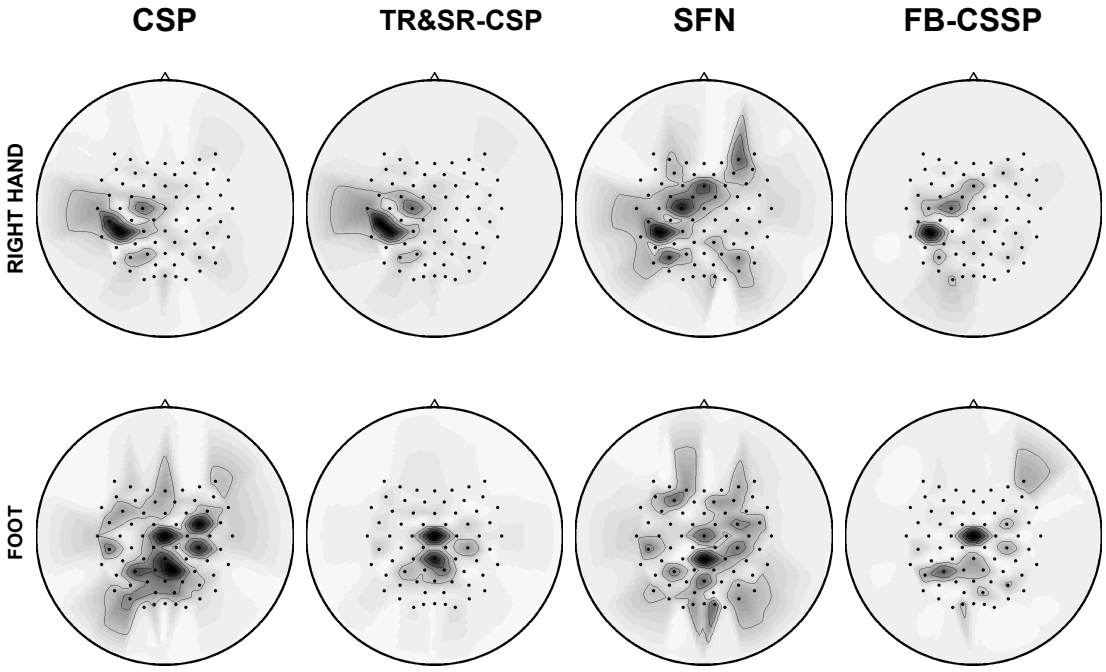

Figure 4.17 : Obtained spatial filters for subject *aa*.

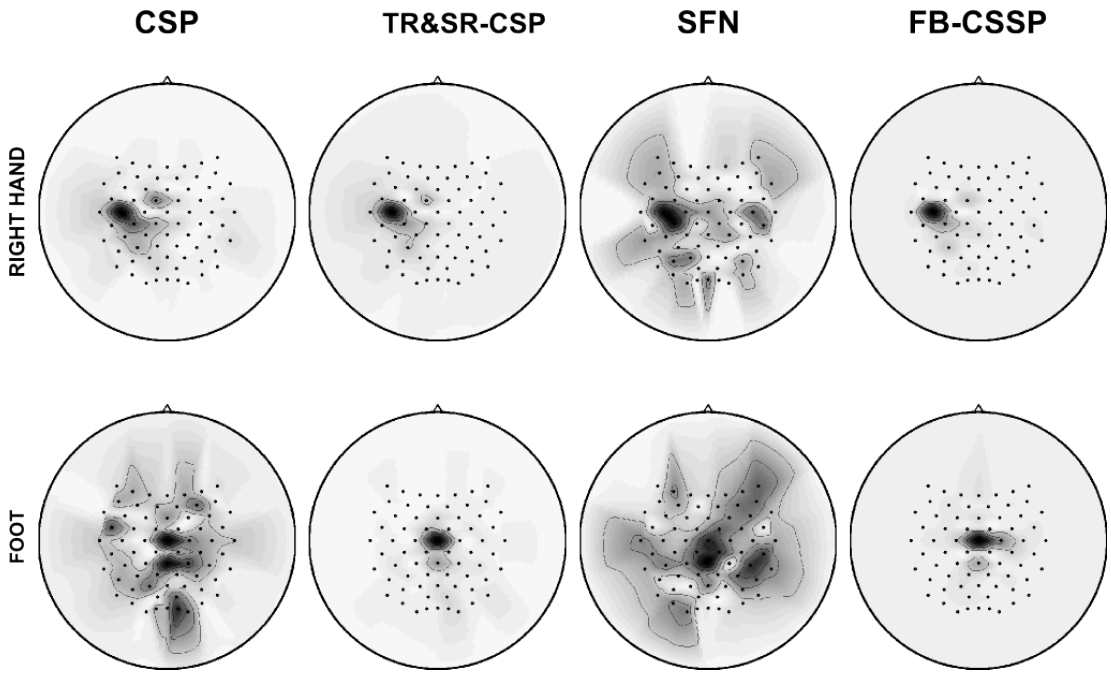

Figure 4.18 : Obtained spatial filters for subject *al*.

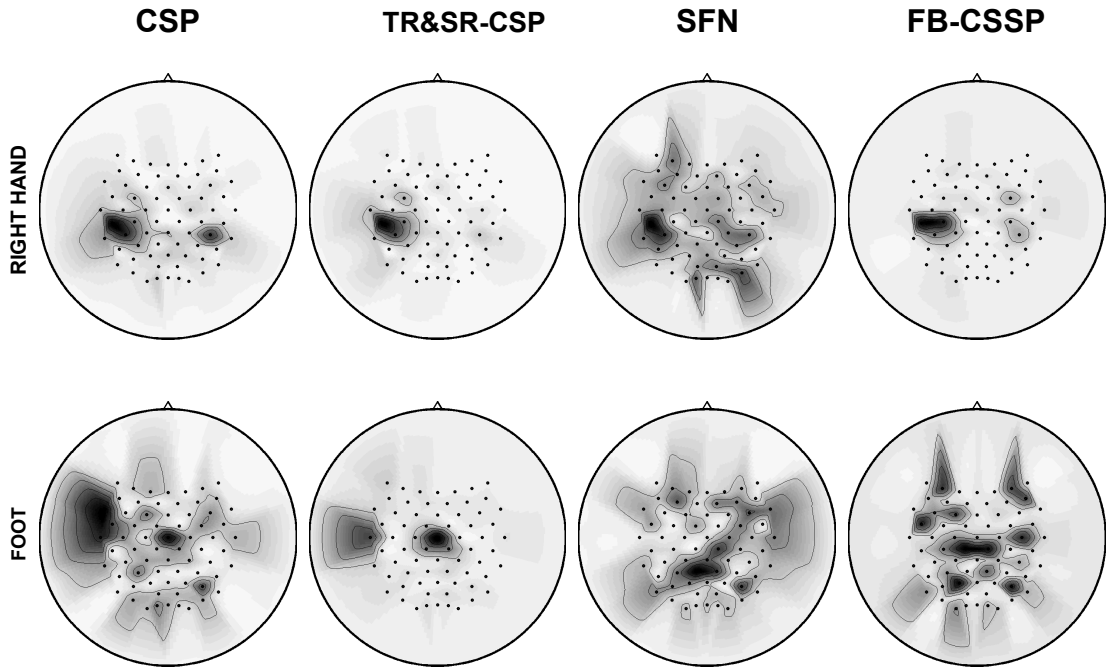

Figure 4.19 : Obtained spatial filters for subject *av*.

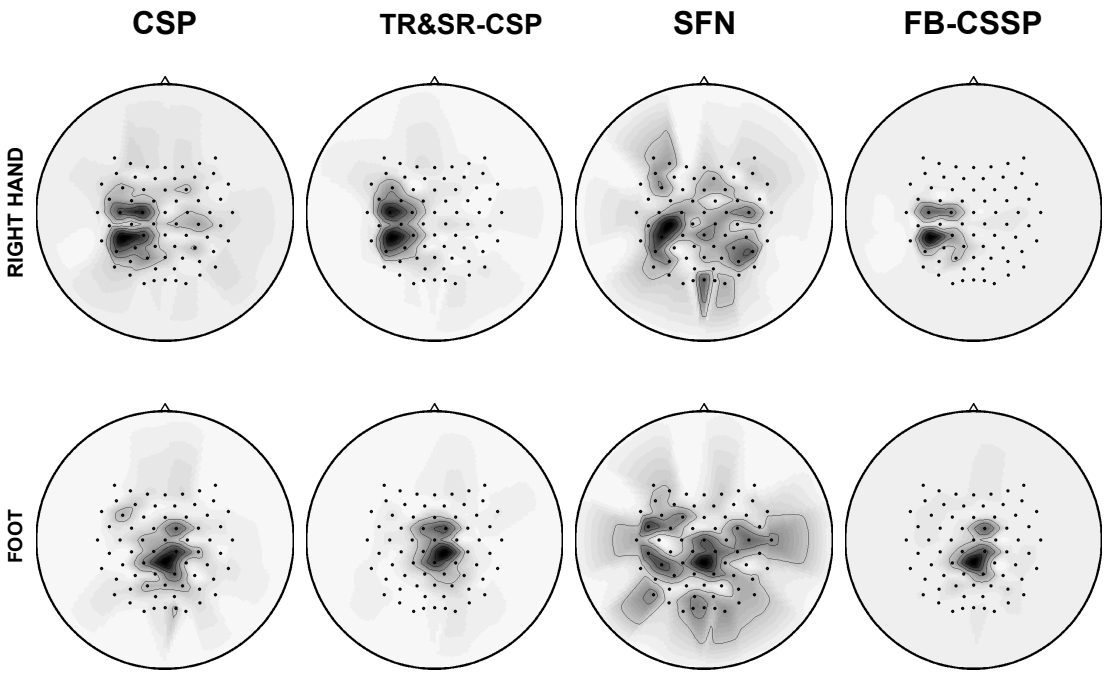

Figure 4.20 : Obtained spatial filters for subject *aw*.

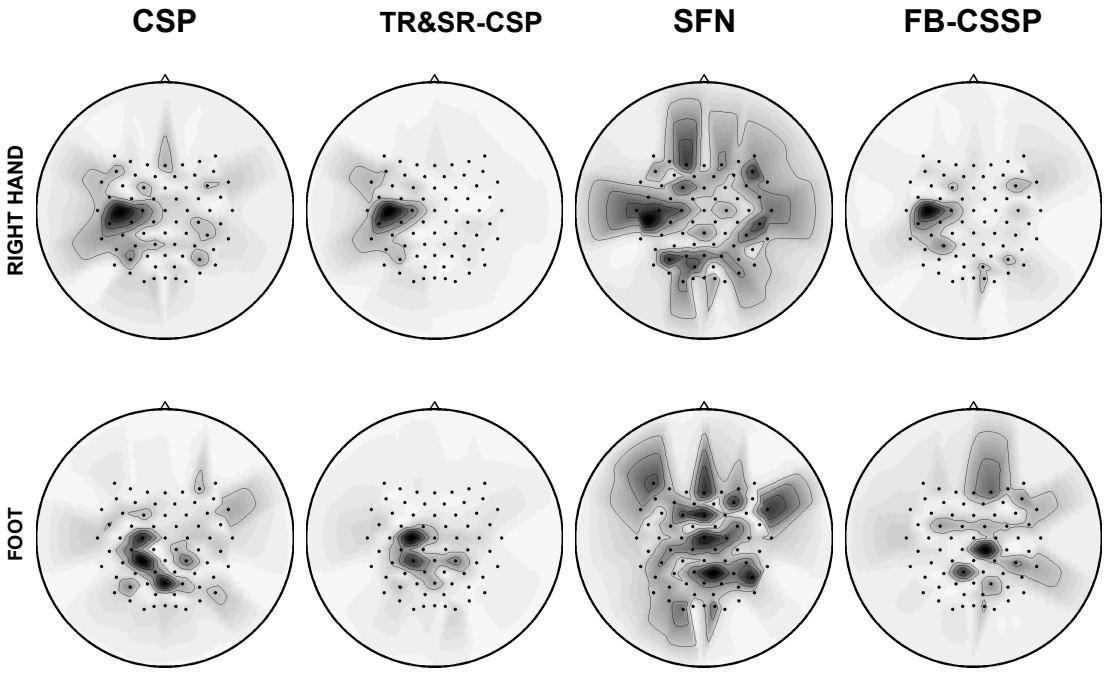

Figure 4.21 : Obtained spatial filters for subject *ay*.

method's spatial filters are successful at finding the true locations even if there is no spatial filter guidance. Not being as successfull as FBCSSP and TR&SR-CSP, SFN method's spatial filters reflects the underlying cortical activation to a certain extent.

It may be noticed that, there is a relation between the average classification performance of a subject and the clearance of his/her spatial filter maps. For example, subject *al* and *k3b* are the subjects whose classification accuracies are the highest among other subjects in the same datasets, their spatial filter mapping are the most focused ones. In the same manner, the spatial filters of *av* and *k6b*, who are the subjects with the worst classification accuracies reflects a scattered or wrongly localized mappings. The reason of poor classification rates and scattered spatial filter mappings of these subjects may be because of their psychological situation keeping them from concentrating to the given motor imagery task or their misunderstanding of imaging a motor movement by thinking just a visual scenery of that movement rather than imaging it. However, no matter how good or bad, the correlation between attained results of quantitative performance evaluations and spatial filter mappings that the motor imagery phenomenon is real and the proposed methods are useful algorithms for uncovering the underlying motor cortex activation.

4.5.2 Spectral filters of the proposed method

FBCSSP algorithm optimizes the spatial and spectral characteristic of the filters at the same time so that it learns the given training data in both spatial and spectral domains. In this subsection, spectral characteristics of the trained FBCSSP filters will be inspected.

With the imagination of limb movement, attenuation of EEG signals happens in the specific region of the motor and somatosensory cortex due to loss of synchrony in μ and β bands, classically defined in the 12-16Hz and 18-24Hz respectively which is called event-related de-synchronization (ERD) [114]. According to this statement, spectral characteristics of the filters should have a passband at μ and β bands.

Figure 4.25 shows the spectral characteristics of the FBCSSP filters obtained from the subjects of the motor imagery datasets. Frequency response of the trained FBCSSP network is calculated by feeding all of the inputs with signal at a given frequency and

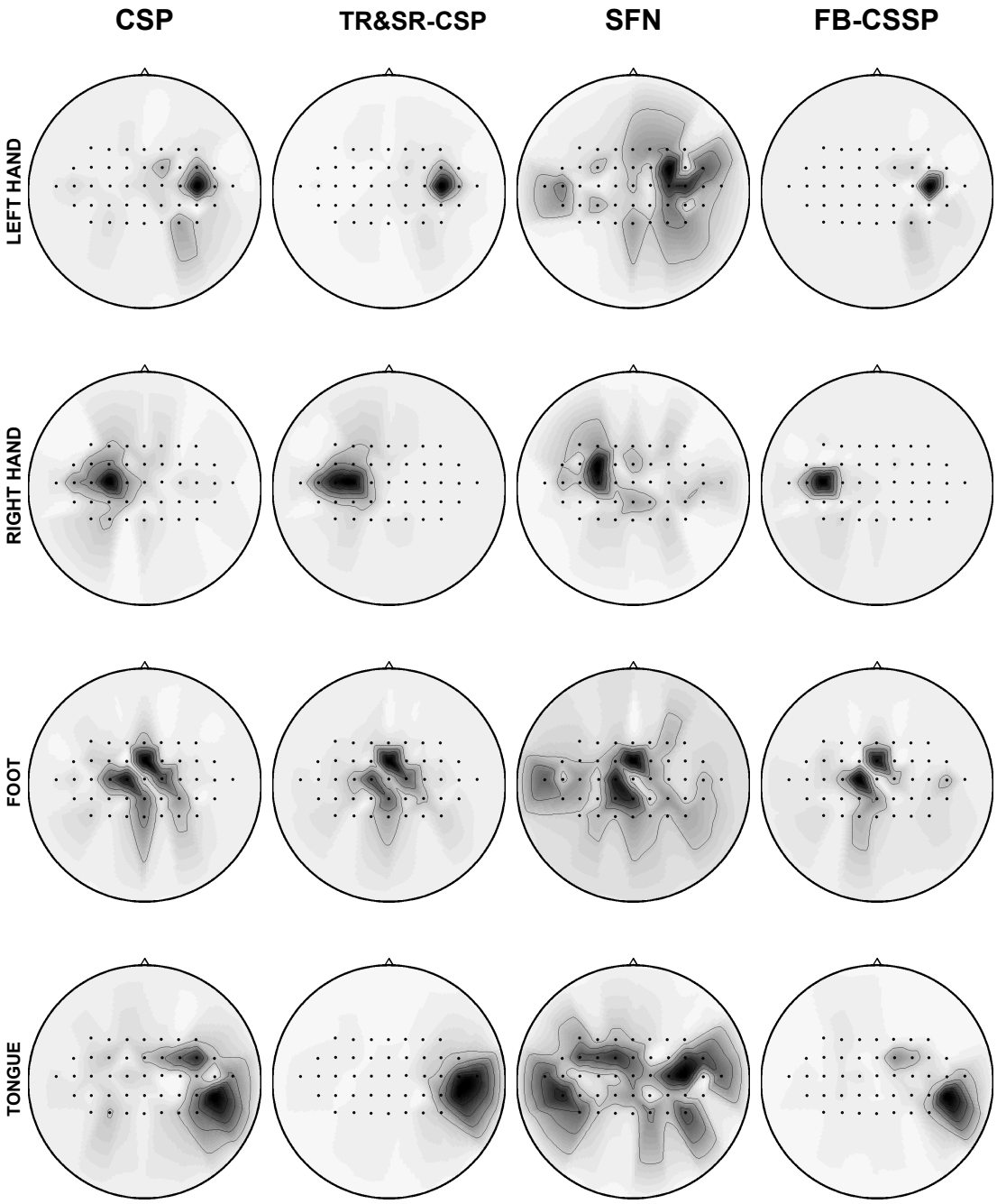

Figure 4.22 : Obtained spatial filters for subject *k3b*.

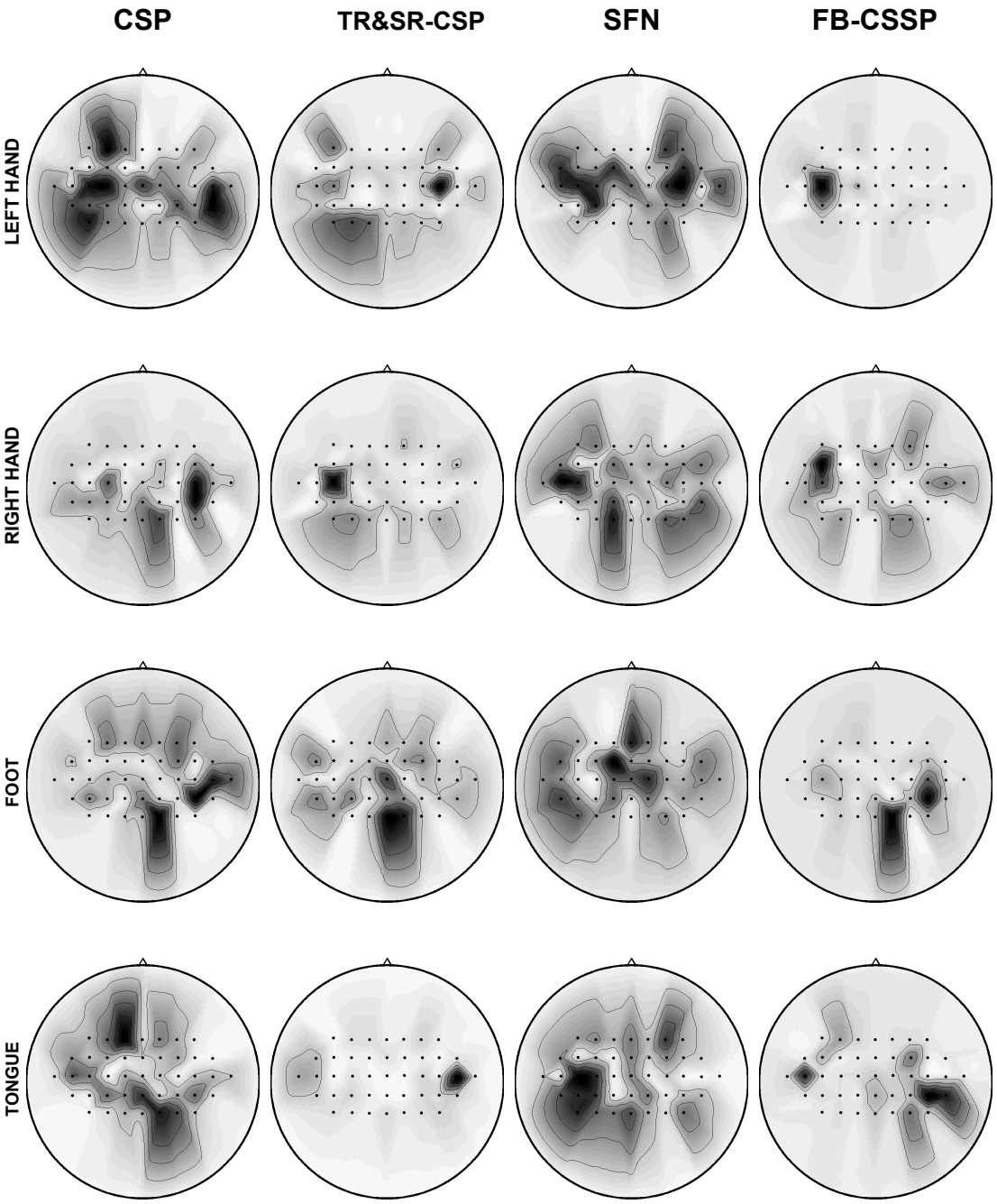

Figure 4.23 : Obtained spatial filters for subject *k6b*.

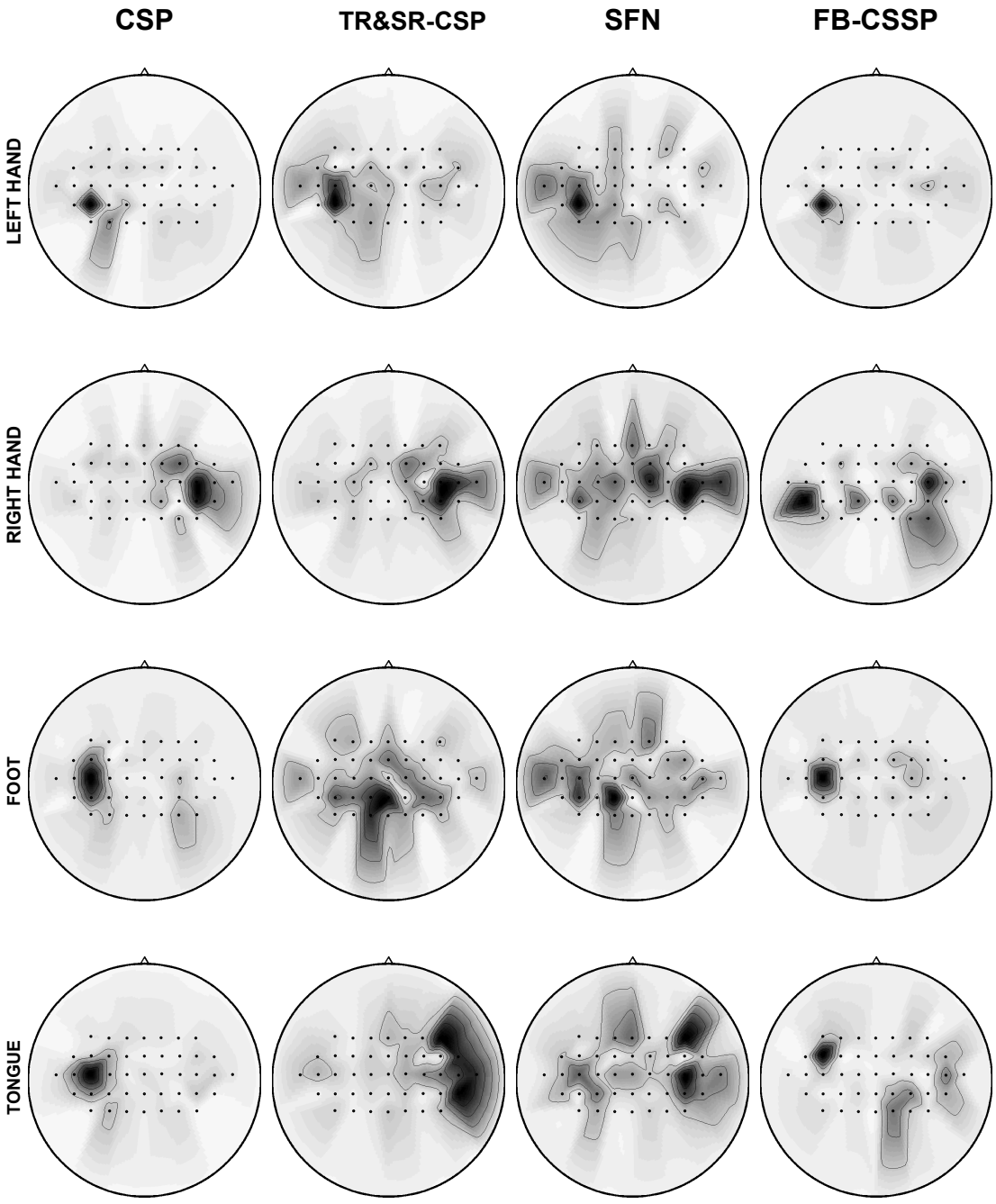

Figure 4.24 : Obtained spatial filters for subject *l1b*.

measuring the average power at the output. In the figures, the spectral filter response is normalized so that maximum obtained power was set to 1.

In the spectral filter plots, the pass band of the obtained spectral filters are located approximately within the band 8-16 Hz, which is associated with the sensorimotor cortex [203]. The filters reveal similar characteristics with minimal differences. As stated for the spatial filter, there is a correlation between the classification performance of the subject and suitability of the spectral filters to the given physiological facts, which is an evidence to the effectiveness of the FBCSSP method.

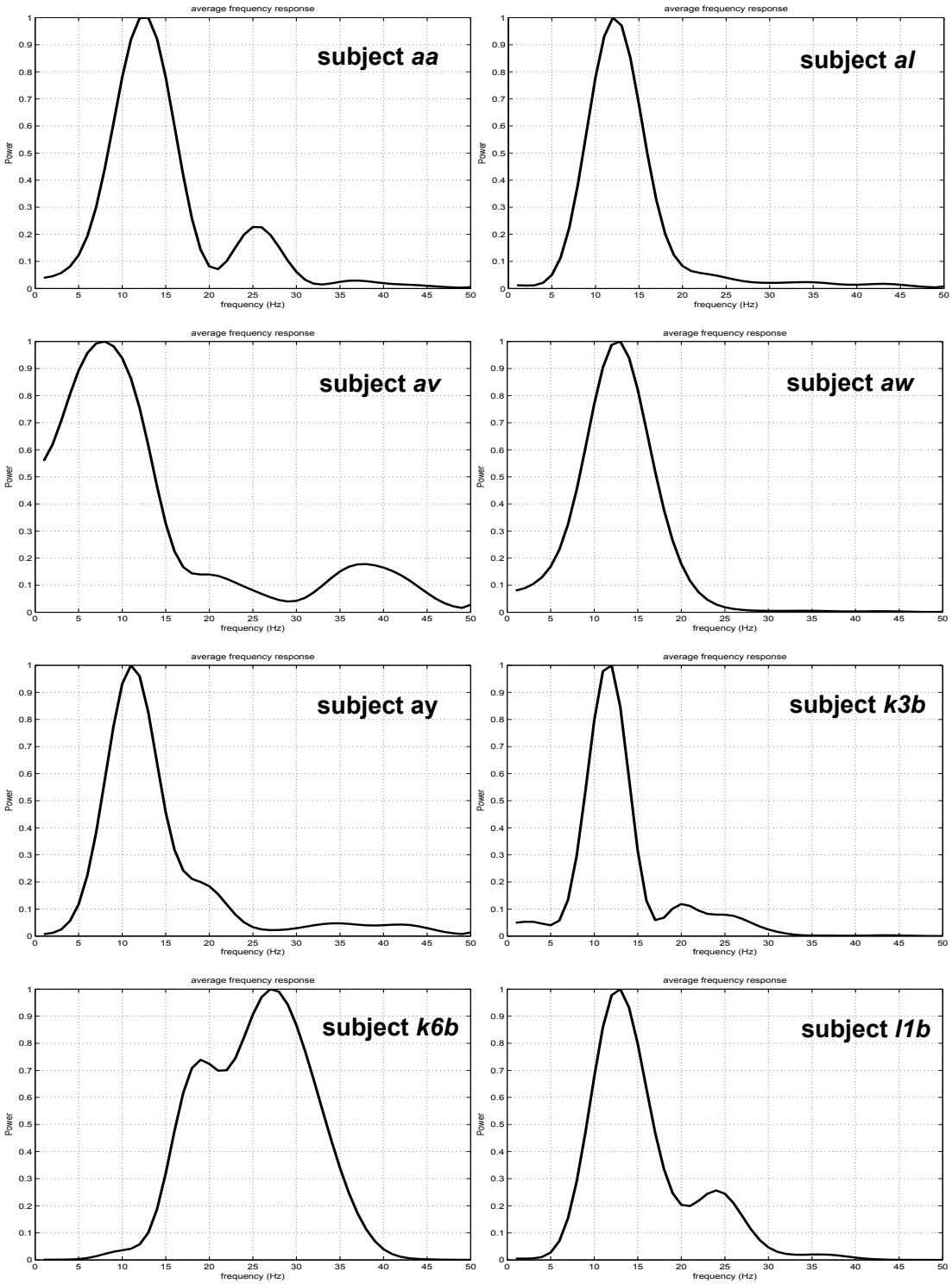

Figure 4.25 : Obtained spectral filters of the FBCSSP method for all subjects from the both datasets. First five figures (From *aa* to *ay*) are from dataset IVA and the rest (*k3b*,*k6b* and *l1b*) are from dataset IIIA.

5. CONCLUSION

BCI is an emerging technology applied on various areas from gaming equipment to health assistive devices. Among different types of BCI methodologies, MI is a popular paradigm which analyzes user's motor movement imagery with no physical activation of the limbs. The main reasons of its popularity are being noninvasive and classified as independent BCI which does not depend on brain's normal output channels like peripheral nerves in any way. Also, an MI user does not need to look at a screen like SSVEP or P300, which makes it a practical methodology. These factors caused the literature to focus on researching MI based BCI systems and develop MI signal classification methods.

Despite its popularity, MI signal classification is not a simple task. Low spatial resolution makes the classification harder because the signal acquired from the motor cortex is mixed up with the other signal sources of the brain. Also, being in the same frequency bands with the MI signals, much stronger alpha waves from the occipital area are observed on the motor cortex. Also, the subject to subject variability of spatial and spectral characteristics of the MI signals makes automated classification algorithms essential.

Spatial filtering methods are the pioneer methods for MI classification. Spatial filtering is simply linear combination of the EEG signals acquired from different EEG electrodes over the head. However, coefficients of linear combination should be set such that motor imagery related signals are amplified while the signals from other sources are weakened. CSP algorithm is the most popular spatial filtering method which calculates the spatial filters by analyzing the input EEG data. In the BCI competitions held from 2003 to 2005, CSP was proved to be a successful and effective algorithm.

Despite its popularity and effectiveness, CSP has some drawbacks. For example it is affected by the outliers in the dataset which mislead the spatial filters to focus on wrong electrodes. RCSP methods were the outcome of this drawback. A RCSP method

uses CSP's fitness formula with a simple penalty term addition. Each proposed RCSP method in the literature presented its individual way of penalty matrix calculation. The first contribution of this study, TR&SR-CSP method is a RCSP method presented in an international conference. Besides, CSP ignores within class scatterings which may be a drawback for EEG signal. Another contribution of this study called SFN method addresses this problem and solves spatial filtering problem with a neural network methodology. Its structure, forward and backward equations and training strategies for the SFN algorithm were described with details.

Spectral filtering is an important preprocessing step in MI classification. However, frequency characteristic of MI signal is subject specific. Manual tuning of the spectral filters specific to a subject is an exhaustive task. Therefore, generally a filter with a wide bandwidth is used prior to spatial filtering which is not very effective at eliminated unrelated frequency bands. Spatio-spectral filtering algorithms addresses this problem. They optimize spectral characteristics of the filters along with the spatial filters. As a result, spatio-spectral methods could achieve higher classification rates by automatically tuning both spatial and spectral filters. Proposed FBCSSP algorithm is a spatio-spectral filtering method which incorporates a FIR filter bank, a frequency specific spatial filter layer and a general spatial filter layer whose main job is frequency selection. FBCSSP method and similar spatio spectral methods were described with details in Chapter 3.

In the Computer Simulations chapter, motor imagery dataset used for evaluation was described. There were two different datasets from publicly available BCI competition datasets. The structure of the data, the experimental procedure and signal characteristics were given. Next, a demo application using the given dataset and extracting the ERD signal was presented for validating the given data. Then, EEG signal preprocessing steps for preparation to classification were described with selected configurations for selection of channels, frequency band or active time segment. Next, the methods that were used for evaluations were listed. The procedure for selection of method specific hyper-parameters was clarified. Finally, the performance evaluation results were reported with a table containing the classification accuracy for each subject of the dataset and for each method. Also, figures displaying classification accuracies for every single subject were given. In this chapter, not only the quantitative

performance values but also spatial and spectral characteristics of the proposed methods were inspected.

The results obtained from the proposed methods are promising. Their performance reports are competing with other important methods found in the literature. Especially FBCSSP method outperforms its components by means of classification accuracy and physiological plausibility of the obtained filters from this method are very satisfactory.

Future work may focus on automatic active time segment selection which was manually in this study. A three dimensional method which optimizes the frequency band, active time segment and spatial filters could increase the classification rates and generates better *spatio-spectra-temporal* filters. Also, increasing the number of classes is a challenging issue which may be realized with combination of different motor imagery tasks.

Real time motor imagery classification by utilizing the developed algorithms may be thought as a further target. However, real time classification of motor imagery is not a simple matter. Firstly, variability of EEG data over time for any subject requires online learning which brings computational load to the algorithms. Besides, developed real time algorithm should detect motor imagery on the fly without no cue information as trial based motor imagery classification. In the case of any voluntary motor imagery of the subject, real time detection block could generate triggers for motor imagery classifier and a specific task may be executed. Note that such preventing false positives is more important than false negatives in such a system since the subject may retry to generate a motor imagery signal although he or she has no control over false alarms. An example for such a real time motor imagery system may be a BCI mouse in which there are minimum 4 classes for each directions.

REFERENCES

[1] **Van Erp, J.B.**, **Lotte, F. and Tangermann, M.** (2012). Brain-computer interfaces: beyond medical applications, *Computer*, (4), 26–34.

[2] **Wolpaw, J.R.** *et al.* (2000). Brain-computer interface technology: a review of the first international meeting, *IEEE transactions on rehabilitation engineering*, *8*(2), 164–173.

[3] **Farwell, L.A. and Donchin, E.** (1988). Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials, *Electroencephalography and clinical Neurophysiology*, *70*(6), 510–523.

[4] **Pfurtscheller, G.** *et al.* (1997). EEG-based discrimination between imagination of right and left hand movement, *Electroencephalography and clinical Neurophysiology*, *103*(6), 642–651.

[5] **Gouy-Pailler, C.** *et al.* (2010). Nonstationary brain source separation for multiclass motor imagery, *Biomedical Engineering, IEEE Transactions on*, *57*(2), 469–478.

[6] **Müller-Putz, G.R. and Pfurtscheller, G.** (2008). Control of an electrical prosthesis with an SSVEP-based BCI, *Biomedical Engineering, IEEE Transactions on*, *55*(1), 361–364.

[7] **Wolpaw, J.R.** *et al.* (2002). Brain–computer interfaces for communication and control, *Clinical neurophysiology*, *113*(6), 767–791.

[8] **Ficke, R.C.** (1992). Digest of Data on Persons with Disabilities.

[9] **on Medical Rehabilitation Research, N.A.B.** (1992). Report and Research Plan for the National Center for Medical Rehabilitation Research, **Technical Report**, National Institute of Health.

[10] **Lopez, A.D.** *et al.* (1996). *The global burden of disease: a comprehensive assessment of mortality and disability from diseases, injuries, and risk factors in 1990 and projected to 2020*, Harvard School of Public Health.

[11] **Abresch, R.T.**, **Han, J.J. and Carter, G.T.** (2009). Rehabilitation management of neuromuscular disease: the role of exercise training, *Journal of clinical neuromuscular disease*, *11*(1), 7–21.

[12] **Santhosh, J.** *et al.* (2004). Quantitative EEG analysis for assessment to 'plan' a task in amyotrophic lateral sclerosis patients: a study of executive functions (planning) in ALS patients, *Cognitive brain research*, *22*(1), 59–66.

[13] **Madarame, T.** *et al.* (2008). The development of a brain computer interface device for amyotrophic lateral sclerosis patients, *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*, IEEE, pp.2401–2406.

[14] **Dornhege, G.**, (2007). Toward brain-computer interfacing, MIT press, pp.43–64.

[15] **Van Kokswijk, J. and Van Hulle, M.** (2010). Self adaptive BCI as service-oriented information system for patients with communication disabilities, *New Trends in Information Science and Service Science (NISS), 2010 4th International Conference on*, IEEE, pp.264–269.

[16] **Coyle, D.** *et al.* (2011). EEG-based continuous control of a game using a 3 channel motor imagery BCI: BCI game, *Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2011 IEEE Symposium on*, IEEE, pp.1–7.

[17] **Bos, D.P.O.** *et al.* (2010). Human-computer interaction for bci games: Usability and user experience, *Cyberworlds (CW), 2010 International Conference on*, IEEE, pp.277–281.

[18] **van de Laar, B.** *et al.* (2013). Experiencing BCI control in a popular computer game, *Computational Intelligence and AI in Games, IEEE Transactions on*, *5*(2), 176–184.

[19] **Chin, Z.Y.** *et al.* (2010). Online performance evaluation of motor imagery BCI with augmented-reality virtual hand feedback, *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, IEEE, pp.3341–3344.

[20] **Dornhege, G.**, (2007). Toward brain-computer interfacing, MIT press, pp.393–408.

[21] **Edlinger, G.** *et al.* (2009). Brain-computer interfaces for goal orientated control of a virtual smart home environment, *Neural Engineering, 2009. NER'09. 4th International IEEE/EMBS Conference on*, IEEE, pp.463–465.

[22] **Chae, Y.**, **Jeong, J. and Jo, S.** (2011). Noninvasive brain-computer interface-based control of humanoid navigation, *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, IEEE, pp.685–691.

[23] **Galán, F.** *et al.* (2008). A brain-actuated wheelchair: asynchronous and non-invasive brain–computer interfaces for continuous control of robots, *Clinical Neurophysiology*, *119*(9), 2159–2169.

[24] **Esfahani, E.T. and Sundararajan, V.** (2012). Classification of primitive shapes using brain–computer interfaces, *Computer-Aided Design*, *44*(10), 1011–1019.

[25] **Ahangi, A.** *et al.* (2013). Multiple classifier system for EEG signal classification with application to brain–computer interfaces, *Neural Computing and Applications*, *23*(5), 1319–1327.

[26] **Pfurtscheller, G. and Neuper, C.** (2001). Motor imagery and direct brain-computer communication, *Proceedings of the IEEE*, *89*(7), 1123–1134.

[27] **Sabra, N.I. and Abdel Wahed, M.** (2011). The use of MEG-based brain computer interface for classification of wrist movements in four different directions, *Radio Science Conference (NRSC), 2011 28th National*, IEEE, pp.1–7.

[28] **Zhang, J.** *et al.* (2011). Clustering linear discriminant analysis for MEG-based brain computer interfaces, *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, *19*(3), 221–231.

[29] **Sitaram, R.** *et al.* (2008). fMRI brain-computer interfaces, *Signal Processing Magazine, IEEE*, *25*(1), 95–106.

[30] **Andersson, P.** *et al.* (2009). fMRI based BCI control using spatial visual attention at 7T, *Neural Engineering, 2009. NER'09. 4th International IEEE/EMBS Conference on*, IEEE, pp.444–446.

[31] **Ayaz, H.** *et al.* (2011). An optical brain computer interface for environmental control, *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, IEEE, pp.6327–6330.

[32] **Sitaram, R.** *et al.* (2007). Temporal classification of multichannel near-infrared spectroscopy signals of motor imagery for developing a brain–computer interface, *NeuroImage*, *34*(4), 1416–1427.

[33] **Lee, P.L.** *et al.* (2012). A brain-wave-actuated small robot car using ensemble empirical mode decomposition-based approach, *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, *42*(5), 1053–1064.

[34] **Li, J. and Zhang, L.** (2012). Active training paradigm for motor imagery BCI, *Experimental brain research*, *219*(2), 245–254.

[35] **Ge, S.**, **Wang, R. and Yu, D.** (2014). Classification of four-class motor imagery employing single-channel electroencephalography, *PloS one*, *9*(6), e98019.

[36] **Decety, J. and Ingvar, D.H.** (1990). Brain structures participating in mental simulation of motor behavior: a neuropsychological interpretation, *Acta Psychologica*, *73*(1), 13–34.

[37] **Wang, Y.** *et al.* (2007). Design of electrode layout for motor imagery based brain-computer interface, *Electronics Letters*, *43*(10), 1.

[38] **Sutter, E.E.** (1992). The brain response interface: communication through visually-induced electrical brain responses, *Journal of Microcomputer Applications*, *15*(1), 31–45.

[39] **Sutter, E.E. and Tran, D.** (1992). The field topography of ERG components in man—I. The photopic luminance response, *Vision research*, *32*(3), 433–446.

[40] **Cheng, M.** *et al.* (2002). Design and implementation of a brain-computer interface with high transfer rates, *Biomedical Engineering, IEEE Transactions on*, *49*(10), 1181–1186.

[41] **İşcan, Z.**, **Özkaya, Ö. and Dokur, Z.**, (2011). Classification of EEG in a steady state visual evoked potential based brain computer interface experiment, Adaptive and Natural Computing Algorithms, Springer, pp.81–88.

[42] **Vialatte, F.B.** *et al.* (2010). Steady-state visually evoked potentials: focus on essential paradigms and future perspectives, *Progress in neurobiology*, *90*(4), 418–438.

[43] **Guo, J.**, **Gao, S. and Hong, B.** (2010). An auditory brain–computer interface using active mental response, *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, *18*(3), 230–235.

[44] **Matsumoto, Y.** *et al.* (2012). Auditory steady-state response stimuli based BCI application-the optimization of the stimuli types and lengths, *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, IEEE, pp.1–7.

[45] **Muller-Putz, G.R.** *et al.* (2006). Steady-state somatosensory evoked potentials: suitable brain signals for brain-computer interfaces?, *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, *14*(1), 30–37.

[46] **Ahn, M.**, **Hong, J.H. and Jun, S.C.** (2012). Feasibility of approaches combining sensor and source features in brain–computer interface, *Journal of neuroscience methods*, *204*(1), 168–178.

[47] **Lee, P.L.** *et al.* (2011). An SSVEP-based BCI using high duty-cycle visual flicker, *Biomedical Engineering, IEEE Transactions on*, *58*(12), 3350–3359.

[48] **Calhoun, G.** *et al.* (1995). Control of functional electrical stimulation with a direct brain interface, *Proc. RESNA'95*, 696–698.

[49] **Lalor, E.** *et al.* (2004). Brain computer interface based on the steady-state VEP for immersive gaming control, *Biomed. Tech*, *49*(1), 63–64.

[50] **Teder-Sälejärvi, W.A.** *et al.* (1999). Intra-modal and cross-modal spatial attention to auditory and visual stimuli. An event-related brain potential study, *Cognitive Brain Research*, *8*(3), 327–343.

[51] **Walter, W.** *et al.* (1964). Contingent negative variation: an electric sign of sensori-motor association and expectancy in the human brain, *Nature*, *203*, 380–384.

[52] **Sutton, S.** *et al.* (1965). Evoked-potential correlates of stimulus uncertainty, *Science*, *150*(3700), 1187–1188.

[53] **Donchin, E. and Smith, D.** (1970). The contingent negative variation and the late positive wave of the average evoked potential, *Electroencephalography and clinical Neurophysiology*, *29*(2), 201–203.

[54] **Bernat, E.**, **Shevrin, H. and Snodgrass, M.** (2001). Subliminal visual oddball stimuli evoke a P300 component, *Clinical neurophysiology*, *112*(1), 159–171.

[55] **Gonsalvez, C.J. and Polich, J.** (2002). P300 amplitude is determined by target-to-target interval, *Psychophysiology*, *39*(3), 388–396.

[56] **Fazel-Rezai, R.** *et al.* (2012). P300 brain computer interface: current challenges and emerging trends, *Frontiers in Neuroengineering*, *5*(14), 14.

[57] **Zhao, H.b.** *et al.* (2009). Brain-computer interface design based on slow cortical potentials using matlab/simulink, *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*, IEEE, pp.1044–1048.

[58] **Elbert, T.** *et al.* (1980). Biofeedback of slow cortical potentials. I, *Electroencephalography and Clinical Neurophysiology*, *48*(3), 293–301.

[59] **Birbaumer, N.** *et al.* (1999). A spelling device for the paralysed, *Nature*, *398*(6725), 297–298.

[60] **Hinterberger, T.**, **Mellinger, J. and Birbaumer, N.** (2003). The thought translation device: Structure of a multimodal brain-computer communication system, *Neural Engineering, 2003. Conference Proceedings. First International IEEE EMBS Conference on*, IEEE, pp.603–606.

[61] **Hinterberger, T.** *et al.* (2004). Brain-computer communication and slow cortical potentials, *Biomedical Engineering, IEEE Transactions on*, *51*(6), 1011–1018.

[62] **Kübler, A.** (2000). *Brain Computer Communication: Development of a Brain Computer Interface for Locked-in Patients on the Basis of the Psychophysiological Self-regulation Training of Slow Cortical Potentials (SCP)*, Schwäbische Verlags-Gesellschaft.

[63] **Mensh, B.D.**, **Werfel, J. and Seung, H.S.** (2004). BCI competition 2003-data set Ia: combining gamma-band power with slow cortical potentials to improve single-trial classification of electroencephalographic signals, *Biomedical Engineering, IEEE Transactions on*, *51*(6), 1052–1056.

[64] **Fetz, E. and Finocchio, D.** (1975). Correlations between activity of motor cortex cells and arm muscles during operantly conditioned response patterns, *Experimental Brain Research*, *23*(3), 217–240.

[65] **Wyler, A.R. and Burchiel, K.J.** (1978). Factors influencing accuracy of operant control of pyramidal tract neurons in monkey, *Brain research*, *152*(2), 418–421.

[66] **Wyler, A.R.**, **Burchiel, K.J. and Robbins, C.A.** (1979). Operant control of precentral neurons in monkeys: evidence against open loop control, *Brain Research*, *171*(1), 29–39.

[67] **Schmidt, E.M.** (1980). Single neuron recording from motor cortex as a possible source of signals for control of external devices, *Annals of biomedical engineering*, *8*(4-6), 339–349.

[68] **Kennedy, P.R. and Bakay, R.A.** (1998). Restoration of neural output from a paralyzed patient by a direct brain connection, *Neuroreport*, *9*(8), 1707–1711.

[69] **Kennedy, P.R.** *et al.* (2000). Direct control of a computer from the human central nervous system, *Rehabilitation Engineering, IEEE Transactions on*, *8*(2), 198–202.

[70] **Khorshidtalab, A.**, **Salami, M.J.E. and Hamedi, M.** (2012). Evaluating the effectiveness of time-domain features for motor imagery movements using SVM, *Computer and Communication Engineering (ICCCE), 2012 International Conference on*, IEEE, pp.909–913.

[71] **Penfield, W.** (1950). The supplementary motor area in the cerebral cortex of man, *European Archives of Psychiatry and Clinical Neuroscience*, *185*(6), 670–674.

[72] **Mitz, A.R. and Wise, S.P.** (1987). The somatotopic organization of the supplementary motor area: intracortical microstimulation mapping, *The Journal of neuroscience*, *7*(4), 1010–1021.

[73] **Roland, P.E.** *et al.* (1980). Supplementary motor area and other cortical areas in organization of voluntary movements in man, *Journal of neurophysiology*, *43*(1), 118–136.

[74] **Halsband, U.**, **Matsuzaka, Y. and Tanji, J.** (1994). Neuronal activity in the primate supplementary, pre-supplementary and premotor cortex during externally and internally instructed sequential movements, *Neuroscience research*, *20*(2), 149–155.

[75] **Brinkman, C.** (1981). Lesions in supplementary motor area interfere with a monkey's performance of a bimanual coordination task, *Neuroscience letters*, *27*(3), 267–270.

[76] **He, S.Q.**, **Dum, R.P. and Strick, P.L.** (1993). Topographic organization of corticospinal projections from the frontal lobe: motor areas on the lateral surface of the hemisphere, *The Journal of neuroscience*, *13*(3), 952–980.

[77] **Picard, N. and Strick, P.L.** (2003). Activation of the supplementary motor area (SMA) during performance of visually guided movements, *Cerebral Cortex*, *13*(9), 977–986.

[78] **Graziano, M.** (2008). *The intelligent movement machine: an ethological perspective on the primate motor system*, Oxford University Press.

[79] **Graziano, M.S.**, **Aflalo, T.N. and Cooke, D.F.** (2005). Arm movements evoked by electrical stimulation in the motor cortex of monkeys, *Journal of Neurophysiology*, *94*(6), 4209–4223.

[80] **Penfield, W. and Boldrey, E.** (1937). Somatic motor and sensory representation in the cerebral cortex of man as studied by electrical stimulation., *Brain: A journal of neurology*.

[81] **Sage, G.** (1971). *Introduction to motor behavior: a neuropsychological approach*, Addison-Wesley series in physical education, Addison-Wesley Pub. Co., http://books.google.com.tr/books?id=F0VqAAAAMAAJ.

[82] **Pfurtscheller, G.** (1977). Graphical display and statistical evaluation of event-related desynchronization (ERD), *Electroencephalography and clinical neurophysiology*, *43*(5), 757–760.

[83] **Pfurtscheller, G. and Aranibar, A.** (1977). Event-related cortical desynchronization detected by power measurements of scalp EEG, *Electroencephalography and clinical neurophysiology*, *42*(6), 817–826.

[84] **Pfurtscheller, G. and Klimesch, W.** (1992). Functional topography during a visuoverbal judgment task studied with event-related desynchronization mapping., *Journal of Clinical Neurophysiology*, *9*(1), 120–131.

[85] **Jeannerod, M.** (1995). Mental imagery in the motor context, *Neuropsychologia*, *33*(11), 1419–1432.

[86] **Decety, J.** *et al.* (1994). Mapping motor representations with positron emission tomography.

[87] **Malouin, F.** *et al.* (2003). Brain activations during motor imagery of locomotor-related tasks: A PET study, *Human brain mapping*, *19*(1), 47–62.

[88] **Lacourse, M.G.** *et al.* (2005). Brain activation during execution and motor imagery of novel and skilled sequential hand movements, *Neuroimage*, *27*(3), 505–519.

[89] **Arvaneh, M.** *et al.* (2011). Optimizing the channel selection and classification accuracy in EEG-based BCI, *Biomedical Engineering, IEEE Transactions on*, *58*(6), 1865–1873.

[90] **Garg, A. and Binderup, A.** (2007). Implementation of Online Brain Computer Interface Using Motor Imagery In LabVIEW.

[91] **Lou, B.** *et al.* (2008). Bipolar electrode selection for a motor imagery based brain? computer interface, *Journal of Neural Engineering*, *5*(3), 342.

[92] **Tam, W.K.** *et al.* (2011). A minimal set of electrodes for motor imagery BCI to control an assistive device in chronic stroke subjects: a multi-session study, *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, *19*(6), 617–627.

[93] **Wolters, C. and Munck, J.C.d.** (2007). Volume conduction, *Scholarpedia*, *2*(3), 1738.

[94] **Kam, T.E.**, **Suk, H.I. and Lee, S.W.** (2013). Non-homogeneous spatial filter optimization for ElectroEncephaloGram (EEG)-based motor imagery classification, *Neurocomputing*, *108*, 58–68.

[95] **Vučković, A. and Sepulveda, F.** (2012). A two-stage four-class BCI based on imaginary movements of the left and the right wrist, *Medical engineering & physics*, *34*(7), 964–971.

[96] **Thomas, K.P.** *et al.* (2011). Adaptive tracking of discriminative frequency components in electroencephalograms for a robust brain–computer interface, *Journal of neural engineering*, *8*(3), 036007.

[97] **Shin, Y.** *et al.* (2012). Sparse representation-based classification scheme for motor imagery-based brain–computer interface systems, *Journal of neural engineering*, *9*(5), 056002.

[98] **Devlaminck, D.** *et al.* (2011). Multisubject learning for common spatial patterns in motor-imagery BCI, *Computational intelligence and neuroscience*, *2011*, 8.

[99] **Naeem, M.**, **Brunner, C. and Pfurtscheller, G.** (2009). Dimensionality reduction and channel selection of motor imagery electroencephalographic data, *Computational intelligence and neuroscience*, *2009*.

[100] **Hwang, H.J.**, **Kwon, K. and Im, C.H.** (2009). Neurofeedback-based motor imagery training for brain–computer interface (BCI), *Journal of neuroscience methods*, *179*(1), 150–156.

[101] **Vuckovic, A. and Sepulveda, F.** (2008). Delta band contribution in cue based single trial classification of real and imaginary wrist movements, *Medical & biological engineering & computing*, *46*(6), 529–539.

[102] **Samek, W.** *et al.* (2012). Stationary common spatial patterns for brain computer interfacing, *Journal of neural engineering*, *9*(2), 026013.

[103] **Zhou, S.M.**, **Gan, J.Q. and Sepulveda, F.** (2008). Classifying mental tasks based on features of higher-order statistics from EEG signals in brain–computer interface, *Information Sciences*, *178*(6), 1629–1640.

[104] **Lee, S. and Lim, H.S.**, (2011). Predicting text entry for brain-computer interface, Future Information Technology, Springer, pp.309–312.

[105] **Wang, Y.**, **Gao, S. and Gao, X.** (2006). Common spatial pattern method for channel selelction in motor imagery based brain-computer interface, *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, IEEE, pp.5392–5395.

[106] **Sharbrough, F.** *et al.* (1991). American electroencephalographic society guidelines for standard electrode position nomenclature, *J. clin. Neurophysiol*, *8*(2), 200–202.

[107] **Malmivuo, J. and Plonsey, R.** (1995). *Bioelectromagnetism: principles and applications of bioelectric and biomagnetic fields*, Oxford University Press, USA.

[108] **Amzica, F. and Lopes da Silva, F.** (2011). Cellular substrates of brain rhythms, *Niedermeyer's Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*, 33–63.

[109] **Kim, J.Y.** *et al.*, (2013). Optimal EEG channel selection for motor imagery BCI system using BPSO and GA, Robot Intelligence Technology and Applications 2012, Springer, pp.231–239.

[110] **He, L.** *et al.* (2013). Channel selection by Rayleigh coefficient maximization based genetic algorithm for classifying single-trial motor imagery EEG, *Neurocomputing*, *121*, 423–433.

[111] **Ramoser, H.**, **Muller-Gerking, J. and Pfurtscheller, G.** (2000). Optimal spatial filtering of single trial EEG during imagined hand movement, *Rehabilitation Engineering, IEEE Transactions on*, *8*(4), 441–446.

[112] **Müller-Gerking, J.**, **Pfurtscheller, G. and Flyvbjerg, H.** (1999). Designing optimal spatial filters for single-trial EEG classification in a movement task, *Clinical neurophysiology*, *110*(5), 787–798.

[113] **Novi, Q.** *et al.* (2007). Sub-band common spatial pattern (SBCSP) for brain-computer interface, *Neural Engineering, 2007. CNE'07. 3rd International IEEE/EMBS Conference on*, IEEE, pp.204–207.

[114] **Pfurtscheller, G. and Da Silva, F.L.** (1999). Event-related EEG/MEG synchronization and desynchronization: basic principles, *Clinical neurophysiology*, *110*(11), 1842–1857.

[115] **Toro, C.** *et al.* (1994). Event-related desynchronization and movement-related cortical potentials on the ECoG and EEG, *Electroencephalography and Clinical Neurophysiology/Evoked Potentials Section*, *93*(5), 380–389.

[116] **Naeem, M.** *et al.* (2006). Seperability of four-class motor imagery data using independent components analysis, *Journal of neural engineering*, *3*(3), 208.

[117] **Hsu, W.Y.** (2012). Enhanced active segment selection for single-trial EEG classification, *Clinical EEG and neuroscience*, *43*(2), 87–96.

[118] **Blankertz, B.** *et al.* (2008). Optimizing spatial filters for robust EEG single-trial analysis, *Signal Processing Magazine, IEEE*, *25*(1), 41–56.

[119] psychophysiology blog, `http://psychophysiology.blogspot.com.tr/2007_11_01_archive.html`.

[120] **Dornhege, G.** *et al.* (2006). Combined optimization of spatial and temporal filters for improving brain-computer interfacing, *Biomedical Engineering, IEEE Transactions on*, *53*(11), 2274–2281.

[121] **Blankertz, B.** *et al.* (2007). Invariant common spatial patterns: Alleviating nonstationarities in brain-computer interfacing, *Advances in neural information processing systems*, pp.113–120.

[122] **Schalk, G. and Mellinger, J.** (2010). *A Practical Guide to Brain–Computer Interfacing with BCI2000: General-Purpose Software for Brain-Computer Interface Research, Data Acquisition, Stimulus Presentation, and Brain Monitoring*, Springer Science & Business Media.

[123] **Le, J. and Gevins, A.** (1993). Method to reduce blur distortion from EEG's using a realistic head model, *Biomedical Engineering, IEEE Transactions on*, *40*(6), 517–528.

[124] **McFarland, D.J.** *et al.* (1997). Spatial filter selection for EEG-based communication, *Electroencephalography and clinical Neurophysiology*, *103*(3), 386–394.

[125] **Nunez, P.** *et al.* (1994). A theoretical and experimental study of high resolution EEG based on surface Laplacians and cortical imaging, *Electroencephalography and clinical Neurophysiology*, *90*(1), 40–57.

[126] **Makeig, S.** *et al.* (1996). Independent component analysis of electroencephalographic data, *Advances in neural information processing systems*, 145–151.

[127] **Zhou, B.** *et al.* (2014). Robust Spatial Filters on Three-Class Motor Imagery EEG Data Using Independent Component Analysis, *Journal of Biosciences and Medicines*, *2*(02), 43.

[128] **Blankertz, B.** *et al.* (2004). The BCI competition 2003: progress and perspectives in detection and discrimination of EEG single trials, *Biomedical Engineering, IEEE Transactions on*, *51*(6), 1044–1051.

[129] **Blankertz, B.** *et al.* (2006). The BCI competition III: Validating alternative approaches to actual BCI problems, *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, *14*(2), 153–159.

[130] **Fukunaga, K. and Koontz, W.L.** (1970). Application of the Karhunen-Lo? ve Expansion to Feature Selection and Ordering, *IEEE Transactions on Computers*, (4), 311–318.

[131] **Wang, H.**, **Tang, Q. and Zheng, W.** (2012). L1-norm-based common spatial patterns, *Biomedical Engineering, IEEE Transactions on*, *59*(3), 653–662.

[132] **Falzon, O.**, **Camilleri, K.P. and Muscat, J.** (2012). The analytic common spatial patterns method for EEG-based BCI data, *Journal of Neural Engineering*, *9*(4), 045009.

[133] **Reuderink, B. and Poel, M.** (2008). Robustness of the common spatial patterns algorithm in the BCI-pipeline, **Technical ReportTR-CTIT-08-52**.

[134] **Lotte, F. and Guan, C.** (2011). Regularizing common spatial patterns to improve BCI designs: unified theory and new algorithms, *Biomedical Engineering, IEEE Transactions on*, *58*(2), 355–362.

[135] **Fattahi, D.**, **Nasihatkon, B. and Boostani, R.** (2013). A general framework to estimate spatial and spatio-spectral filters for EEG signal classification, *Neurocomputing*, *119*, 165–174.

[136] **Yüksel, A. and Ölmez, T.** (2014). Task Related and Spatially Regularized Common Spatial Patterns for Brain Computer Interfaces., *IWBBIO*, pp.42–53.

[137] **Yuksel, A. and Olmez, T.** (2015). A Neural Network-Based Optimal Spatial Filter Design Method for Motor Imagery Classification, *PloS one*, *10*(5), e0125039.

[138] **Lemm, S.** *et al.* (2005). Spatio-spectral filters for improving the classification of single trial EEG, *Biomedical Engineering, IEEE Transactions on*, *52*(9), 1541–1548.

[139] **Tomioka, R.** *et al.* (2006). Spectrally weighted common spatial pattern algorithm for single trial EEG classification, *Dept. Math. Eng., Univ. Tokyo, Tokyo, Japan, Tech. Rep*, *40*.

[140] **Higashi, H. and Tanaka, T.** (2013). Simultaneous design of FIR filter banks and spatial patterns for EEG signal classification, *Biomedical Engineering, IEEE Transactions on*, *60*(4), 1100–1110.

[141] **Fisher, R.A.** (1936). The use of multiple measurements in taxonomic problems, *Annals of eugenics*, *7*(2), 179–188.

[142] **Zhang, A.**, **Yang, B. and Huang, L.** (2008). Feature extraction of EEG signals using power spectral entropy, *BioMedical Engineering and Informatics, 2008. BMEI 2008. International Conference on*, volume 2, IEEE, pp.435–439.

[143] **Chiappa, S. and Bengio, S.** (2003). HMM and IOHMM modeling of EEG rhythms for asynchronous BCI systems, **Technical Report**, IDIAP.

[144] **Kaper, M.** *et al.* (2004). BCI competition 2003-data set IIb: support vector machines for the P300 speller paradigm, *Biomedical Engineering, IEEE Transactions on*, *51*(6), 1073–1076.

[145] **Kaiser, V.** *et al.* (2011). First steps toward a motor imagery based stroke BCI: new strategy to set up a classifier, *Front Neurosci*, *5*, 86.

[146] **Brunner, C.** *et al.* (2011). A comparison of univariate, vector, bilinear autoregressive, and band power features for brain–computer interfaces, *Medical & biological engineering & computing*, *49*(11), 1337–1346.

[147] **Wang, T.**, **Deng, J. and He, B.** (2004). Classifying EEG-based motor imagery tasks by means of time–frequency synthesized spatial patterns, *Clinical Neurophysiology*, *115*(12), 2744–2753.

[148] **Penny, W.D.** *et al.* (2000). EEG-based communication: a pattern recognition approach, *IEEE Transactions on Rehabilitation Engineering*, *8*(2), 214–215.

[149] **Pfurtscheller, G.** *et al.* (1998). Separability of EEG signals recorded during right and left motor imagery using adaptive autoregressive parameters, *Rehabilitation Engineering, IEEE Transactions on*, *6*(3), 316–325.

[150] **Qin, L.**, **Ding, L. and He, B.** (2004). Motor imagery classification by means of source analysis for brain–computer interface applications, *Journal of Neural Engineering*, *1*(3), 135.

[151] **Kamousi, B.**, **Liu, Z. and He, B.** (2005). Classification of motor imagery tasks for brain-computer interface applications by means of two equivalent dipoles analysis, *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, *13*(2), 166–171.

[152] **Congedo, M.**, **Lotte, F. and Lécuyer, A.** (2006). Classification of movement intention by spatially filtered electromagnetic inverse solutions, *Physics in medicine and biology*, *51*(8), 1971.

[153] **Lotte, F.** *et al.* (2007). A review of classification algorithms for EEG-based brain–computer interfaces, *Journal of neural engineering*, *4*(2), R1.

[154] **Noh, E. and de Sa, V.R.** (2013). Canonical correlation approach to common spatial patterns, *Neural Engineering (NER), 2013 6th International IEEE/EMBS Conference on*, IEEE, pp.669–672.

[155] **Duda, R.O.**, **Hart, P.E. and Stork, D.G.** (2012). *Pattern classification*, John Wiley & Sons.

[156] **Cortes, C. and Vapnik, V.** (1995). Support vector machine, *Machine learning*, *20*(3), 273–297.

[157] **Blankertz, B.**, **Curio, G. and Muller, K.R.** (2002). Classifying single trial EEG: Towards brain computer interfacing, *Advances in neural information processing systems*, *1*, 157–164.

[158] **Burges, C.J.** (1998). A tutorial on support vector machines for pattern recognition, *Data mining and knowledge discovery*, *2*(2), 121–167.

[159] **Obermaier, B.** *et al.* (2001). Hidden Markov models for online classification of single trial EEG data, *Pattern recognition letters*, *22*(12), 1299–1309.

[160] **Palaniappan, R.** (2005). Brain computer interface design using band powers extracted during mental tasks, *Neural Engineering, 2005. Conference Proceedings. 2nd International IEEE EMBS Conference on*, IEEE, pp.321–324.

[161] **Garrett, D.** *et al.* (2003). Comparison of linear, nonlinear, and feature selection methods for EEG signal classification, *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, *11*(2), 141–144.

[162] **Hsu, W.Y.** (2012). Fuzzy Hopfield neural network clustering for single-trial motor imagery EEG classification, *Expert Systems with Applications*, *39*(1), 1055–1061.

[163] **Hoya, T.** *et al.* (2003). Classification of single trial EEG signals by a combined principal+ independent component analysis and probabilistic neural network approach, *Proc. ICA2003*, volume197.

[164] **Palaniappan, R.** *et al.* (2002). A new brain-computer interface design using fuzzy ARTMAP, *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, *10*(3), 140–148.

[165] **Kostov, A. and Polak, M.** (2000). Parallel man-machine training in development of EEG-based cursor control, *Rehabilitation Engineering, IEEE Transactions on*, *8*(2), 203–205.

[166] **Haselsteiner, E. and Pfurtscheller, G.** (2000). Using time-dependent neural networks for EEG classification, *Rehabilitation Engineering, IEEE Transactions on*, *8*(4), 457–463.

[167] **Barreto, A.B.**, **Taberner, A.M. and Vicente, L.M.** (1996). Classification of spatio-temporal EEG readiness potentials towards the development of a brain-computer interface, *Southeastcon'96. Bringing Together Education, Science and Technology., Proceedings of the IEEE*, IEEE, pp.99–102.

[168] **Kayikcioglu, T. and Aydemir, O.** (2010). A polynomial fitting and k-NN based approach for improving classification of motor imagery BCI data, *Pattern Recognition Letters*, *31*(11), 1207–1215.

[169] Positive Definite Matrix, `http://mathworld.wolfram.com/PositiveDefiniteMatrix.html`.

[170] **Bertsekas, D.P.** (1982). Constrained optimization and Lagrange multiplier methods, *Computer Science and Applied Mathematics, Boston: Academic Press, 1982*, *1*.

[171] **comte de Lagrange, J.L.** (1811). *Mecanique analytique, Nouvelle edition, revue et augmentee par l'auteur*, Veuve Courcier.

[172] **MATLAB** (2013). *version 8.1.0 (R2013a)*, The MathWorks Inc., Natick, Massachusetts.

[173] **Dornhege, G.** *et al.* (2004). Boosting bit rates in noninvasive EEG single-trial classifications by feature combination and multiclass paradigms, *Biomedical Engineering, IEEE Transactions on*, *51*(6), 993–1002.

[174] **Yeredor, A.** (2002). Non-orthogonal joint diagonalization in the least-squares sense with application in blind source separation, *Signal Processing, IEEE Transactions on*, *50*(7), 1545–1553.

[175] **Ziehe, A.** *et al.* (2003). A linear least-squares algorithm for joint diagonalization, *Proc. 4th Intern. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003)*, Citeseer, pp.469–474.

[176] **Vollgraf, R. and Obermayer, K.** (2006). Quadratic optimization for simultaneous matrix diagonalization, *Signal Processing, IEEE Transactions on*, *54*(9), 3270–3278.

[177] **Li, X.L. and Zhang, X.D.** (2007). Nonorthogonal joint diagonalization free of degenerate solution, *Signal Processing, IEEE Transactions on*, *55*(5), 1803–1814.

[178] **Tichavskỳ, P. and Yeredor, A.** (2009). Fast approximate joint diagonalization incorporating weight matrices, *Signal Processing, IEEE Transactions on*, *57*(3), 878–891.

[179] **Iferroudjene, R.**, **Meraim, K.A. and Belouchrani, A.** (2009). A new Jacobi-like method for joint diagonalization of arbitrary non-defective matrices, *Applied Mathematics and Computation*, *211*(2), 363–373.

[180] **Xu, X.F.**, **Feng, D.Z. and Zheng, W.X.** (2011). A fast algorithm for nonunitary joint diagonalization and its application to blind source separation, *Signal Processing, IEEE Transactions on*, *59*(7), 3457–3463.

[181] **Trainini, T. and Moreau, E.** (2011). A least squares algorithm for global joint decomposition of complex matrix sets, *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2011 4th IEEE International Workshop on*, IEEE, pp.313–316.

[182] **Wang, K.**, **Gong, X.F. and Lin, Q.H.**, (2012). Complex non-orthogonal joint diagonalization based on LU and LQ decompositions, Latent Variable Analysis and Signal Separation, Springer, pp.50–57.

[183] **Mesloub, A.**, **Abed-Meraim, K. and Belouchrani, A.** (2014). A new algorithm for complex non-orthogonal joint diagonalization based on shear and givens rotations, *Signal Processing, IEEE Transactions on*, *62*(8), 1913–1925.

[184] **Grosse-Wentrup, M. and Buss, M.** (2008). Multiclass common spatial patterns and information theoretic feature extraction, *Biomedical Engineering, IEEE Transactions on*, *55*(8), 1991–2000.

[185] **Rumelhart, D.E.**, **Hintont, G.E. and Williams, R.J.** (1986). Learning representations by back-propagating errors, *Nature*, *323*(6088), 533–536.

[186] **Levenberg, K.** (1944). A method for the solution of certain problems in least squares, *SIAM J Appl Math*, *11*(2), 431–441.

[187] **Marquardt, D.W.** (1963). An algorithm for least-squares estimation of nonlinear parameters, *Journal of the Society for Industrial & Applied Mathematics*, *11*(2), 431–441.

[188] **Yu, H. and Wilamowski, B.M.** (2011). Levenberg-marquardt training, *The Industrial Electronics Handbook*, *5*, 1–15.

[189] **Hagan, M.T. and Menhaj, M.B.** (1994). Training feedforward networks with the Marquardt algorithm, *Neural Networks, IEEE Transactions on*, *5*(6), 989–993.

[190] **Yamashita, N. and Fukushima, M.**, (2001). On the rate of convergence of the Levenberg-Marquardt method, Topics in numerical analysis, Springer, pp.239–249.

[191] **Moré, J.J.**, (1978). The Levenberg-Marquardt algorithm: implementation and theory, Numerical analysis, Springer, pp.105–116.

[192] **Ang, K.K.** *et al.* (2008). Filter bank common spatial pattern (FBCSP) in brain-computer interface, *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, IEEE, pp.2390–2397.

[193] **Ang, K.K. and Quek, C.** (2006). Rough Set-Based Neuro-Fuzzy System., *IJCNN*, pp.742–749.

[194] BCI Competitions, `http://www.bbci.de/competition/`.

[195] BCI competition III Data Set IVa, `http://www.bbci.de/competition/iii/#data_set_iva`.

[196] Electro Cap International, `http://www.electro-cap.com/`.

[197] Neuroscan, `http://compumedicsneuroscan.com/`.

[198] **Schlögl, A.** (2006). GDF-a general dataformat for biosignals, *arXiv preprint cs/0608052*.

[199] **Schlogl, A. and Brunner, C.** (2008). BioSig: a free and open source software library for BCI research, *Computer*, *41*(10), 44–50.

[200] Table of event codes for GDF, `https://github.com/donnchadh/biosig/blob/master/biosig/doc/eventcodes.txt`.

[201] **Seni, G. and Elder, J.F.** (2010). Ensemble methods in data mining: improving accuracy through combining predictions, *Synthesis Lectures on Data Mining and Knowledge Discovery*, *2*(1), 1–126.

[202] **Sokolova, M. and Lapalme, G.** (2009). A systematic analysis of performance measures for classification tasks, *Information Processing & Management*, *45*(4), 427–437.

[203] **Pfurtscheller, G.** *et al.* (2006). Mu rhythm (de) synchronization and EEG single-trial classification of different motor imagery tasks, *NeuroImage*, *31*(1), 153–159.

APPENDICES

APPENDIX A.1 : MATLAB Code for ERD demonstration

APPENDIX A.2 : MATLAB Code for Epoch parsing functions

APPENDIX A.3 : MATLAB Code for filtering epochs

APPENDIX A.4 : MATLAB Code for cropping epochs

APPENDIX A.5: MATLAB Code for splitting epochs

APPENDIX A.6 : MATLAB Code for CSP training& testing functions

APPENDIX A.7 : MATLAB Code for TR&SR-CSP training function

APPENDIX A.8 : MATLAB Code for SFN training& testing functions

APPENDIX A.9 : MATLAB Code for FBCSSP training& testing functions

APPENDIX A.10 : MATLAB Code for performance evaluation

APPENDIX A.11 : MATLAB Code for spatial filter visualization

APPENDIX A.1: MATLAB code for ERD demonstration

ERD demo application example. This demo script demostrates ERD signal for the subject defined in the code.

```
%%%%%%%%%% demoERD.m %%%%%%%%%%
clear all
close all
%% load data
channels={'C3','CZ'};
EPOCHS=parseEpochsIVA('AY',channels);
%% filter epochs
fparams.filterFreq=[8,12];
fparams.filterType='butter';
fparams.filterDegree=5;
fparams.fs=EPOCHS.fs;
EPOCHS=filterEpochs(EPOCHS,fparams);
%% crop epochs
cParams.t0=-1;  %after motor imagery trigger
cParams.t1=3.5;  %after motor imagery trigger
EPOCHS=cropEpochs(EPOCHS,cParams);
N=numel(channels);
K=numel(EPOCHS.EPDT);
T=size(EPOCHS.EPDT{1},2);
C=numel(unique(EPOCHS.EPLB));
for c=1:C
    EPPWR{c}=zeros(N,T);
    Nc(c)=0;
end
ma_len=40;
for k=1:K
   X= EPOCHS.EPDT{k};
   XP=X.*X;
   XP=tsmovavg(XP,'s',ma_len,2);
   c=EPOCHS.EPLB(k);
   EPPWR{c}=EPPWR{c} + XP;
   Nc(c)=Nc(c)+1;
end
for c=1:C
    EPPWR{c}=EPPWR{c}/ Nc(c);
end
txx=(1:T)/EPOCHS.fs+cParams.t0;
clsName{1}='Right Hand';
clsName{2}='Foot';
for c=1:C
    EPPWR{c}=EPPWR{c}/ Nc(c);
    for n=1:N
        EPPWR{c}(n,:)=100*EPPWR{c}(n,:)/max(EPPWR{c} ...
            (n,ma_len:ceil(T/4)));
    end
    figure('Position', [100, 100, 800, 600]);
    co = get(gca,'ColorOrder') % Initial
    % Change to new colors.
```

```
    set(gca, 'ColorOrder', [0 0 1; 1 0 0], 'NextPlot', ...
        'replacechildren');
    plot(txx,EPPWR{c}','linewidth',2);
    line([0,0],[20,120],'LineWidth',2,'LineStyle','-.',...
        'Color','k')
    legend(channels);
    xlabel('time (seconds)');
    ylabel('mean signal power,moving average, relative(%)');
    title(sprintf('Motor imagery ERD signal for class "%s"'...
        ,clsName{c}));
    grid
end
```

APPENDIX A.2: Epoch parsing functions

These functions is specific to the data format given as input. Function prototypes for epoch loading are given below,

```
function EPOCHS = parseEpochsIVA(subjectName, channels)
function EPOCHS = parseEpochsIIIA(subjectName, channels)
```

where, input and output structures are described below.

- `subjectName`: String type variable. Subject name of the data set. For dataset IVA, subject names should be one of *aa*, *al*, *av*, *aw* or *ay*. For dataset IIIA, subject names should be *k3b*, *k6b* or *l1b*.
- `channels`: Cell array of channel names which are being used for analyzing. Channel names are different for both data sets. For dataset IVA, channel names are standard 10-20 electrode names as given in Figure 4.1 and for dataset IIIA, channel names are as given in Figure 4.3.
- `EPOCHS`: Structured output data involving parsed epochs and data set information. Structure is described as follows,
  - `EPOCHS.EPDT`: Cell array of all parsed epochs. An epoch is a matrix with T columns and N rows. Where T is the number of samples in a trial and N is the number of channels selected.
  - `EPOCHS.EPLB`: Array of class labels. For data set IVA, class labels are [1,2] and for data set IIIA, class labels are [1,2,3 and 4].
  - `EPOCHS.channelNames`: Cell array of strings describing the channel name of each row in `EPDT`.
  - `EPOCHS.positions`: A two column matrix containing the x and y positions of each channel whom number of rows is equal to the number of channel names.
  - `EPOCHS.fs`: Integer describing the sampling rate in Hertz for the given data set.
  - `EPOCHS.trigtime`: Double type number describing the timing offset where the cue is displayed for the given data set.

Epoch parser functions creates epochs for further processing. In this function, the starting and ending of an epoch is determined according to the cue onset time with a fixed offset. For data set IVA, it is 3.5 seconds or 350 samples ($fs = 100$). For data set IVA, it is 3 seconds or 750 samples ($fs = 250$).

Example:

Load subject *AA* from data set IVA. The channels are the row starting with T7 and ending with T8 in Figure 4.1.

```
channels={'T7' 'C5' 'C3' 'C1' 'CZ' 'C2' 'C4' 'C6' 'T8'} ;
EPOCHS=parseEpochsIVA('aa', channels) ;
```

When the command executed, EPOCHS variable will be displayed in MATLAB as below,

```
EPOCHES=
EPDT: {1x280 cell}
EPLB: [1x280 double]
channelNames: {1x9 cell}
fs: 100
trigtime: 3.5000
positions: [9x2 double]
```

MATLAB Code for Epoch Parsing Functions

```
%%%%%%%%%% parseEpochsIVA.m %%%%%%%%%%
function EPOCHS=parseEpochsIVA(subjectName, channels)
EPOCHS=[];
%load data file:
dataname=sprintf('data_set_IVa_%s.mat',subjectName);
load(dataname);

%load true labels:
trueLabelName=sprintf('true_labels_%s.mat',subjectName);
load(trueLabelName);

%set channels:
channel_indexes=[];

for chi=1:numel(channels)
    for chj=1:numel(nfo.clab)
        if(strcmpi(nfo.clab{chj},channels{chi}))
            channel_indexes(chi)=chj;
            break;
        end
    end
    if(~strcmpi(nfo.clab{chj},channels{chi}))
        fprintf('channel name %s not found!\n',channels{chi});
        return;
    end

end

%gen number of channels:
N=numel(channel_indexes);
```

```
%get number of events:
ev_sz=numel(mrk.pos);

%epoch offset in seconds
epoch_offset=3.5;

for ei=1:ev_sz
    epst=mrk.pos(ei)-ceil(nfo.fs*epoch_offset);
    epend=mrk.pos(ei)+ceil(nfo.fs*epoch_offset);

    data=0.1*double(cnt(epst:epend,channel_indexes));

    EPOCHS.EPDT{ei}=data';
    EPOCHS.EPLB(ei)=true_y(ei);
end


%set channel names, sampling rate, trig time
EPOCHS.channelNames=channels;
EPOCHS.fs=nfo.fs;
EPOCHS.trigtime=epoch_offset;
EPOCHS.positions=[nfo.xpos(channel_indexes),nfo.ypos(channel_indexes)];


%%%%%%%%%% parseEpochsIIIA.m %%%%%%%%%%
function EPOCHS=parseEpochsIIIA(subjectName, channels)
EPOCHS=[];
%load gdf data file:
dataname=sprintf('%s.gdf',subjectName);
[s,HDR]=sload(dataname,0, 'OVERFLOWDETECTION:OFF');
%load true labels:
trueLabelName=sprintf('true_labels_%s.mat',subjectName);
load(trueLabelName);
%set channels:
channel_indexes=[];
for chi=1:numel(channels)
    for chj=1:HDR.NS
        if(strcmpi(HDR.Label{chj}(1,1:4),channels{chi}))
            channel_indexes(chi)=chj;
            break;
        end
    end
    if(~strcmpi(HDR.Label{chj}(1,1:4),channels{chi}))
        fprintf('channel name %s not found!\n',channels{chi});
        HDR.Label
        channels
        return;
    end
end

evnt=0;
fs=HDR.SampleRate;
for k=1:numel(HDR.TRIG)

    if(~HDR.ArtifactSelection(k))
        epst=HDR.TRIG(k);
        epend=HDR.TRIG(k)+(fs*7);
```

```
        data=double(s(epst:epend,channel_indexes));
        class=classlabel(k);
        evnt=evnt+1;
        XEP{evnt}=  data';
        CLS(evnt)=  class;
    end
end

EPOCHS.EPDT=XEP;
EPOCHS.EPLB=CLS';
EPOCHS.channelNames=channels;
EPOCHS.fs=HDR.SampleRate;
EPOCHS.trigtime=3.0;
```

APPENDIX A.3: Epoch filtering function

This function accepts EPOCHS data type as input and filters every single epoch data according to the filter specifications given as input parameters.The function prototype is given below:

```
function EPOCHSOUT = filterEpochs(EPOCHS, fParams)
```

where `EPOCHS` and `EPOCHSOUT` are input and ouput data that are in epochs format described in epoch parser function. `fparams` is a structure containing the band pass filter parameters. The elements of filter params structure are described below.

- `fparams.filterFreq`: 1x2 array containing the band pass filter cut off frequencies defined in Hertz. (i.e. $[f_1, f_2]$). Note that,if not defined, default value of filter frequency is 8 to 30 Hz .
- `fparams.filterType`: String type variable defining the filter type to be used. Valid filter types are `butter` for Butterworth digital filter, `cheby1` for Chebyshev Type I digital filter or `fir1` for finite impulse response filter. Note that,if not defined, default filter type is `butter`.
- `fparams.ripple`: Determines the ripple value if filter type is selected as `cheby1`.
- `fparams.filterDegree`: Filter degree (N). Default value is 5.
- `fparams.fs`: Sample rate of the given data. This parameter is required and must be defined in order to calculate the normalized cut off frequencies.

Example: Let the filter to be used is a fifth order Butterworth filter with a pass band 8 to 14 Hz. Let `EPOCHS` be the input data created previously by `parseEpochs` function. A sample code for setting up the filter parameters and filtering the input data and overwriting on it would be as below:

```
fparams.filterFreq=[8,14];
fparams.filterType='butter';
fparams.filterDegree=5;
fparams.fs=EPOCHS.fs;
EPOCHS=filterEpochs(EPOCHS,fparams);
```

Matlab code for the filterEpochs function is given below.

MATLAB Code for Epoch Filtering function:

```
%%%%%%%%%% filterEpochs.m %%%%%%%%%%
function EPOCHSOUT=filterEpochs(EPOCHS,fparams)
%set defaults if not defined:
if(~isfield(fparams,'filterFreq'))
     fparams.filterFreq=[8,30];
end
if(~isfield(fparams,'filterType'))
     fparams.filterType='butter';
end
if(~isfield(fparams,'filterDegree'))
     fparams.filterDegree=5;
end
if(~isfield(fparams,'fs'))
     fprintf('SAMPLE FREQUENCY (fparams.fs) MUST BE GIVEN!\n');
     EPOCHS_OUT={};
     return;
end
%number of samples
T=size(EPOCHS.EPDT{1},2);
%number of channels
N=size(EPOCHS.EPDT{1},1);
%number of epochs in training set
K=numel(EPOCHS.EPDT);
switch(fparams.filterType)
    case 'butter'
        [F_B,F_A]=butter(fparams.filterDegree,...
            [fparams.filterFreq(1)*2/fparams.fs, ...
            fparams.filterFreq(2)*2/fparams.fs]);
    case 'cheby1'
        if(~isfield(params,'ripple'))
             fprintf(['RIPPLE VALUE (params.ripple) MUST BE GIVEN' ...
             'FOR CHEBY1!\n']);
             return;
        end
         [F_B,F_A]=cheby1(fparams.filterDegree,fparams.ripple,...
             [fparams.filterFreq(1)*2/fparams.fs, ...
             fparams.filterFreq(2)*2/fparams.fs]);
    case 'fir1'
        [F_B,F_A]=fir1(fparams.filterDegree,...
        [fparams.filterFreq(1)*2/fparams.fs, ...
        fparams.filterFreq(2)*2/fparams.fs]);
    otherwise
        fprintf('FILTER TYPE %s NOT DEFINED\n',fparams.filterType );
end

if(isfield(fparams,'showFilter'))
    if(fparams.showFilter)
        figure('Position', [100, 100, 800, 400]);
        [H,F]=freqz(F_B,F_A,512,fparams.fs);
        plot(F,abs(H),'k-','LineWidth',2);
        line([0,50],[0.707,0.707],'Color','r',...
            'LineWidth',2,'LineStyle','-.');
        grid;
```

```
        xlabel('frequency (Hz)');
        ylabel('Amplitude');
        title(sprintf('frequency response for %s degree %d',...
            fparams.filterType, fparams.filterDegree));
    end
end

EPOCHSOUT=EPOCHS;
for k=1:K
        X=EPOCHS.EPDT{k};
        XF=filter(F_B,F_A,X')';
        EPOCHSOUT.EPDT{k}=XF;
end
```

APPENDIX A.4: Epoch Cropping Function

This function crops all epochs of the `EPOCH` structure in time according to the cropping parameters for removing the part of the signal not related with motor imagery while keeping the interval that is most likely to involve the motor imagery signal. The cropping operation is applied identically to all channels of all epochs in the input data. The cropping interval is determined manually and generally it is about between 1 to 3 seconds after the trigger. Note that, since epoch crop function is applied after the epoch filter function, we can get rid of the "rising up" period of the filter by this operation. The prototype for the epoch crop function is as follows:

```
function EPOCHSOUT=cropEpochs(EPOCHS,cparams)
```

Where, `EPOCHS` and `EPOCHSOUT` are input and ouput data that are in epochs format described in epoch parser function. `cparams` is a structure containing the crop parameters. The elements of `cparams` is given below:

- `cparams.t0`: Double type variable defining the beginning of time window for cropping. The value is defined in seconds and it is related to the trigger time. Therefore, a positive value means a time window starting before the trigger and a positive value means a time window starting after the trigger.
- `cparams.t1`: Double type variable defining the end of time window for cropping. Similar to the `cparams.t0`, the value is defined in seconds and it is related to the trigger time. Note that `cparams.t1` should be bigger than `cparams.t0`.

Example: Let `EPOCHS` be a variable obtained after parsing and filtering functions. Let the new time limits of the epoch data be 0.5 seconds to 2.5 seconds after the trigger time. (i.e. in case of Figure 4.4, this will means the interval from 3.5 s. to 5.5 s.). According to this, epoch cropping function would be called as below:

```
cParams.t0=0.5;  %after motor imagery trigger
cParams.t1=2.5;  %after motor imagery trigger
EPOCHS=cropEpochs(EPOCHS,cParams);
```

MATLAB Code for Epoch Cropping Function

```
%%%%%%%%%% cropEpochs.m %%%%%%%%%%
function EPOCHSOUT=cropEpochs(EPOCHS,cparams)

if(~isfield(cparams,'t0') || ~isfield(cparams,'t1'))
     fprintf('params.t0 & params.t1 are required');
     return;
end

n0=(cparams.t0+EPOCHS.trigtime)*EPOCHS.fs;
n1=(cparams.t1+EPOCHS.trigtime)*EPOCHS.fs;

%number of epochs in training set
K=numel(EPOCHS.EPDT);
EPOCHSOUT=EPOCHS;
for k=1:K
        X=EPOCHS.EPDT{k};
        EPOCHSOUT.EPDT{k}=X(:,n0:n1);
end
```

APPENDIX A.5: Epoch Splitting Function

Epoch split function accepts EPOCH data type as input and creates test and train sets with their true labels according to given indexes. The output of this function may be used by the successive classification algorithms. Function prototype is specified as,

```
function [TRDT,TSDT,TRLB,TSLB]=splitEpochs(EPOCHS,TRIDX,TSIDX)
```

Where, `EPOCHS` is input data that was in epochs format and described in epoch parser function. Other input and output parameters are described below:

- `TRIDX`: Logical array with K elements where K is the number of epochs in the `EPOCHS` structure. In the logical array, the indexes with true (1) means to put the corresponding epoch into the training set.
- `TSIDX`: Logical array with K elements where K is the number of epochs in the `EPOCHS` structure. In the logical array, the indexes with true (1) means to put the corresponding epoch into the test set.
- `TRDT`: Cell array of training set containing the epoch data matrices. The number of elements in the cell array is determined by the `sparams.TRIDX` variable.
- `TRLB`: Integer array containing the true labels of the training set. The class labels are represented with integer numbers (1,2,3,etc..).
- `TSDT`: Cell array of test set containing the epoch data matrices.
- `TSLB`: Integer array containing the true labels of the test set.

The matlab code for `splitEpochs` function is given below.

MATLAB Code for Epoch Splitting Function

```
%%%%%%%%%% splitEpochs.m %%%%%%%%%%
```

```
function [TRDT,TSDT,TRLB,TSLB]=splitEpochs(EPOCHS,TRIDX,TSIDX)

    TRDT=EPOCHS.EPDT(TRIDX);
    TSDT=EPOCHS.EPDT(TSIDX);

    TRLB=EPOCHS.EPLB(TRIDX);
    TSLB=EPOCHS.EPLB(TSIDX);
end
```

APPENDIX A.6: CSP Train & Test Functions

CSP Train function implements the common spatial patterns algorithm and outputs CSP spatial filters along with λ values related to each calculated spatial filter. The function name is `train_csp` and its prototype is defined as,

```
function [WCSP,L]=train_csp(TRDATA,TRLB,trainParams)
```

The description of the input and output arguments of the `train_csp` function is given below,

- `TRDATA`: Cell array of training epochs.
- `TRLB`: Array of true labels for the training data set.
- `trainParams`: structure containing the options for `train_csp` function. The following option is defined:
  - `trainParams.m`:the number of spatial filters used per class while building the CSP matrix.
- `WCSP`: holds the optimized spatial filter coefficients which is a *mCxN* matrix. Where, *m* is `trainParams.m` and *C* is the number of classes in the training data. Note that, `train_csp` function extracts *C* from the `TRLB` automatically.
- `L`:holds the λ value for each spatial filter. `L` is a cell based array containing *C* cells and each cell includes *m* λ values.

CSP test function applies calculated spatial filters to training and test data sets, generates the feature sets, classifies the test features with a classifier and outputs the estimated labels of the test data set. The prototype for the test function is given as,

```
function [LABELS,ZTR,ZTS]=test_csp(TSDATA,TRDATA,TRLB,WCSP,testparams)
```

- `TSDATA` and `TRDATA`: Cell array of testing and training epochs, respectively.
- `TRLB`: Array of true labels for the training data set. `WCSP`: the spatial filter matrix which is the output of the `train_csp` function.
- `testparams`: structure containing the options for `test_csp` function. The following option is defined:

  - `testparams.classifier`:the string type variable which describes the selected classifier. Valid options for the classifier are `'LDA'` for linear discriminant analysis classifier and `'SVM'` for the support vector machines classifier.

- `LABELS`: the array of estimated class numbers of the test set.
- `ZTR` and `ZTS`: Cell array containing the spatial filter output of the training and test sets, respectively.

MATLAB Code for CSP Train and Test Functions

```
%%%%%%%%%% train_csp.m %%%%%%%%%%
function [WCSP,L]=train_csp(TRDATA,TRLB,trainParams)
%number of classes
C=numel(unique(TRLB));
%number of channels
N=size(TRDATA{1},1);
%number of epochs in training set
K=numel(TRDATA);
%number of epochs in test set
% KTS=numel(TSDATA);
RA=zeros(N,N);
RB=zeros(N,N);
na=0;
nb=0;
for c=1:C
    R{c}=zeros(N,N);
    n(c)=0;
end
%% calculate average covariance matrices
for k=1:K
    X=TRDATA{k};
    r=X*X'/trace(X*X');
    c=TRLB(k);

    R{c}=R{c}+r;
    n(c)=n(c)+1;
end
for c=1:C
    R{c}=R{c}/n(c);
end
%% calculate spatial filters for each class
WCSP=[];    % csp matrix
for c=1:C
   %calculate num & den
   RA=R{c};
   RB=zeros(N,N);
   for ci=1:C
       if ci~=c
           RB=RB+R{ci};
       end
   end
   %calculate CSP matrix
   Q=inv(RB)*RA;
   [W A]=eig(Q);
   %sort eigenvalues in descending order
```

```
    [A order] = sort(diag(A),'descend');
    % sort eigen vectors
%     W=inv(W)';
    W = W(:,order);
    WCSP=[WCSP;W(:,1:trainParams.m)'];
    L{c}=A(1:trainParams.m);
end


%%%%%%%%%% test_csp.m %%%%%%%%%%
function [LABELS,ZTR,ZTS]=test_csp(TSDATA,TRDATA,TRLB,WCSP,params)

Ktr=numel(TRDATA);
for k=1:Ktr
  X=TRDATA{k};
  ZTR{k}=WCSP*X;
  ftr(k,:)=[log(var(ZTR{k},0,2)'/sum(var(ZTR{k},0,2)))];
end

Kts=numel(TSDATA);
for k=1:Kts
  X=TSDATA{k};
  ZTS{k}=WCSP*X;
  fts(k,:)=[log(var(ZTS{k},0,2)'/sum(var(ZTS{k},0,2)))];
end


if(params.classifier=='LDA')
    LABELS=classify(fts, ftr, TRLB)';
elseif(params.classifier=='SVM')
    svmStruct = svmtrain(ftr,TRLB);
    LABELS = svmclassify(svmStruct,fts)';
end
```

APPENDIX A.7: TR&SR-CSP Train Function

`train_trsrcsp` function optimizes spatial filters according to the proposed task related & spatially regularized common spatial patterns method. the prototype for this function is defined as,

```
function [WCSP L]=train_trsrcsp(TRDATA,TRLB,params)
```

Here, input parameters for the function are given below,

- `params.m`:the number of spatial filters used per class while building the CSP matrix.
- `params.r`: An hyper parameter representing the maximum distance between two electrodes.
- `params.alpha`: An hyper parameter representing multiplier of the penalty term in the regularized CSP method.
- `params.centerElectrodes`:A cell array of string containing the names of the electrodes defined as "center electrode" for the proposed TR&SR-CSP method.

- `params.electrodes`:A cell array of string containing the names of the electrodes used in the input data.
- `params.locations`:An *Nx*2 sized matrix which includes the positions of each used electrode in *x* and *y* axes, respectively.

The Matlab code for the `train_trsrcsp` function may be found below. Testing function for this regularized CSP function is the same with CSP test function that was described in the previous subsection.

As an example, `train_trsrcsp` and `test_csp` functions for dataset IVA may be called by the following arguments,

```
tparams.m=2;
tparams.r=0.7;
tparams.centerElectrodes={'C3','CZ'};
tparams.electrodes=EPOCHS.channelNames;
tparams.locations=EPOCHS.positions;
tparams.alpha=0.04;

[WCSP L]=train_trsrcsp(TRDT,TRLB,tparams);
tparams.classifier='LDA';
LABELS=test_csp(TSDT,TRDT,TRLB,WCSP,tparams);
```

Above, electrodes C3 and CZ were selected as center electrodes since their spatial locations correspond to the *right hand* and *foot* on the motor cortex.

MATLAB Code for TR&SR-CSP Train Function

```
%%%%%%%%%% train_trsrcsp.m %%%%%%%%%%
function [WCSP L]=train_trsrcsp(TRDATA,TRLB,params)

%number of classes
C=numel(unique(TRLB));
%number of channels
N=size(TRDATA{1},1);
%number of epochs in training set
K=numel(TRDATA);
%number of epochs in test set
% KTS=numel(TSDATA);
RA=zeros(N,N);
RB=zeros(N,N);
na=0;
nb=0;
for c=1:C
    R{c}=zeros(N,N);
    n(c)=0;
end
%% calculate average covariance matrices
for k=1:K
    X=TRDATA{k};
    r=X*X'/trace(X*X');
    c=TRLB(k);

    R{c}=R{c}+r;
    n(c)=n(c)+1;
```

```
end
for c=1:C
    R{c}=R{c}/trace(R{c});
end
WCSP=[];

for c=1:C
    PM{c}=zeros(N,N);
    CE=params.centerElectrodes{c}; %center electrode
     for ci=1:numel(params.electrodes)
                if(strcmpi(CE,params.electrodes{ci}))
                    chc=ci;
                    break;
                end
     end

    for n=1:N
         dij=sqrt((params.locations(chc,:)-params.locations(n,:))*...
             (params.locations(chc,:)-params.locations(n,:))');
         PM{c}(n,n)=1-exp(-(dij/params.r)^2);
    end
    PM{c}=PM{c}/trace(PM{c});
end


for c=1:C
    RA=R{c};
    RB=zeros(N,N);
    for ci=1:C
        if ci~=c
            RB=RB+R{ci};
        end
    end
    RB=RB/trace(RB);
    %calculate CSP matrix
    Q=inv((1-params.alpha)*RA+params.alpha*PM{c})*RB;
    [W A]=eig(Q);
    %sort eigenvalues in descending order
    [A order] = sort(diag(A),'descend');
    % sort eigen vectors
    W = W(:,order);
    WCSP=[WCSP;W(:,1:params.m)'];
    L{c}=A(1:params.m);

end
```

APPENDIX A.8: SFN Train & Test Functions

Train function for the spatial filter network (`train_sfn`) creates a spatial filter structure according to the given filter parameters and optimizes its weights by learning the input data. Presented function takes input parameters with the same convention. The function prototype for the `train_sfn` is given below,

```
function SFNNET=train_sfn(TRDATA,TRLB,sfnparams)
```

input `sfnparam` structure hold the following elements,

- `sfnparams.M`: Number of neurons in hidden layer. (default: equals to number of classes in the data set)
- `sfnparams.EMIN`: Error value for terminating the training (default: 0.1)
- `sfnparams.ITRMAX`: Maximum number of iterations for terminating the training (default: 1000)
- `sfnparams.trainMethod`:String type variable for the training method of the algorithm. Valid options are 'BP' for back propogation method and 'LM' for Levenberg-Marquardt method. (default: 'BP').
- `sfnparams.mu`: Learning rate (μ) for BP method or Initial combination coefficient for LM method. (default value is 1e-3 for BP and 100 for LM)
- `sfnparams.beta`: μ multiplier/divider for LM method (default: 2)
- `sfnparams.W`: Initial spatial filter layer matrix (defaullt: randomly generated)
- `sfnparams.V`: Initial classifier layer matrix (defaullt: randomly generated)
- `sfnparams.b`: Initial classifier layer bias vector (defaullt: randomly generated)
- `sfnparams.dbg`: Debug on/off (verbosity) (default: 0, no debug output)

And,the output `SFNNET` structure contains the following elements,

- `SFNNET.W`: Spatial filter layer weight matrix
- `SFNNET.V`: Classifier layer weight matrix
- `SFNNET.V`: Classifier layer bias vector
- `SFNNET.lastErr`: Final error value of the network at the end of the training
- `SFNNET.itr`: Number of iterations completed for training
- `SFNNET.mu`: Learning rate value if the training method is BP or final combination coefficient value at the end of the training if the training method is LM.
- `SFNNET.CLS`: A set of class labels containing each unique labels in `TRLB`.

SFN testing function `test_sfn` accepts created `SFNNET` structure and test data as inputs and outputs estimated labels as classifier output. Note that, training data set is not needed for `test_sfn` since the linear classifier included with the spatial filter network is already trained while running the `train_sfn` function. `test_sfn` is defined as below,

```
function [LABELS F]=test_sfn(SFNNET,TSDATA)
```

In the above function definition, `LABELS` is an array including the estimated class labels and `F` is a *KxM* matrix which is the output of first layer (log-variance features) to be classified with classifier layer where *K* is the number of epochs in the `TSDATA` and *M* is the number of neurons in the hidden layer.

`train_sfn` and `test_sfn` functions is given below.

MATLAB Code for SFN Train & Test Functions

```
%%%%%%%%%% train_sfn.m %%%%%%%%%%
function SFNNET=train_sfn(TRDATA,TRLB,sfnparams)

K=numel(TRDATA);        %number of epochs in training set
N=size(TRDATA{1},1);    %number of channels, number of input neurons
T=size(TRDATA{1},2);    %number of samples in an epoch
CLSLB=unique(TRLB);     %class labels
C=numel(CLSLB);         %number of classes in the trainin se
%number of output neurons
if(C==2)
    O=1;
    %create desired output vectors
    for k=1:K
        cls=TRLB(k);
        if(cls==CLSLB(1))
            DSR(k,1)=1;
        else
            DSR(k,1)=-1;
        end
    end
else
    O=C;
    %create desired output vectors
    DSR=-ones(K,O);
    for k=1:K
        cls=TRLB(k);
        cx=find(CLSLB==cls);
        DSR(k,cx)=1;
    end

end
%default params
EMIN=1e-1;      %maximum acceptable error
ITRMAX=1000;    %maximum number of iteration
M=C;          %number of neurons in hidden layer.
mu=1e-2;
dbg=0;          %verbosity
if(isfield(sfnparams,'M'))
    M=sfnparams.M;
end
if(isfield(sfnparams,'EMIN'))
    EMIN=sfnparams.EMIN;
end
if(isfield(sfnparams,'ITRMAX'))
    ITRMAX=sfnparams.ITRMAX;
end
if(isfield(sfnparams,'mu'))
    mu=sfnparams.mu;
end
```

```
if(isfield(sfnparams,'dbg'))
    dbg=sfnparams.dbg;
end
if(isfield(sfnparams,'W'))
    W=sfnparams.W;
else
    W=randn(N,M)/10;
end
if(isfield(sfnparams,'V'))
    V=sfnparams.V;
else
    V=randn(M,O)/10;
end
if(isfield(sfnparams,'b'))
    b=sfnparams.b;
else
    b=randn(O,1)/10;
end
if(isfield(sfnparams,'beta'))
    beta=sfnparams.beta;
else
    beta=2;
end
if(isfield(sfnparams,'trainMethod'))
    trainMethod=sfnparams.trainMethod;
else
    trainMethod='BP';
end
if(strcmpi(trainMethod,'BP'))
    ME=EMIN;     %mean error
    itr=0;       %iteration counter
    E=zeros(K*O,1); %error of each output for each epoch
    while(ME>=EMIN && itr<ITRMAX)
        itr=itr+1;
        wl=sqrt(diag(W'*W));
        wlx=ones(N,1)*wl';
        for k=1:K
            % forward net:
            X=TRDATA{k};
            y=(W./wlx)'*X;
            v=var(y')';
            f=log(v);
            z=V'*f+b;
            Phi=tanh(z);

            % backward net:
            E(k)= 0.5*(DSR(k,:)'-Phi)'*(DSR(k,:)'-Phi);
            dephi=Phi-DSR(k,:)';
            dPhiZ=1-(Phi).^2;
            dzV=f;
            dzf=V;
            dfy=(1./v)*ones(1,T)*(2/T).*(y);
            dzb=ones(O,1);
            DV=((dephi.*dPhiZ)*dzV')';
            DB=dephi.*dPhiZ.*dzb;
            for m=1:M
                dyw=(X*wl(m)-W(:,m)*y(m,:))/(wl(m)^2);
                DFYW(:,m)=dyw*dfy(m,:)';
            end
```

```
                DW=ones(N,1)*(V*(dephi.*dPhiZ))'.*DFYW;
                b=b-mu*DB;
                V=V-mu*DV;
                W=W-mu*DW;
            end
            ME=mean(E);
            errval(itr)=ME;
            if(dbg>0)
                disp(sprintf('ens=%d %0.4f', itr, ME));
            end
        end
        SFNNET.W=W;
        SFNNET.V=V;
        SFNNET.b=b;
        SFNNET.lastErr=ME;
        SFNNET.itr=itr;
        SFNNET.mu=mu;
        SFNNET.CLS=CLSLB;
    elseif(strcmpi(trainMethod,'LM'))
        ME=EMIN;     %mean error
        itr=0;       %iteration counter
        E=zeros(K*O,1); %error of each output for each epoch
        ME_=ME;  %previous error value.
        while(ME_>=EMIN && itr<ITRMAX)
            itr=itr+1;
            JM=zeros(O, M*O + O+N*M);
            wl=sqrt(diag(W'*W));
            wlx=ones(N,1)*wl';
            for k=1:K
                % forward net:
                X=TRDATA{k};
                y=(W./wlx)'*X;
                v=var(y')';
                f=log(v);
                z=V'*f+b;
                Phi=tanh(z);
                % backward net:
                dPhiZ=1-(Phi).^2;
                dzV=f;
                dzf=V;
                dfy=(1./v)*ones(1,T)*(2/T).*(y);
                dzb=ones(O,1);
                rown=(k-1)*O+1;
                coln=1;
                E(rown:rown+O-1)= DSR(k,:)'-Phi;
                dPhiZDiag=diag(dPhiZ);
                for m=1:M
                    JM(rown:rown+O-1,(m-1)*O+1:m*O)= -1*dPhiZDiag*dzV(m);
                end
                coln=coln+M*O;
                JM(rown:rown+O-1,coln:coln+O-1)=diag((-1*dPhiZ));
                coln=coln+O;
                for m=1:M
                    dyw=(X*wl(m)-W(:,m)*y(m,:))/(wl(m)^2);
                    DFYW(:,m)=dyw*dfy(m,:)';
                end
                DPhiZf=(-1*ones(M,1)*dPhiZ').*dzf;
                for m=1:M
                    JM(rown:rown+O-1,coln:coln+N-1)=(DFYW(:,m)*DPhiZf(m,:))';
```

```
                coln=coln+N;
            end
        end
        ME=mean(E.*E);  %calculate mean error
        if(ME_<ME && itr>1)
            mu=mu*beta;
            W=W_;
            V=V_;
            b=b_;
            if(dbg>0)
                disp(sprintf('ens=%d u=%d %0.8f, %0.8f',itr, mu, ME_, ME));
            end
        else
            W_=W;
            V_=V;
            b_=b;
            mu=mu/beta;
            if(dbg>0)
                disp(sprintf('ens=%d u=%d %0.8f, %0.8f *',itr,mu,ME_,ME));
            end
            ME_=mean(E.*E);
        end
        if(itr<ITRMAX)  %update, if this is not the last iteration
            HS1=JM'*JM;
            NN=size(HS1,1);
            DWW=inv(HS1+mu*eye(NN))*JM'* E;
            coln=1;
            DV=reshape(DWW(coln:coln+M*O-1),O,M)';
            coln=coln+M*O;
            DB=reshape(DWW(coln:coln+O-1),O,1);
            coln=coln+O;
            DW=reshape(DWW(coln:coln+N*M-1),N,M);
            coln=coln+N*M;
            %update the network
            V=V-DV;
            b=b-DB;
            W=W-DW;
        end
    end
    SFNNET.W=W;
    SFNNET.V=V;
    SFNNET.b=b;
    SFNNET.lastErr=ME_;
    SFNNET.itr=itr;
    SFNNET.mu=mu;
    SFNNET.CLS=unique(TRLB);
end


%%%%%%%%%% test_sfn.m %%%%%%%%%%
function [LABELS F]=test_sfn(SFNNET,TSDATA)

K=numel(TSDATA);        %number of epochs in test set
N=size(TSDATA{1},1);    %number of channels, number of input neurons
T=size(TSDATA{1},2);    %number of samples in an epoch
W=SFNNET.W;
V=SFNNET.V;
b=SFNNET.b;
```

```
N=size(W,1);              %W: NxM
M=size(W,2);
O=size(V,2);              %V: MxO
%create desired output vector for each class
C=numel(SFNNET.CLS);
DSR=-ones(C,O);
for c=1:C
    DSR(c,c)=1;
end
wl=sqrt(diag(W'*W));
wlx=ones(N,1)*wl';
W=W./wlx;
for k=1:K
    X=TSDATA{k};
    y=W'*X;
    f=log(var(y'))';
    F(k,:)=f';
    z=V'*f+b;
    Phi=tanh(z);
    for c=1:C
        d(c)=(DSR(c,:)-Phi')*(DSR(c,:)-Phi')';
    end
    [mind minx]=min(d);
    LABELS(k)=SFNNET.CLS(minx);
end
```

APPENDIX A.9: FBCSSP Train & Test Functions

Training function of filter bank common spectra-spatial patterns method (`train_fbcssp`) is defined defined by the following function prototype,

```
function FBCSSP=train_fbcssp(TRDATA,TRLB,params)
```

Input parameters `params` holds the following elements:

- `params.bands`: A two column matrix whose each row stores the cut off frequencies (Hz) ($[f_1, f_2]$) of the FIR filters in the filter bank. The number of rows (F) in this matrix determines the number of filters.
- `params.fs`: Sampling frequency (Hz) which is required for FBCSSP in order to create the FIR filters.
- `params.P`: Degree of FIR filters (P)to be created by the algorithm.
- `params.m`: Number of spatial filters per class for one CSP block (m) in the first CSP layer. Note that, if the number of classes is C, then, total $m*C*F$ spatial filters will be generated in the first CSP layer.
- `params.r`: Number of spatial filters per class in the second CSP layer. Therefore, the output signal of the FBCSSP should have $r*C$ channels.

And the following elements stays in the output structure `FBCSSP`,

- `FBCSSP.W1`: The cell array of spatial filter matrices for the first CSP layer. `W1` includes F cells and each cell contains the $m*CxN$ sized spatial filter matrix specific to the associated filter bank. Where, N is the number of channels in the input signal.
- `FBCSSP.W2`: A matrix for the second CSP layer with size $(r*C)x(m*C*F)$. The role of this matrix in the FBCSSP structure is mainly a frequency selection by weighting different frequency bands.
- `FBCSSP.H`: A cell array of filter coefficients for each FIR filter in the filter bank. The number of cells in the cell array is F and each cell contains an array with P coefficients.

Test function for FBCSSP is called `test_fbcssp` and it is defined as,

```
[LABELS,ZTR,ZTS]=test_fbcssp(FBCSSP,TRDATA,TRLB,TSDATA,params)
```

where,`FBCSSP` is the structure for FBCSSP network that is the output of the `train_fbcssp` function. `params` is input parameter structure to the test function and it holds a single option `params.classifier` for classifier selection. `LDA` for linear discriminant analysis and `SVM`for support vector machines. As in the `test_csp` funct,on, the output is estimated classifier output and filter outputs for train and test sets.

A code sample for calling FBCSSP is given below,

```
params.bands=[2,10; 8,16;14,22; 20,28;26,34;32,40;38,46];
params.fs=EPOCHS.fs;
params.P=20;
params.m=2;
params.r=1;
FBCSSP=train_fbcssp(TRDT,TRLB,params)
params.classifier='LDA';
LABELS=test_fbcssp(FBCSSP,TRDT,TRLB,TSDT,params);
```

In the above code example, the filter bank includes 7 filters whose cut off frequencies are defined with `params.bands`. The degree of the FIR filters in the filter bank is 20, number of spatial filters per class is 2 and 1 at the first and second CSP layers, respectively. `train_fbcssp` creates `FBCSSP` structure for `test_fbcssp` function which classifies the input test data `TSDT` with a LDA classifier.

MATLAB Code for FBCSSP Train & Test Functions

```
%%%%%%%%%% train_fbcssp.m %%%%%%%%%%
function FBCSSP=train_fbcssp(TRDATA,TRLB,params)

bands=params.bands;
F=size(bands,1);
P=params.P;
C=numel(unique(TRLB));  %number of classes
KTR=numel(TRDATA);
%% filter bank
for s=1:F
    H{s}=fir1(P,[bands(s,1)*2/params.fs,...
```

```
        bands(s,2)*2/params.fs]);
    for k=1:KTR
        x=TRDATA{k};
        xf=filter(H{s},1,x')';
        XF{s}{k}=xf;
    end
end

%% first stage CSP1
m= params.m;
cspparams.m=m;
for s=1:F
    W1{s}=train_csp(XF{s},TRLB,cspparams);
    for k=1:KTR
      X=XF{s}{k};
      OUT{k}=W1{s}*X;
    end
    Y1S{s}=OUT;
end
fprintf('input size:%dx%dx%dx%d\n',...
    F,numel(XF{1}),size(XF{1}{1},1),size(XF{1}{1},2));

for k=1:KTR
    rown=1;
    for s=1:F
        Y1{k}(rown:rown+C*m-1,:) = Y1S{s}{k};
        rown=rown+C*m;
    end
end

%% 2. stage
cspparams.m= params.r;
W2=train_csp(Y1,TRLB,cspparams);

FBCSSP.W1=W1;
FBCSSP.W2=W2;
FBCSSP.H=H;


%%%%%%%%%% test_fbcssp.m %%%%%%%%%%
function [LABELS,ZTR,ZTS]=test_fbcssp(FBCSSP,TRDATA,TRLB,TSDATA,params)


W1=FBCSSP.W1;
W2=FBCSSP.W2;
H=FBCSSP.H;

F=numel(H);
KTR=numel(TRDATA);
KTS=numel(TSDATA);
C=numel(unique(TRLB));

for s=1:F
    h=H{s};
    for k=1:KTR
        x=TRDATA{k};
        xf=filter(h,1,x')';
        XF{s}{k}=xf;
```

```
    end
end


for s=1:F
    for k=1:KTR
          X=XF{s}{k};
          OUT{k}=W1{s}*X;
    end
    Y1S{s}=OUT;
end

m=size(W1{1},1)/C;
for k=1:KTR
    rown=1;
    for s=1:F
        Y1{k}(rown:rown+C*m-1,:) = Y1S{s}{k};
        rown=rown+C*m;
    end
    ZTR{k}=W2*Y1{k};
    ftr(k,:)=[log(var(ZTR{k}')/sum(var(ZTR{k}')))];
end

%% test data

for s=1:F
    h=H{s};
    for k=1:KTS
        x=TSDATA{k};
        xf=filter(h,1,x')';
        XFT{s}{k}=xf;
    end
end

for s=1:F
    for k=1:KTS
          X=XFT{s}{k};
          OUTT{k}=W1{s}*X;
    end
    Y1ST{s}=OUTT;
end

m=size(W1{1},1)/C;
for k=1:KTS
    rown=1;
    for s=1:F
        Y1T{k}(rown:rown+C*m-1,:) = Y1ST{s}{k};
        rown=rown+C*m;
    end
    ZTS{k}=W2*Y1T{k};
    fts(k,:)=[log(var(ZTS{k}')/sum(var(ZTS{k}')))];
end

if(params.classifier=='LDA')
    LABELS=classify(fts, ftr, TRLB)';
elseif(params.classifier=='SVM')
    svmStruct = svmtrain(ftr,TRLB);
    LABELS = svmclassify(svmStruct,fts)';
end
```

APPENDIX A.10: Performance Evaluation Function

The function implemented for performance evaluation called `perfCalc` evaluates accuracy and other performance measurements that are *precision*, *sensitivity*, *specificity* and *f-score*. `perfCalc` is defined by the following prototype,

```
function PERF=perfCalc(LB,TRLB)
```

Where, `LB` is the output labels of the classifier and `TRLB` is the true labels for the given data set. `perfCalc` function outputs a structured variable `PERF`, which holds the following performance evaluations,

- `PERF.confM`: The confusion matrix of size *CxC* obtained by comparing predicted and true labels.
- `PERF.TP`: A vector with C elements containing number of true positives (TP_c) for each class.
- `PERF.FP`: A vector with C elements containing number of false positives (FP_c) for each class.
- `PERF.TN`: A vector with C elements containing number of true negatives (TN_c) for each class.
- `PERF.FP`: A vector with C elements containing number of false negatives (FN_c) for each class.
- `PERF.ACC`: Overall accuracy of the classifier calculated by formula in (4.1)
- `PERF.PRE`: A vector with C elements which holds precision ($TP_c/(TP_c+FP_c)$) of the classifier for each class.
- `PERF.SENS`: A vector with C elements which holds sensitivity ($TP_c/(TP_c+FN_c)$) of the classifier for each class.
- `PERF.SPEC`: A vector with C elements which holds specificity ($TN_c/(TN_c+FP_c)$) of the classifier for each class.
- `PERF.SPEC`: A vector with C elements containing F1-score ($2*precision*sensitivity/(precision+sensitivity)$) of the classifier for each class.

MATLAB Code for Performance Evaluation Function

```
%%%%%%%%%% perfCalc.m %%%%%%%%%%
function PERF=perfCalc(LB,TRLB)
K=numel(LB);
C=numel(unique(LB));
%create confusion matrix
confM=zeros(C,C);
for k=1:K
    ctr=LB(k);
    ces=TRLB(k);
    confM(ctr,ces)=confM(ctr,ces)+1;
```

```
end
%% calculate number of true/false positive/negatives
for c=1:C
    TP(c)=confM(c,c);
    FP(c)=sum(confM(:,c))-TP(c);
    FN(c)=sum(confM(c,:))-TP(c);
    TN(c)=K-TP(c)-FP(c)-FN(c);
end
%% calculate metrics
ACC=0;
PRE=zeros(1,C);
SENS=zeros(1,C);
SPEC=zeros(1,C);
F1SC=zeros(1,C);
for c=1:C
    ACC=ACC+(TP(c))/(TP(c)+TN(c)+FP(c)+FN(c));
    PRE(c)=(TP(c))/(TP(c)+FP(c));
    SENS(c)=(TP(c))/(TP(c)+FN(c));
    SPEC(c)=(TN(c))/(TN(c)+FP(c));
    F1SC(c)=2* PRE(c)*SENS(c)/(PRE(c)+SENS(c));
end
PERF.confM=confM;
PERF.TP=TP;
PERF.TN=TN;
PERF.FP=FP;
PERF.FN=FN;
PERF.ACC=ACC;
PERF.PRE=PRE;
PERF.SENS=SENS;
PERF.SPEC=SPEC;
PERF.F1SC=F1SC;
```

APPENDIX A.11: Spatial Filter Visualization Function

In order to visualize the spatial filters, Matlab function called `img_head` was coded. This function is defined as below,

```
function img_head(values,imgparams)
```

This function accepts spatial filter coefficient values and used channel names as input and outputs a head figure according to the given input parameters. The input arguments of the function is listed below,

- `values`: A vector containing N elements that are the spatial filter coefficients to be plotted where, N is the number of channels. Note that, since the amplitudes is important for visualizing, filter coefficients should be `values` absolutes or squares of the coefficients.
- `imgparams.channels`: The cell array with N elements holding the channel names used.
- `imgparams.locations`: An $Nx2$ sized matrix which includes the positions of each used electrode in x and y axes, respectively.

- `imgparams.caxis`: Optional parameter (2x1 array) for setting color axis describing the levels corresponding to white and black. By default, color axis is set automatically grading from white to black.
- `imgparams.plotcontour`: Optional parameter describing the number of contour plots to be plotted. By default, it is set to zero and no contour is plotted.
- `imgparams.plottext`: Optional parameter for printing channel names near the electrodes. By default, channel names are not printed. Set to 1 to display channel names.

A sample code for calling `img_head` is given below. Let `WCSP` is a $4xN$ spatial filter matrix whose first and last rows corresponds to the most important spatial filters for either classes. Therefore, generally the first and last rows are plotted while displaying the spatial filters. Also note that, absolute values of spatial filters are given as input argument to the plotting function because amplitudes are important for demonstrating the significance of certain channels.

```
%% plot head figures
imgparams.channels=EPOCHS.channelNames;
imgparams.locations=EPOCHS.positions;
imgparams.plotcontour=4;
imgparams.plottext=0;
figure;
img_head(abs(WCSP(1,:)),imgparams)
figure;
img_head(abs(WCSP(2,:)),imgparams)
%%%%%%%%%%%%%%%%%%%%%%%%
```

MATLAB Code for Spatial Filter Visualization Function

```
%%%%%%%%%% img_head.m %%%%%%%%%%
function img_head(values,imgparams)

x=imgparams.locations(:,1);
y=-imgparams.locations(:,2);

z=double(values);
if(size(z,1)<size(z,2))
    z=z';
end

r_set=1.1;
%draw angular
n = 100;
r = r_set*((0:(n-1))'/(n-1));
theta = pi*(-n:(n-1))/(n-1);
X = r*cos(theta);
Y = r*sin(theta);
%%extend extrapolation to the edges of head:
n=10;
r=r_set;
theta = pi*(-n:(n-1))/n;
x=[x;r*cos(theta)'];
```

```
y=[y;r*sin(theta)'];
z=[z;zeros(2*n,1)];
Z = griddata(x,y,z,X,Y,'cubic');
%plot extrapolated image
pcolor(X,Y,Z);
%prepare colormap
for coli=1:32
    cmp(coli,:)=[abs(1-coli/32),abs(1-coli/32),...
        abs(1-coli/32)];
end
colormap(cmp);
%color axis manual:
if(isfield(imgparams,'caxis'))
    caxis(imgparams.caxis);
end
shading flat
hold on
%plot countours:
if(isfield(imgparams,'plotcontour'))
    if(imgparams.plotcontour>0)
        contour(X,Y,Z,imgparams.plotcontour,'k');
    end
end
%draw head circle
rectangle('Curvature',[1,1],'Position',...
    [-1*r_set,-1*r_set,2*r_set,2*r_set],...
    'EdgeColor',[0,0,0],'LineWidth',3);

for i=1:numel(imgparams.channels)
    %draw electrode positions
    scatter(x(i), y(i),10,'MarkerEdgeColor',[0,0,0],'LineWidth',2);
    %draw electrode names
    if(isfield(imgparams,'plottext'))
        if(imgparams.plottext==1)
            text(x(i),y(i),['\bf ' channels{i}],'FontSize',8);
        end
    end
end
%draw a tiny nose
scatter(0,-r_set-0.025,100,'^',...
    'MarkerEdgeColor',[0,0,0],'LineWidth',2)
%set axis
axis([-r_set, r_set, -r_set, r_set]);
axis ij equal off
hold off;
```

CURRICULUM VITAE

Name Surname: Ayhan Yuksel

Place and Date of Birth: Istanbul 20.12.1982

E-Mail: yukselay@itu.edu.tr

EDUCATION:

- **B.Sc.:** 2005, Istanbul Technical University, Faculty of Electrical and Electronic Engineering, Electronics & Communication Engineering
- **M.Sc.:** 2008, Istanbul Technical University, Graduate School of Science Engineering and Technology , Biomedical Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2015-2016 Electronic R&D engineer at AORT medical company, İstanbul
- 2010-2013 Electronic R&D engineer at Sayısal Sistemler Tasarım Evi, İstanbul
- 2008-2015 TUBITAK PhD scolarship
- 2006-2010 Research Assistant at Istanbul Technical University
- 2005-2008 TUBITAK M.Sc scolarship

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Yuksel, A.**, & Olmez, T. (2015). A Neural Network-Based Optimal Spatial Filter Design Method for Motor Imagery Classification. PloS one, 10(5), e0125039.
- **Yüksel, A.**, & Ölmez, T. (2014). Task Related and Spatially Regularized Common Spatial Patterns for Brain Computer Interfaces. In IWBBIO (pp. 42-53).

OTHER PUBLICATIONS, PRESENTATIONS AND PATENTS:

- Duman, M. E., **Yuksel, A.**, & Olmez, T. (2015, May). Divergent common spatial patterns method. In Signal Processing and Communications Applications Conference (SIU), 2015 23th (pp. 612-615). IEEE.

- Dogan, B., & **Yuksel, A.** (2015, May). Analog filter group delay optimization using the Vortex Search algorithm. In Signal Processing and Communications Applications Conference (SIU), 2015 23th (pp. 288-291). IEEE.

- Yıldız, M., **Yuksel, A.**, Korürek, M., Aykan, A. Ç., Şahin Yıldız, B., & Şahin, A. Classification of Aatrial septal defect and ventricular septal defect with documented hemodynamic parameters via vardiac catheterization by genetic algorithms and multi-layered artificial neural network. Kosuyolu Kalp Derg 2012; 15: 45, 50.

- Korurek, M., Yildiz, M., **Yuksel, A.**, & Şahin, A. (2012). Simulation of Eisenmenger syndrome with ventricular septal defect using equivalent electronic system. Cardiology in the Young, 22(03), 301-306. ISO 690

- Yıldız, M., **Yuksel, A.**, Korürek, M., Aykan, A. Ç., Yıldız, B. Ş., Şahin, A., ... & Özkan, M. (2011). Kalp Kateterizasyonu ile Hemodinamik Ölçümleri Saptanmış Atriyal Septal Defekt ve Ventriküler Septal Defektli Olguların Genetik Algoritmalar ve Çok Katmanlı Yapay Sinir Ağı ile Sınıflandırılması. Koşuyolu Kalp Dergisi, 15(2), 45-50.

- Korürek, M., **Yuksel, A.**, Dokur, Z., & Ölmez, T. (2010). Dimension reduction by a novel unified scheme using divergence analysis and genetic search. Digital Signal Processing, 20(6), 1535-1546.

- Korürek, M., Yildiz, M., & **Yuksel, A.** (2010). Simulation of normal cardiovascular system and severe aortic stenosis using equivalent electronic model/Normal kardiyovasküler sistem ve ciddi aort darliginin bir elektronik devre esdegeri ile benzetimi. Anadulu Kardiyoloji Dergisi: AKD, 10(6), 471.

- Korürek, M., **Yuksel, A.**, Iscan, Z., Dokur, Z., & Ölmez, T. (2010). Retrospective correction of near field effect of X-ray source in radiographic images by using genetic algorithms. Expert Systems With Applications, 37(3), 1946-1954.

- Iscan, Z., **Yuksel, A.**, Dokur, Z., Korürek, M., & Ölmez, T. (2009). Medical image segmentation with transform and moment based features and incremental supervised neural network. Digital Signal Processing, 19(5), 890-901.

- **Yuksel, A.**, & Olmez, T. (2009). Automatic segmentation of bone tissue in x-ray hand images. In Adaptive and Natural Computing Algorithms (pp. 590-599). Springer Berlin Heidelberg.

- Korurek, M., Yildiz, M., & **Yuksel, A.** (2008). Modeling of the arterial compliance in the different stage of primary hypertension using equivalent electronic system. Atherosclerosis Supplements, 9(1), 235.

- **Yuksel, A.**, Dokur, Z., Korurek, M., & Olmez, T. (2008). Modeling of inhomogeneous intensity distribution of X-ray source in radiographic images. In 2008 23rd International Symposium on Computer and Information Sciences.