Comparison and Development of Algorithms for Motor Imagery Classification in EEG-based Brain-Computer Interfaces

James W. Ailsworth Jr.

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Mechanical Engineering

Andrew J. Kurdila, Chair
Alexander Leonessa
Dhruv Batra

April 29, 2016
Blacksburg, Virginia

Keywords: Brain-Computer Interface, Motor Imagery, Common Spatial Patterns, Riemannian Geometry

Comparison and Development of Algorithms for Motor Imagery Classification in EEG-based Brain-Computer Interfaces

James W. Ailsworth Jr.

(ABSTRACT - Academic)

Brain-computer interfaces are an emerging technology that could provide channels for communication and control to severely disabled people suffering from locked-in syndrome. It has been found that motor imagery can be detected and classified from EEG signals. The motivation of the present work was to compare several algorithms for motor imagery classification in EEG signals as well as to test several novel algorithms.

The algorithms tested included the popular method of common spatial patterns (CSP) spatial filtering followed by linear discriminant analysis (LDA) classification of log-variance features (CSP+LDA). A second set of algorithms used classification based on concepts from Riemannian geometry. The basic idea of these methods is that sample spatial covariance matrices (SCMs) of EEG epochs belong to the Riemannian manifold of symmetric positive-definite (SPD) matrices and that the tangent space at any SPD matrix on the manifold is a finite-dimensional Euclidean space. Riemannian classification methods tested included minimum distance to Riemannian mean (MDRM), tangent space LDA (TSLDA), and Fisher geodesic filtering followed by MDRM classification (FGDA).

The novel algorithms aimed to combine the CSP method with the Riemannian geometry methods. CSP spatial filtering was performed prior to sample SCM calculation and subsequent classification using Riemannian methods. The novel algorithms were found to improve classification accuracy as well as reduce the computational costs of Riemannian classification methods for binary, synchronous classification on BCI competition IV dataset 2a.

Comparison and Development of Algorithms for Motor Imagery Classification in EEG-based Brain-Computer Interfaces

James W. Ailsworth Jr.

(ABSTRACT - Public)

Brain-computer interfaces are an emerging technology that could provide channels for communication and control to severely disabled people suffering from locked-in syndrome. It has been found that motor imagery (imagination of body part movement) can be detected and classified (e.g., right hand vs. left hand) from EEG signals. The motivation of the present work was to compare several existing algorithms for motor imagery classification in EEG signals as well as to test several novel algorithms.

The performance of the algorithms was evaluated on a pre-recorded dataset available online on a two-class classification task: right hand vs. left hand motor imagery. The novel algorithms were found to improve the classification accuracy of the existing methods. Furthermore, it was found that the novel algorithms reduced the computational costs of several of the existing algorithms.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

Brain-Computer Interfaces (BCIs) are an emerging technology that could provide channels for communication and control to severely disabled people suffering from locked-in syndrome. In this condition, the person retains full cognitive capabilities, but loses all voluntary muscle control rendering movement and communication impossible. Causes of locked-in syndrome include amyotrophic lateral sclerosis (ALS), brain stem stroke, and spinal cord injury. These and other diseases that disrupt the neural pathways controlling the muscles or impair the muscles themselves are a major cause of disability worldwide affecting nearly two million people in the U.S. alone. [48] BCIs are devices that can measure a person's brain activity and convert it into communication and control signals to be used by an external device. Thus, BCI technology holds promise in restoring motor function or communication to severely disabled individuals, as well as enhancing function in healthy individuals.

One method that has been used by researchers over the years is motor imagery. In motor imagery, a user is asked to imagine the movement of a body part such as their left hand or right hand. Using BCI technology, researchers are able to determine the body part that the user is imagining moving from his/her EEG signals. Using this approach can allow users to control a variety of external devices.

The most popular method of classifying motor imagery from EEG signals is the so-called Common Spatial Patterns and Linear Discriminant Analysis (CSP+LDA) method. [6] In this method, EEG signals that have been band-passed filtered at the relevant frequencies are spatially filtered with CSP spatial filters and log-variance features are extracted and used to train an LDA prediction function. In the testing phase, the unknown attended body part can then be predicted using band-pass filtering, CSP spatial filtering, log-variance feature extraction, and input of the features into the LDA prediction function.

An alternative approach has been recently developed by Barachant and his colleagues which avoids the spatial filtering step and instead aims to merge spatial filtering and classification into one unique step using concepts from Riemannian geometry. The basic idea is to treat

sample spatial covariance matrices in their natural space, a Riemannian manifold. Using the Riemannian concept of the tangent space, Barachant et. al have developed several methods that improve BCI classification accuracy from the CSP+LDA method in both the binary [1] and multiclass [2] case. The most significant drawback to Riemannian geometry approaches is high computational costs leading to long computation times which is not desirable for real-time applications.

The purpose of the present study will be to compare the performance of these two approaches and then to propose an alternative approach that combines them in a novel way. A modest improvement in classification accuracy was observed in the proposed methods. Furthermore, a significant improvement in the computation time of the Riemannian geometry methods was achieved.

Chapter 2

Literature Review

2.1 Historical Perspective

The origins of BCIs can be traced back to 1929 when German scientist Hans Berger [4] first recorded the human brain's electrical activity from electrodes placed on the scalp, the world's first EEG. The first true BCI was described by Dr. Grey Walter who in 1964 connected electrodes directly to the motor neurons of a patient's brain and recorded the brain's electrical activity while the patient pressed a button to advance a slide projector. Dr. Walter then connected the system so that the slide projector would advance whenever the brain signals indicated that the patient had pressed the button. Dr. Walter found that he had to add a time delay because the slide projector started to advance before the button had actually been pressed. [18] In 1969, Eberhard Fetz [15] used operant conditioning to demonstrate that a monkey could learn to move the needle of an analog meter by changing the firing rate of a single cortical neuron. In 1973, Jacques Vidal [46] described his work to identify event-related potentials (ERPs) from EEG data and coined the term "brain-computer interface." Then in 1977, Vidal [45] reported using visually evoked potentials (VEPs) that detected eye gaze direction from single ERP epochs to allow a subject to navigate a virtual maze in real-time.

Elbert et al. [13] in 1980 showed that people could use slow cortical potentials recorded with EEG to control the vertical position of a virtual rocket ship. In 1988, Farwell and Donchin [14] showed that people could use EEG-recorded P300 ERPs to control a spelling program. In 1991, Wolpaw and colleagues [52] used EEG recorded over the sensorimotor cortex which allowed trained subjects to modulate the amplitude of their sensorimotor rhythms (SMRs) to move a cursor in one dimension to a target at the top or bottom edge of a video screen. Subsequently, several researchers [29, 30, 34] extended one-dimensional SMR-based cursor control by developing systems that allowed users to select from more than two targets. BCIs that utilize sensorimotor rhythms were the focus of the present study and will be the focus

of the remaining paper.

2.2 Applications of EEG-based Brain-Computer Interfaces Using Sensorimotor Rhythms

BCIs utilizing sensorimotor rhythms (SMRs) are based on the idea that observable changes in SMR patterns occur with motor imagery and actual movement. Although BCIs using SMRs typically rely on imagery, how exactly they do this depends on the BCI system and the application. In one approach, imagery is only used as a starting point, and users gain control of the application while a continuous adaptation between the user to the BCI and the BCI to the user occurs. As the process proceeds, imagery becomes less important and BCI control becomes automatized like a well-learned muscle action.

In a different approach, a BCI may be designed whereby specific imagery and other mental tasks are cued, the associated SMR changes are defined, and those relationships are used throughout BCI operation. Basically, this approach assumes that SMR patterns associated with a particular task do not change with repeated BCI usage, and that any changes can be rectified through recalibration. Thus, this approach does not actively leverage the adaptive capabilities of the user's brain. Whichever approach the researcher chooses to follow, applications of BCIs that utilize SMRs can be roughly divided by communication applications and control applications.

2.2.1 Communication Applications

As discussed in the previous section, one major application of BCI technology is in communication which can become impaired in a variety of disease states. The development of spelling systems that rely on SMRs has been a major research thrust among researchers in the BCI field. These spelling systems allow users to express themselves through the selection of letters, characters, and various other items.

By imagining right hand movement, users could turn the arrow clockwise, and by imagining right foot movement, users could increase the length of the arrow until the hexagon containing the desired letter was selected. Later in 2009, Friedrich et al. [17] described an application that used an automatic scanning protocol that allowed users to select from one of four possible targets using modulation of their sensorimotor rhythms.

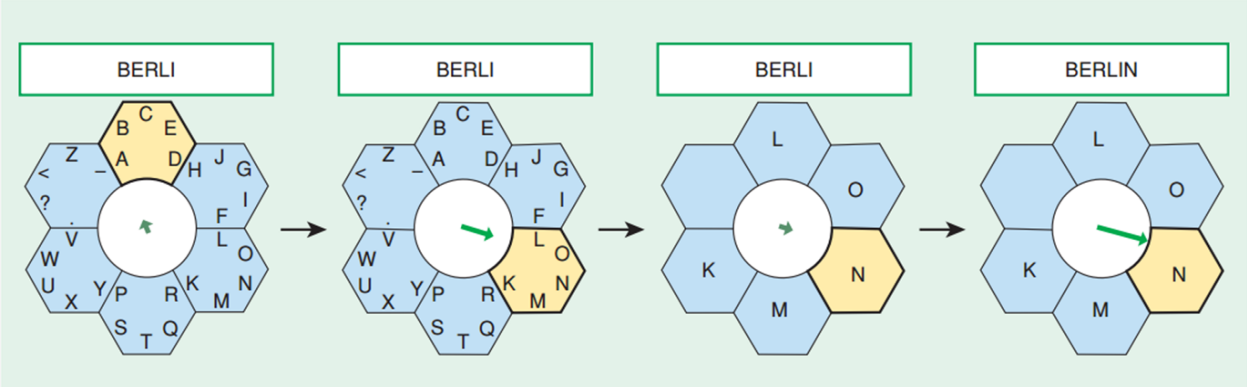

Figure 2.1: Hex-O-Spell. [32] [fair use]

2.2.2 Control Applications

Virtual Applications

Following the one-dimensional cursor control protocols, multiple researchers [8, 20, 50, 51] developed systems that allow users to control two SMR signals simultaneously to achieve two-dimensional cursor control. Expanding on two-dimensional cursor control, McFarland and his colleagues added an SMR-based select function [27] and developed SMR-based three-dimensional cursor control [31].

A second virtual control application for SMR-based BCIs is in navigation of virtual environments. Virtual reality allows researchers to simulate physical applications that might otherwise be too costly, dangerous, or unfeasible at the present. In 2007, Scherer et al. [41] developed a BCI system that allowed users to navigate a virtual environment or use the Google Earth website. Leeb et al. [24] described an experiment in which a person paralyzed with a spinal cord injury was placed with his wheelchair in the middle of a multiprojection-based stereo and head-tracked virtual reality system which allowed him to explore a street populated by 15 virtual people (avatars). By using foot vs. hand motor imagery, the disabled user could move from avatar to avatar stopping to have a brief exchange if the avatar was highlighted with a communication bubble. In 2010, Royer et al. [37] showed that two-dimensional control could be combined with a constant velocity in the forward direction to navigate a virtual helicopter in three-dimensions.

Physical Applications

In addition to virtual applications, there are numerous applications of SMR-based BCIs that can allow a user to actually interact with the physical world. In one potential application, a user could use BCI technology to control his/her environment (e.g. lights, temperature, television, etc.) [48]. An interesting application that first got the author interested in the

field of BCI was the use of SMRs to control robotic systems. LaFleur et al. [22] showed that users could be trained to use SMRs to control the acceleration of a quadcopter having constant forward velocity to fly through rings in a physical, three-dimensional space.

Although SSVEP-based systems have shown promising results in allowing users to control complex robotic devices such as the FRIEND II [18], researchers have also shown interest in incorporating SMR-based control in advanced robotic applications. In 2012, Onose et al. [33] described a robotic system consisting of a seven degree-of-freedom robotic arm in which high-level grasp tasks could be initiated with motor imagery. Two pupil tracking cameras mounted on glasses worn by the user detected gaze direction while two other cameras tracked the position and orientation of the user's head. When gaze was focused onto an object (a standard drinking glass) that the robotic arm could reach and the BCI output from motor imagery exceeded a threshold, the arm would pick up the glass and bring it within about 15 cm of the subject's chin to allow him to drink. The 'place' command to place the glass back onto the table could then be initiated by another BCI activation.

Because the main target users of a BCI system are patients with paralysis or locked-in syndrome, a major potential application is in restoring mobility or motor function. In 2000, Pfurtscheller et al. [35] described a BCI-controlled hand orthosis that opened and closed a tetraplegic user's left hand (figure 2.2) in response to foot motor imagery and right

Figure 2.2: Left: tetraplegic patient with portable BCI hardware. Right: Electrically driven hand orthosis on the patient's left hand. [35] [fair use]

hand motor imagery, respectively. The BCI system was later combined with the Freehand, an implanted neuroprosthesis that uses functional electrical stimulation (FES) of the hand muscles, to restore the paralyzed user's grasping capability [36].

In an effort to restore mobility to the severely disabled, Hema et al. [19] developed a motor imagery-based wheelchair that classified four mental states (move forward, turn left, turn right, and stop) to allow users to navigate around obstacles in a 7×5 m indoor environment. A different approach in restoring mobility to patients with spinal cord injury was taken by researchers at UC Irvine. In 2013, Do et al. [10] reported the development of a BCI-controlled robotic gait orthosis (RoGO) (figure 2.3). By alternating between an idling mental state and

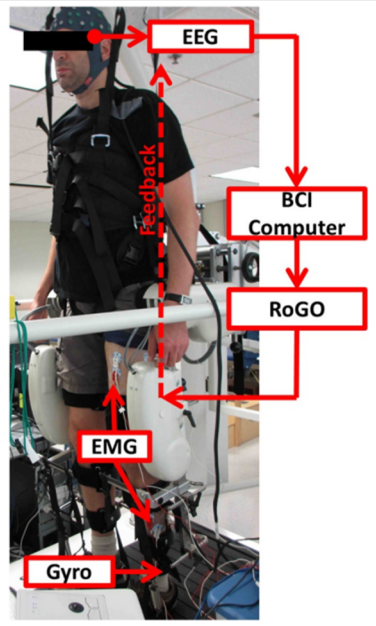

Figure 2.3: Robotic gait orthosis (RoGO) system. [10] [fair use]

walking kinesthetic motor imagery, a healthy user and a user with a spinal cord injury were able to walk on a treadmill with the RoGO.

Chapter 3

Brain-Computer Interface Theory

The basic brain-computer interface chain can be seen in figure 3.1. The process consist of recording a set of signals, preprocessing (to remove artifacts, temporally filter, spatially filter, etc.), extraction of feature vectors, and translation of feature vectors (using machine learning classification algorithms or regression) into useful device commands. A typical set up might be as follows: a person's EEG is recorded while they are presented with a series of cues to imagine moving either the left hand or right hand. The EEG data is marked where those cues are presented with the known label of the cue (i.e., left or right). This data constitutes what is known as the training data because the true label is assumed to be known. The training data is then preprocessed, features useful for discriminating the two classes are extracted, and a machine learning algorithm is trained. The experiment is then repeated, but this time it is assumed that we do not have knowledge of the true labels and

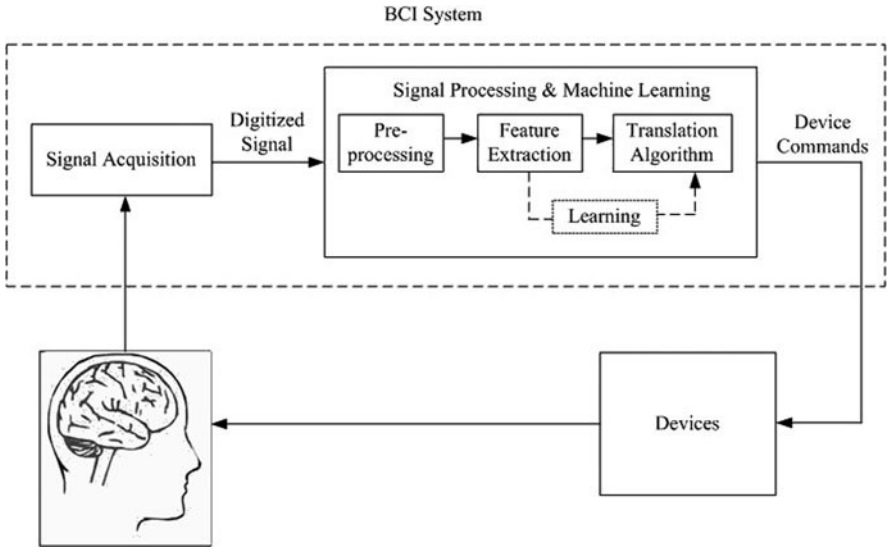

Figure 3.1: Basic BCI chain. [25] [fair use]

that we must predict them using our translation algorithm learned from the training data. This data is called the testing data. To quantify how well the prediction algorithm is doing, we calculate the classification accuracy by comparing the true labels of the testing data to our predicted labels and calculate the proportion of labels that are predicted correctly.

3.1 Brain Sensors and Signals

BCIs can use electroencephalography (EEG), electrocorticography (ECoG), the activity of single neurons, local field potentials, magnetoencephalography (MEG), positron emission tomography (PET), functional magnetic resonance imaging (fMRI), or functional near-infrared (fNIR)imaging to measure a person's brain activity. [48] While ECoG-based BCIs have shown great promise in both monkey and human subjects [9, 44], they involve an invasive surgical procedure to implant an electrode array into the subject's cerebral cortex. MEG, PET, fMRI and fNIR imaging all require expensive equipment and are technically difficult to acquire. Further, PET, fMRI, and fNIR imaging depend on blood flow which leads to long time constants rendering them unsuitable for communication applications. EEG has the advantages that it is non-invasive, relatively inexpensive, easy to acquire in most environments, portable, and characterized by relatively short time constants (has high temporal resolution). EEG has several disadvantages as compared to the more invasive ECoG such as lower spatial resolution (cm vs. tenths of mm), lower bandwidth (0-50 Hz vs. 0-500 Hz), and higher susceptibility to artifacts from muscle activity and eye movement. [39] Despite the drawbacks of EEG, it currently appears to be the most suitable modality for a practical BCI system.

EEG measures the electrical fields naturally produced in the brain as a result of changes in membrane potentials of CNS synapses, neurons, and axons as they send and receive information. These changes in membrane in potential occur as a result of the flow ions across the neuronal membrane. [49] Using EEG, the electric fields can be recorded by electrodes placed on the scalp as voltage waveforms on the order 10-20 μV. [39] The two classes of brain activity most relevant to BCI systems are event-related potentials (ERP) and oscillatory processes.

ERPs show up as voltage changes that are time-locked to a particular stimulus. ERPs include P300, visual evoked potentials (VEPs), and slow cortical potentials. The P300 (or "oddball") potential is generated when an auditory, visual, or touch stimulus that is infrequent, desirable, or significant is presented. Donchin and colleagues [11] used the P300 potential to train subjects to control a spelling program which consisted of a 6x6 matrix of letters, numbers, and characters. The rows and columns would flash rapidly and randomly, and depending upon the row and column that elicited a P300 potential, the researchers could determine the desired character. Steady-state VEPs (SSVEPs) can be modulated when a user looks at one of several targets with different properties (e.g. flash rate). The BCI system can then determine the desired target from the properties of the SSVEP generated. SCPs are

the slowest features used in EEG-based BCIs systems to date. Negative SCPs accompany mental preparation while positive SCPs accompany mental inhibition. Birbaumer and his colleagues have shown that subjects can control SCP amplitude to control cursors, a basic word-processing program, and simple control tasks. [47]

The focus of the present study will be on brain-computer interface systems that use oscillatory brain activity originating in the sensorimotor cortex. Rhythmic activity in this area of the brain is modulated by actual movement, motor intention, and motor imagery. The changes typically observed are attenuation in power (event-related desynchronization (ERD)) in the alpha band (8-13 Hz, also known as the mu rhythm) and the beta band (14-26 Hz) as well as an increase in power (event-related synchronization (ERS)) in the gamma band (>30 Hz). Collectively, these rhythms are known as sensorimotor rhythms (SMRs). [53] Thus, the mu and beta sensorimotor rhythms can be thought of as idle rhythms that are desynchronized when a subject engages in motor tasks.

The motor cortex and somatosensory cortex posses the property of cortical homunculus in which particular parts of the body are mapped onto particular regions of the sensorimotor cortex as seen in figure 3.2. In the case of right or left hand motor imagery, the ERD of the

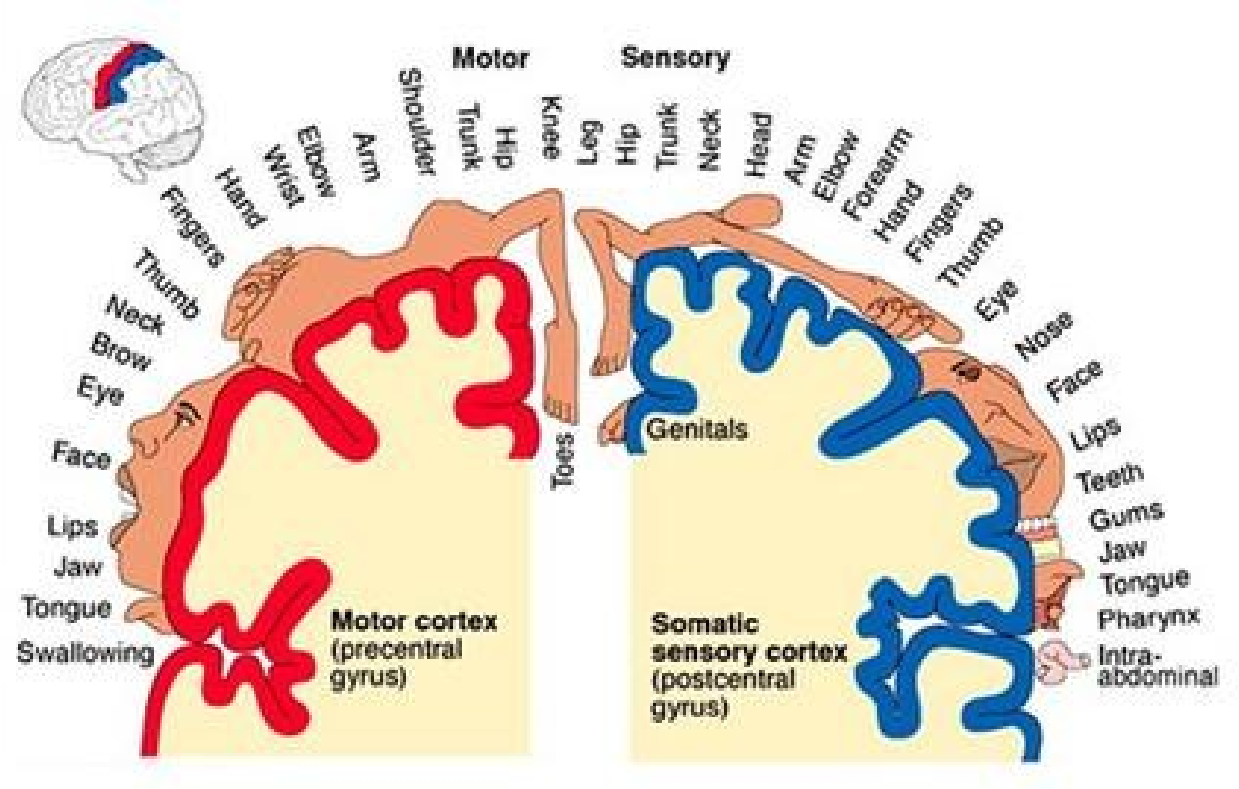

Figure 3.2: Cortical homunculus. [26] [fair use]

mu and beta rhythms is observed contralateral to the hand of interest as shown in figure 3.3. To produce ERD/ERS patterns that can be detected, the cortical areas involved have

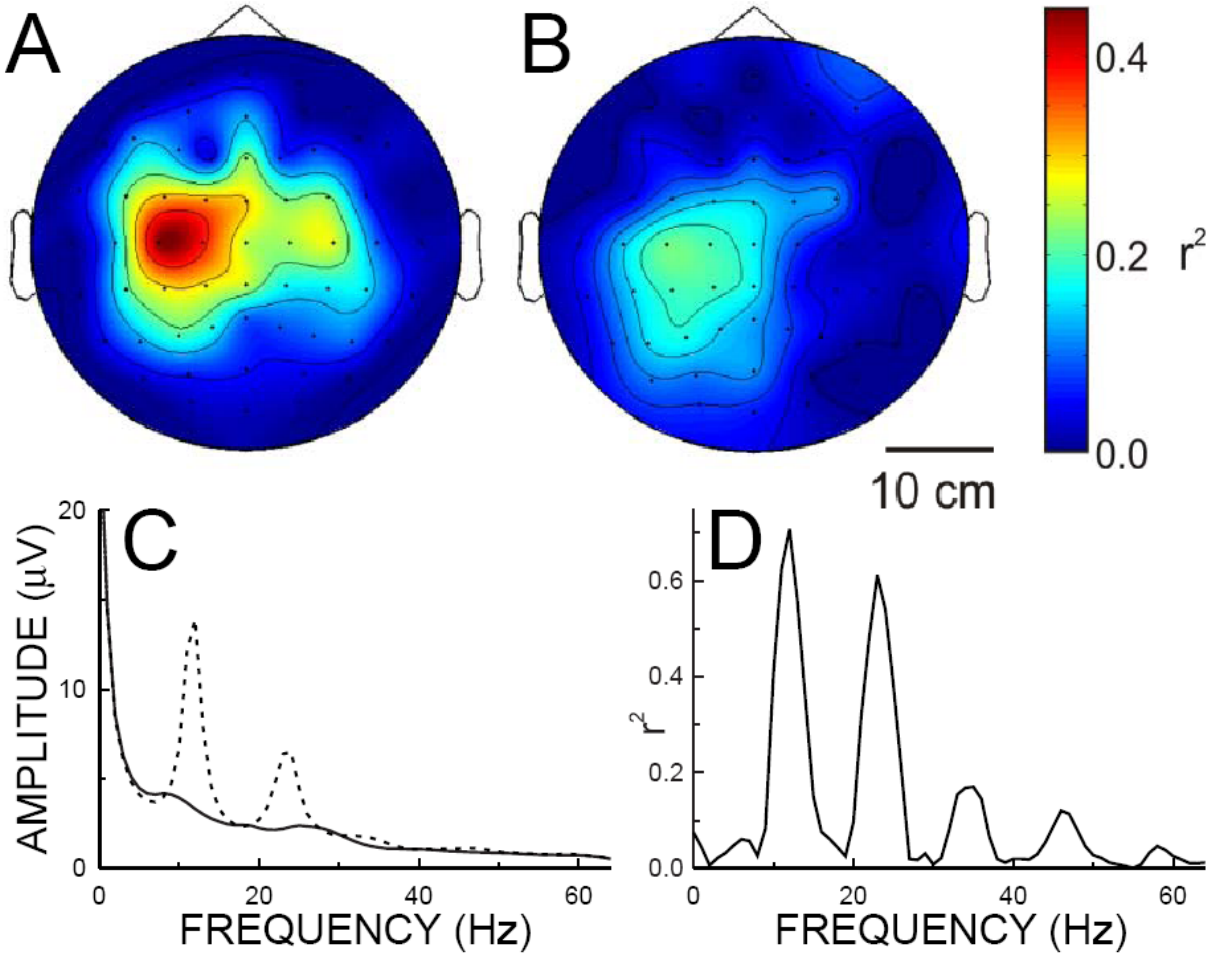

Figure 3.3: (A) and (B): r^2 (proportion of single-trial variance due to the task) scalp topography showing the difference between actual (A) and imagined (B) right hand movement and rest for a 3 Hz band centered at 12 Hz. (C) Amplitude spectrum for a different subject recorded from electrode C3 comparing right hand motor imagery (solid line) to rest (dashed line). (D) The corresponding r^2 spectrum. [39] [fair use]

to be large enough so that the resulting activity is sufficiently prominent compared to the remaining EEG (background EEG). In general, it has been found that the left hand, right hand, both feet, and tongue are comparatively large and topographically different enough to be distinguished in an EEG-based BCI system. [18]

3.2 Data Acquisition

3.2.1 Recording the EEG

l [39]. A fabric, elastic cap with the electrodes already embedded can greatly simplify this process. The standard electrode naming and positioning scheme for EEG applications follows the 10-20 international system (figure 3.4), so named because the most commonly used electrodes are positioned 10, 20, 20, 20, 20, and 10 percent of the total nasion-inion distance [18]. This standard was subsequently extended to more than 70 electrodes. Additionally, a reference electrode to which the voltage at all of the other electrodes is measured with respect to (unipolar rather than bipolar) and a ground electrode are usually placed on the earlobe or mastoid.

3.3 Preprocessing

Signal conditioning, or signal preprocessing, is the process by which raw EEG signals are enhanced by removal of artifacts and enhancement of the spatial, spectral, or temporal characteristics of the signal most relevant to the application. Signal conditioning often requires prior knowledge about the relevant characteristics of the raw EEG signal. Signal preprocessing can include the following steps: temporal filtering, data decimation and normalization, spatial filtering, and artifact removal.

3.3.1 Artifacts

There are several different kinds of artifacts that researchers must deal with when working with EEG signals. These include electrical power line interference, eye blinks, eye movements, and muscle activity. Researchers have developed many techniques for dealing with these artifacts.

Electrical power line interference (50-60 Hz at 110-230 Volts) can exceed the EEG signal (50-100 μVolts) by a factor of 2×10^6 (126 dB) [39]. This problem becomes even more relevant when the EEG is recorded outside of specialized shielded rooms. Fortunately, many EEG amplifiers have a built-in notch filter to suppress power-line interference. High electrode impedance or electrode impedance mismatch can reduce the efficacy of the notch filter and can lead to the reappearance of power line interface. [47]

Eye blink artifacts are generated when the subject's eye lids pass over his corneas. A frictional mechanism leads to a charge separation which forms electrical dipoles oriented in the up-down direction. [39] This activity is recorded by all of the electrodes, but is most strongly picked up by the front-most electrodes and decays with distance. The artifact shows up in

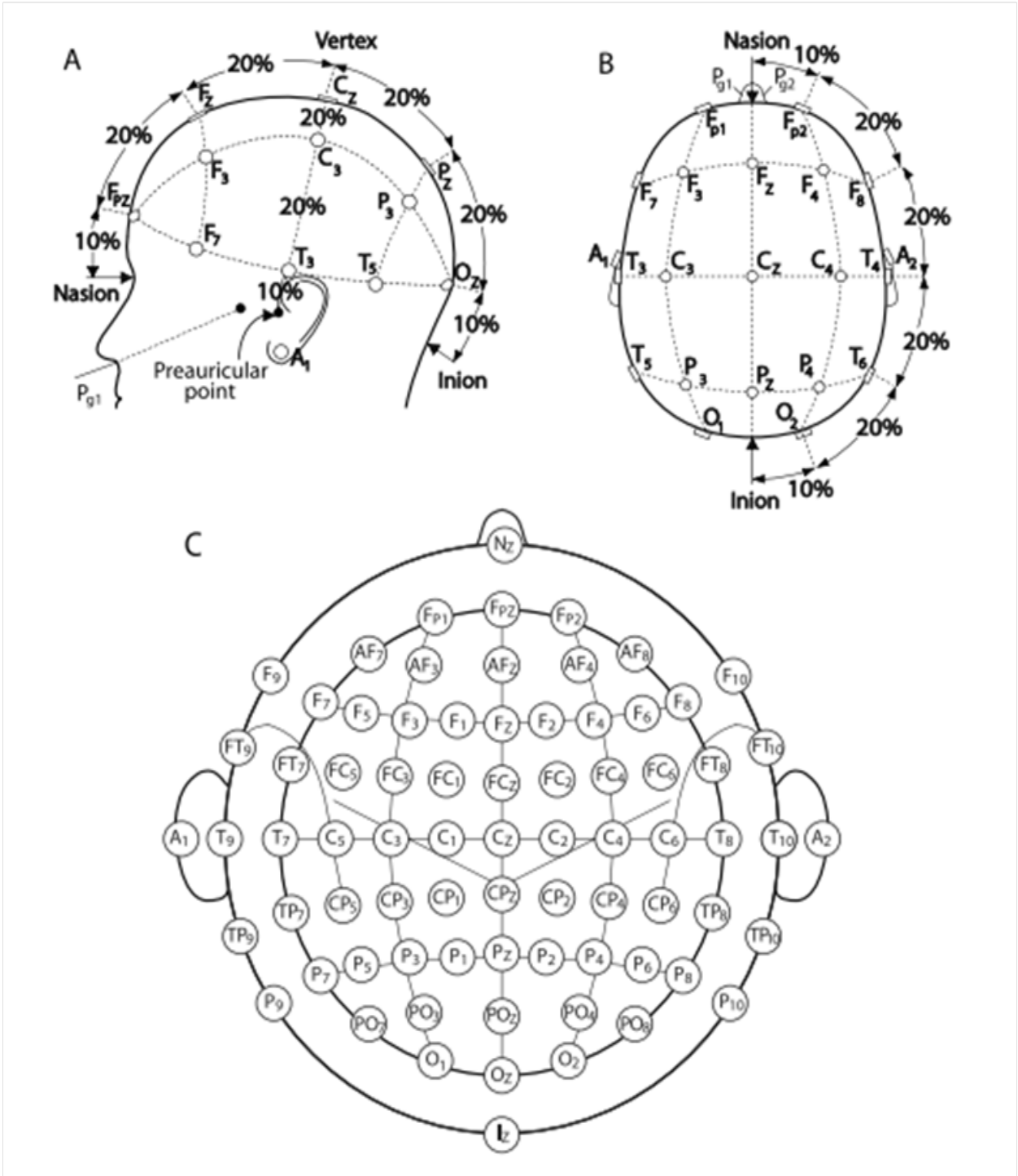

Figure 3.4: (A) and (B): The standard 10-20 system. (C) The extended 10-20 system with more than 70 electrodes. [39] [fair use]

the EEG as a positive peak lasting a few tenths of seconds. The frequency content of this artifact is negligible in the 10 Hz range (relevant for sensorimotor activity), but can affect experiments relying on time-domain amplitude (e.g. averaged P300 waveforms).

Electrooculograhic (EoG) activity generated from eye movement can also contaminate the EEG. Similar to eye blink artifacts, friction against the cornea and retina leads to the generation of dipoles. The artifact can be recorded symmetrically or asymmetrically depending upon if the eye movement was vertical or horizontal, respectively. EoG artifacts tend to be similar to eye blink artifacts except they are at a lower frequency and higher amplitude.

Perhaps the most significant challenge of all EEG artifacts is from muscular (electromyographic, EMG) activity. Minimization of EMG activity must be checked at the beginning of the recording and verified throughout. The frequency distribution of EMG signals are very broad, and they can have a profound effect on the amplitudes in the mu and beta bands. The most common sources of EMG artifacts are the muscles that lift the eyebrows and close the jaw. The jaw closing artifact can be minimized by having the subject keep his mouth open slightly or instructing him to place his tongue between his foreteeth.

Eye-blink and EOG artifacts often occur at low frequencies (1 Hz). Thus, these artifacts can often be reduced using a high-pass or bandpass filter. Reduction of eye blink, EOG, and EMG artifacts can often be accomplished through the process of spatial filtering. The spatial filter weights can be derived from a regression procedure using data from fronto-temporal electrodes recording EMG activity or electrodes next to the eyes recording EOG activity. [47]

3.3.2 Temporal Filtering

Temporal filtering is often used to isolate the brain activity in the frequency range of interest for a given application. Depending on the hardware used, this filtering can take place before or after digitization.

EEG frequencies greater than 40 Hz typically have very low SNR's because the EEG signal power drops off with 1/frequency and the signal is attenuated by the skull and scalp tissue. Further, low-frequency signal drift may occur and can lead to reduced amplitude resolution when ADC with limited amplitude range is performed. Therefore, it is common practice to first apply a 0.5-40 Hz (or narrower) bandpass filter to the raw EEG data to isolate useful brain activity.

For BCI's based on sensorimotor activity, an 8-12 Hz bandpass filter can be used to isolate the mu-band activity or a 8-30 Hz bandpass filter can be used to isolate mu- and beta-band activity. For BCI's using event-related potentials, a bandpass filter with a high-pass cutoff of 0.1-0.5 Hz and a low-pass cutoff of 5-40 Hz is often used. In general, the width of the filter is often set conservatively so as to not lose information relevant for the given application. [47]

3.3.3 Data Decimation and Normalization

Data decimation is the process by which the effective sampling rate is reduced by a periodic elimination of samples (downsampling) for more efficient processing and storage. Data decimation is often done when the original sampling rate is higher than the Nyquist rate to capture the relevant brain signals. The data decimation process also implies a low-pass filtering operation to avoid aliasing from higher frequencies. According to the Nyquist criterion, the cutoff frequency of the low-pass filter should be chosen to equal one-half the decimated sampling frequency.

When comparing signals with different mean values or dynamic (i.e., amplitude) ranges that are not relevant to a particular application, the process of data normalization will often be used. Most commonly a signal will be normalized to have zero mean and unit variance by subtracting from each sample the mean of the signal and dividing each sample by the variance of signal. An example of when data normalization may be useful is when comparing the signals from two different brain areas where the amplitude ranges may be very different but the difference in signal dynamics is of primary interest. Data normalization may also be used to correct for unintended electrode impedance differences. It should be cautioned that data normalization does eliminate amplitude differences within a set of signals which may be useful for a given application. [47]

3.3.4 Spatial Filtering

Spatial filtering is a crucial step in the signal processing chain for motor imagery BCIs. Spatial filtering can reduce the dimensionality of the data, increase its spatial resolution, increase the discriminability of classes, etc.

When measuring brain activity with EEG, the quantity of interest is the potential difference (in volts) between two electrodes. The signal obtained is referred to as a channel. The sensitivity of a channel to different brain sources depends on the size and orientation of the source in relation to the location of the two electrodes. Therefore by properly selecting pairs of electrodes, it is possible to make a channel more sensitive to certain brain sources and less sensitive to others. If all of the channels have been recorded in reference to a single electrode (e.g., the reference electrode), then it is possible through weighting and combining channels to reconstruct any desired alternative set of channels after digitization. This is the idea behind spatial filtering.

When spatial filter matrices are multiplied by the data, a new set of channels and time series are created. Let $x(t) \in \mathbb{R}^{N\times 1}$ be the N channel EEG signal at time t in the original sensor space. The matrix $W \in \mathbb{R}^{N\times J}$ projects this signal into a surrogate sensor space generating a spatially filtered signal $z(t) \in \mathbb{R}^{J\times 1}$:

$$z(t) = W^T x(t)$$

Thus, each channel of $z(t)$ is a weighted sum of the original N channels. Each column of W, w_j, is called a spatial filter and specifies the weight of n^{th} channel. Consider now the EEG data $X \in \mathbb{R}^{N \times M}$ consisting of N channels of M time samples. The spatially filtered data $Z \in \mathbb{R}^{J \times M}$ can be calculated as:

$$Z = W^T X$$

$$\begin{bmatrix} z_{11} & z_{12} & \dots & z_{1M} \\ z_{21} & z_{22} & \dots & z_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ z_{J1} & z_{J2} & \dots & z_{JM} \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1J} \\ w_{21} & w_{22} & \dots & w_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M1} & w_{M2} & \dots & w_{MJ} \end{bmatrix}^T \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1M} \\ x_{21} & x_{22} & \dots & x_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & z_{NM} \end{bmatrix}$$

Note that $J \leq N$.

There are several commonly used methods for generating spatial filters. These can be divided into two distinct categories: data-independent spatial filters and data-dependent spatial filters.

Data-independent spatial filters

Data-independent spatial filters use fixed geometric relationship instead of data to generate spatially filtered channels. These ad hoc filters make assumptions about the spatial distribution of the underlying brain sources and are not guaranteed to optimal. Still, several researchers (e.g. The Wadsworth Group) have implemented data-independent spatial filters with considerable success.

Often, researchers using data-independent spatial filters will only be interested in using features derived from one to a few spatially filtered channels online. Thus, the spatial filter matrix used online may be as small as dimension $N \times 1$. However, for offline analysis, the spatial filter matrix is often set to be $N \times N$ to generate what are called topo-plots, in which the difference between task conditions (usually measured as r^2) of each spatially channel is considered. Figure 3.5 displays the characteristics of three commonly used data-independent spatial filters: common-average reference, and small and large surface Laplacians.

Common-average reference spatial filters

The idea of common-average reference (CAR) spatial filters is to take a set of signals that have been recorded with a common reference and to re-reference them to eliminate the electrical activity that is common to all electrodes. This is realized by averaging the voltage at each electrode at each time point ($\frac{1}{N}\sum_{i=1}^{N} x_i(t)$) and subtracting that average from the signal at the electrode of interest at each time point $x_n(t)$:

$$z_n(t) = x_n(t) - \frac{1}{N}\sum_{i=1}^{N} x_i(t)$$

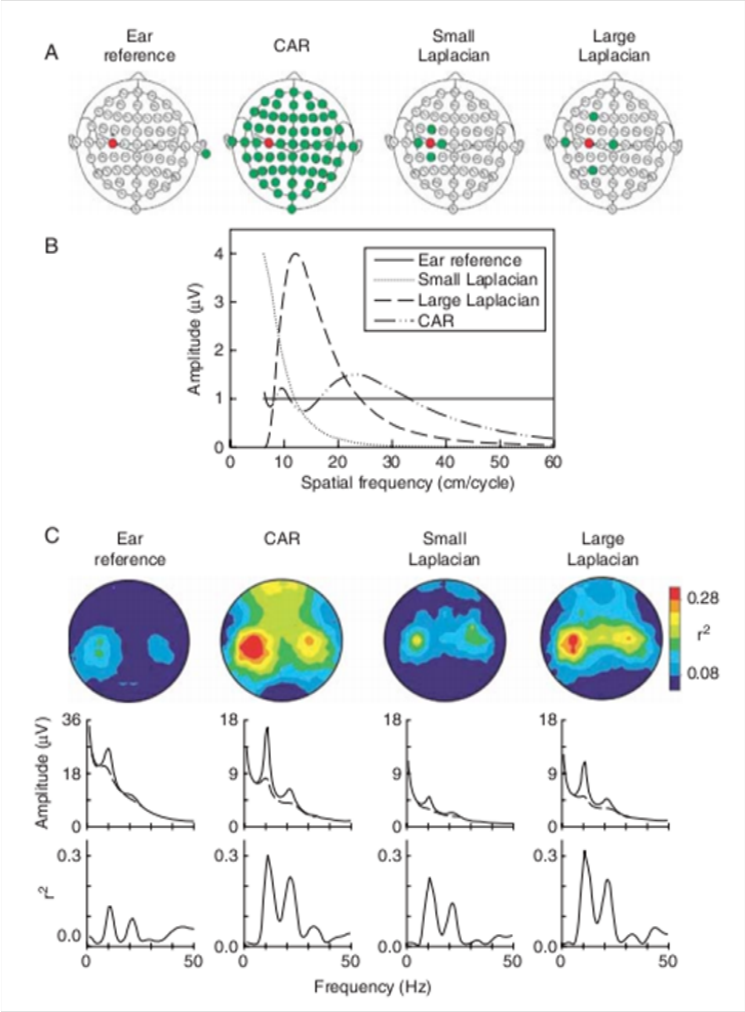

Figure 3.5: Comparison of four different spatial filters used for online cursor control while well-trained subjects moved a cursor to top or bottom targets. Analysis done offline. (A) Electrode locations (green) used by filters for EEG data recorded from electrode C3 (red). (B) Spatial bandpass. The square-root of the RMS amplitude of a signal that varies sinusoidally over the scalp from 6 cm (twice the interelectrode distance to avoid spatial aliasing) to 60 cm (the head circumference). (C) Top: average r^2 topography at the frequency used online. Bottom: average amplitude and r^2 spectrum from the features derived from the electrode used online (C3). r^2 is defined to be the proportion of the feature variance that is accounted for by target location. [47] [fair use]

CAR spatial filters are also useful for reducing artifacts that are common across all electrodes (e.g. 60 Hz power-line noise).

Suppose we are interested in deriving one CAR spatially filtered channel centered at electrode C_3 from a 64 channel EEG recording. The corresponding CAR spatial filter would be:

$$W^T = \begin{matrix} F_{PZ} & \dots & FC_5 & FC_3 & C_5 & C_3 & C_1 & CP_3 & CP_1 & \dots & O_Z \end{matrix}$$
$$[-\tfrac{1}{63} \quad \dots \quad -\tfrac{1}{63} \quad -\tfrac{1}{63} \quad -\tfrac{1}{63} \quad +1 \quad -\tfrac{1}{63} \quad -\tfrac{1}{63} \quad -\tfrac{1}{63} \quad \dots \quad -\tfrac{1}{63}]$$

Multiplying this 1×64 spatial filter by $64 \times M$ EEG data generates $1 \times M$ spatially filtered

EEG data.

Some researchers using data-independent spatial filters will use CAR spatial filters in an initial session to discover which channels and frequencies are most relevant for distinguishing between task conditions (r^2) [39]. Thus a CAR filter will be applied to each electrode so that the number of spatially filtered channels is the same as the number of original channels ($J = N$).

Surface Laplacian spatial filters

A surface Laplacian spatial filter is a discretized approximation of the second spatial derivative of a two-dimensional Gaussian distribution on the scalp surface. It attempts to invert the process that blurred the brain signals on the scalp. [39] At each time point, a weighted sum of the four nearest (small Laplacian) or next nearest (large Laplacian) electrodes is subtracted from the electrode of interest:

$$z_n(t) = x_n(t) - \sum_{i \in x_i} w_{n,i} x_i(t)$$

where w_i is a function of the distance $d_{n,i}$ between the electrode of interest n and its ith neighboring electrode:

$$w_{n,i} = \frac{\frac{1}{d_{n,i}}}{\sum_{i \in x_i} \frac{1}{d_{n,i}}}$$

In practice, researchers often simplify this calculation by subtracting the mean of the four nearest or next nearest neighboring electrodes from the central electrode of interest:

$$z_n(t) = x_n(t) - \frac{1}{4} \sum_{i \in x_i} x_i(t)$$

Spatially adjacent channels tend to be highly correlated since their positions are similar relative to the underlying brain sources. So the Laplacian spatial filter attempts to increase the spatial resolution of the channel by eliminating correlated activity. The choice of whether to use a small or large Laplacian spatial filter depends on the spatial characteristics of the brain activity of interest.

Suppose we were interested in deriving one small Laplacian filtered output channel centered at electrode C_3 in a 64 channel EEG recording. The corresponding small Laplacian spatial filter would be:

$$W^T = \begin{array}{c} \begin{matrix} F_{PZ} & \dots & FC_5 & FC_3 & C_5 & C_3 & C_1 & CP_3 & CP_1 & \dots & O_Z \end{matrix} \\ \begin{bmatrix} 0 & \dots & 0 & -0.25 & -0.25 & +1 & -0.25 & -0.25 & 0 & \dots & 0 \end{bmatrix} \end{array}$$

Multiplying this 1×64 spatial filter by $64 \times M$ EEG data generates $1 \times M$ spatially filtered EEG data.

To generate topo-plots offline for the Laplacian spatial filters, we use an $N \times N$ matrix W to subtract the average voltage of the nearest or next next nearest neighboring electrodes from each electrode. For edge electrodes, neighboring electrodes can be selected as suggested by Zhou. [28]

Data-dependent spatial filters

Data-dependent spatial filters are derived directly from the user's training data. They tend to be more complex in terms of spatial geometry and derivation than data-independent spatial filters, but can lead to better results for a particular application. They are especially useful when the exact characteristics of the relevant brain activity are not known. Spatial filters are derived by placing an objective constraint on the matrix W. The constraints on W are typically designed to produce fewer and/or more meaningful channels Z. Three commonly used methods are principle component analysis, independent component analysis, and common spatial patterns.

Principle component analysis

Principle component analysis (PCA), as well as independent component analysis (ICA), separates EEG signals into components. Some of these components may be of interest, such as those associated with motor imagery, while others may not, such as noise. Therefore, PCA and ICA are often used to separate useful components from noise signals.

PCA is a pre-processing method that performs an orthogonal linear transformation to a new coordinate system in which the spatially filtered signals (components), are uncorrelated and ordered according to decreasing variance. Thus, the first component contains most of the variance of the original signals, the second component contains the second most variance, etc. [25]

Let $X \in \mathbb{R}^{N \times M}$ be N channel EEG data in the original sensor space where each channel has been normalized to have zero-mean, and let $P = XX^T$ be the sample spatial covariance matrix, $P \in \mathbb{R}^{N \times N}$. Note: for an unbiased estimator of the true spatial covariance matrix, P is normalized by $T_s - 1$ where T_s is the number of time samples; this normalization factor will not effect the relevant calculations. The PCA transformation matrix $W = [w_1, \ldots, w_n]$ is then calculated using a generalized eigenvalue decomposition of P, where the spatial filters, or principle components, $w_1, \ldots, w_n$ are the n normalized orthogonal eigenvectors of P ordered according to their decreasing eigenvalues $\lambda_1, \ldots, \lambda_n$. The PCA transformation of X is then:

$$Z = W^T X$$

If one includes all of the principle components in W, then W is a square matrix $W \in N \times N$. However, it is often desirable retain a subset of principle components such that Z contains only relevant brain signals, i.e. those with high variance. Typically only the first few principle components, which capture the most of the variance of the channels in X, are retained while the remaining principle components are discarded. This greatly reduces the dimensionality of the data so that Z contains only a few channels. Ideally, this process discards little relevant information, although this is not guaranteed.

A drawback of PCA is that the spatially filtered signal with high variance may not be associated with the task condition. For example, if X is corrupted with significant 60-Hz power line noise, it is likely that this noise will show up as one of the top components. Therefore, a comparatively low-power channel that is highly correlated with the task condition may be inadvertently discarded. The common spatial patterns method, which will be discussed in a later section, was developed to address this problem.

Independent Component Analysis

Although the output channels generated from PCA are uncorrelated, they are not independent. For statistical independence, the joint CDF $F_{X,Y}(x, y) = Pr[X \leq x, Y \leq y]$ of channels x and y must satisfy: $F_{X,Y}(x, y) = F_X(x)F_Y(y)$. Consequently, the correlation of channels x and y can be written as $E[XY] = E[X]E[Y]$. Thus, requiring independence of the components is a stricter constraint than uncorrelatedness. The goal of ICA is find a matrix W that produces independent channels in Z. [47]

ICA makes the assumption that particular sources of brain activity are localized and function independently. Therefore, independent channels in Z are desired because, theoretically, you are more likely find channels that are sensitive to different signal-generating sources in the brain. ICA can thus be thought of as a solution to the blind source separation problem sometimes described as the "cocktail party effect," whereby many people talk simultaneously in a room but a person must pay attention to just one speaker. Brain signals can be thought of as having a similar cocktail party effect because the signals arise from many neurons, and the signals are mixed and then aggregated at the electrode. Therefore, the actual brain sources and mixing are unknown. [25]

Mathematically, we assume that there are n mutually independent, unknown sources $s_1(t), \ldots, s_n(t)$ in the brain signals denoted $s(t) = [s_1(t), \ldots, s_n(t)]^T$ with zero-mean. Assuming that there are n electrodes such that the sources are linearly mixed to produce n observable mixtures denoted $x(t) = [x_1(t), \ldots, x_n(t)]^T$ then:

$$x(t) = As(t)$$

where A is the $n \times n$ time-invariant mixing matrix whose elements must be estimated from the observations. A is often assumed to be full-rank with n linearly independent columns.

The ICA method then computes the demixing matrix W using the observed signal x to compute n independent components: in $z(t)$:

$$z(t) = Wx(t)$$

where $z(t) = [z_1(t), \ldots, z_n(t)]^T$ are the n estimated independent components.

After decomposing the brain signals with ICA, we can remove the irrelevant components and project the data back into the signal space as:

$$\tilde{x}(t) = W^{-1}z(t)$$

Unfortunately, ICA presents several challenges when applied in the BCI field. Not only are the algorithms relatively complex, but the number of independent sources must be accurately estimated to effectively isolate the desired source activity. Furthermore, the number of independent sources must be less than equal to the original number of channels. When the expected number of independent sources is much less than the number of original channels and/or the original channels are highly correlated, it is common to perform PCA prior to ICA to generate fewer, uncorrelated channels to streamline ICA processing.

ICA algorithms include the information maximization (infomax) approach, which maximizes entropy, and the second or higher order statistics based approach. Many of these algorithms are freely available online. A related source separation approach is called sparse component analysis (SCA) which assumes that the number of components is larger than the number of mixtures and that the components can be dependent on one another. [25]

Common Spatial Patterns

Common spatial patterns (CSP) is a spatial filtering method that is closely related to PCA. However, while PCA maximizes the variance of the first component in the transformed space, CSP maximizes the variance ratio of two classes. Therefore, CSP finds the transformation matrix that maximizes the variance of the samples belonging to one condition while simultaneously minimizing the variance of the samples belonging to the other condition.

For CSP, the input channels are first band-pass filtered so that the variance equals the band-power. This makes CSP quite effective for motor imagery BCIs in which we are interested in power differences within certain frequency bands between task conditions. CSP requires that we have the training samples as well as their true class labels. This extra information is not required for PCA or ICA. Therefore, we call PCA and ICA unsupervised techniques and call CSP a supervised technique.

Mathematically, let $X_i \in \mathbb{R}^{N \times K}$ be a short segment of EEG data, a trial, belonging to class $c \in [1, 2]$ with N channels and K samples. Assume that X has already been time-centered by a high-pass (or band-pass) filtering operation and scaled: $X = \frac{1}{\sqrt{K-1}} X_{tc}$ where X_{tc} is the

time-centered trial. Note that time-centering simply makes X zero-mean. We first calculate the class-related mean sample spatial covariance matrices, $P_c \in \mathbb{R}^{N\times N}$ as:

$$P_c = \frac{1}{|I_c|}\sum_{i\in I_c} X_i X_i^T$$

where I_c is the set of indices corresponding to trials of class c and $|I_c|$ is the number of trials of class c.

CSP aims at extremizing the function:

$$J(w) = \frac{w^T P_1 w}{w^T (P_1 + P_2) w}$$

where w is a spatial filter that we are seeking. $J(w)$ is not changed if w is re-scaled, so $J(w) = J(k * w) : k \in \mathbb{R}$. Extremizing $J(w)$ is equivalent to extremizing $w^T P_1 w$ subject to the constraint $w^T(P_1 + P_2)w = 1$ because we can always find a re-scaling of w such that the constraint holds. Using the Lagrange multipliers method, the constrained optimization problem can be solved by extremizing:

$$L(\lambda_1, w) = w^T P_1 w - \lambda_1 (w^T (P_1 + P_2) w - 1)$$

The filters w can be found as:

$$\frac{\partial L}{\partial w} = 2w^T P_1 - 2\lambda_1 w^T (P_1 + P_2) = 0$$

$$=> P_1 w_j = (\lambda_1)_j (P_1 + P_2) w_j$$

which is the generalized eigenvalue problem. This can be solved simply in Matlab with the eig(P_1,$P_1 + P_2$) command. This problem is often formulated as the joint diagonalization of the P_1 and P_2 covariance matrices subject to an equality constraint:

$$W^T P_1 W = D_1$$

$$W^T P_2 W = D_2$$

$$s.t. : D_1 + D_2 = I_N$$

where $(D_1)_{jj} = (\lambda_1)_j$ and $(D_2)_{jj} = (\lambda_2)_j$. Note that, due to the constraint, $(\lambda_1)_j + (\lambda_2)_j = 1$ so that $(\lambda_1)_j = 1 - (\lambda_2)_j$, with $\{(\lambda_1)_j, (\lambda_2)_j\} \in]0 : 1[$ by definition. Thus, a λ_1 close to one implies that the corresponding spatial filter w_j yields high variance for class 1 while simultaneously yielding minimum variance for class 2, and a λ_1 close to zero implies the converse.

Like in PCA, researchers will usually only retain the components that contain useful information. In this case, we will be interested in retaining spatial filters that yield components with a high variance ratio for one or the other of the two classes. Thus, we will be interested

in eigenvectors w_j associated with the highest and lowest eigenvalues, $(\lambda_1)_j$. If we keep too few eigenvectors for W, then we risk not capturing the discriminative activity, and if we keep too many eigenvectors, then we risk overfitting the classifier. In practice it has been found that, in general, retaining six spatial filters: three associated with the three highest eigenvalues and three associated with the three lowest eigenvalues, yields the best classification accuracies.

As with the other spatial filtering techniques, we calculate the spatially filtered EEG data as:

$$Z = W^T X$$

The first row of Z contains most of the variance of class 1 and the last row of Z contains most of the variance of class 2. If we use six spatial filters in $W \in \mathbb{R}^{N \times 6}$, then the dimensionality of Z will be reduced to $6 \times K$. Because the full W is square, we can invert it to find:

$$X = W^{-1} Z$$

The columns of W^{-1} are called forward projections or common spatial patterns. Interestingly, one can determine the location and orientation of the brain sources active in a given condition by examining the plots of forward projections. Figure 3.6 shows six spatial filters: three yielding high variance for condition 1 and three yielding high variance for condition 2, along with their corresponding forward projections.

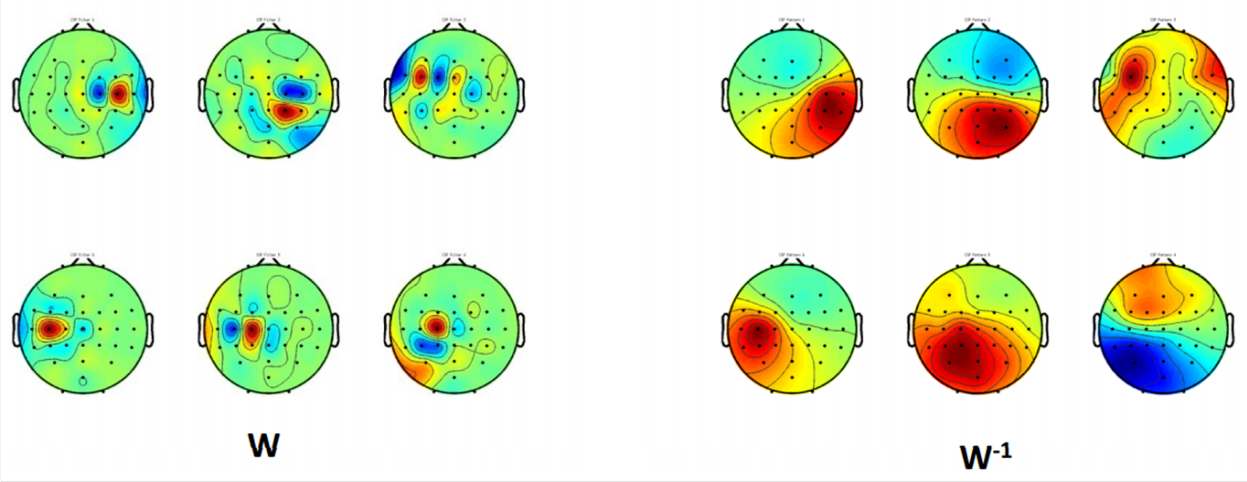

Figure 3.6: 6 spatial filters (left) and 6 forward projections (right) where red corresponds to a positive weight, blue a negative weight, and green a weight near zero. [21] [fair use]

The top three spatial filters yield high variance for the left hand motor imagery condition while the bottom three yield high variance for right hand imagery. Examining the forward projections, we can conclude that the top left and middle sources and bottom left and right sources are likely from tangentially-oriented dipoles residing in cortical sulci, while the top right and bottom middle sources are likely from radially-oriented dipoles residing in cortical

gyri. Note that although similar in appearance, these spatial filter weight plots and forward projection plots are different from the r^2 scalp topographies shown earlier.

An alternative approach suggested by Blankertz et al. [6] uses a criteria based on the eigenvalues to select the spatial filters most useful for discrimination. Spatial filters may be selected which maximize the quantity:

$$|(\lambda_1)_j - 0.5|$$

Thus according to this criteria, eigenvectors with associated eigenvalues closest to 1 or 0 would be selected first. Using this criteria allows more flexibility in spatial filter selection in that it allows designers to select odd numbers of spatial filters which could ultimately improve BCI performance.

3.4 Feature Extraction

Once preprocessing has optimized the signal by enhancing the desired features and reducing the artifacts, the process of feature extraction is performed to extract the raw features that will eventually be used either to train a classifier or regression equation or, in the case of testing data, as an input for these equations. The choice of which features to extract is dictated by the nature of the relevant brain signals as well as the particular BCI application. The brain signals we may be interested in (e.g. sensorimotor activity or evoked potentials) have certain specific spatial, spectral, and temporal characteristics. Therefore, for sensorimotor activity which is characterized by amplitude modulations in specific frequency bands, it is logical to extract frequency-domain features. However, for more exploratory situations, we may be interested in learning about the temporal and spectral characteristics of a brain signal. In this case, we may extract both time-domain and frequency-domain features and use them both to construct feature vectors. However, in the long-run, we will be interested in eliminating unnecessary and redundant features that are not relevant for a particular application. The choice of features should also take into consideration the classifier as feature extraction and feature classification go hand-in-hand. [47]

3.4.1 Block processing

For many BCI applications, it is desirable for processing to occur in real time so that there can be an ongoing interaction between the user and the application. Achieving this goal is the role of block processing. Block processing is the process of segmenting samples into successive sometimes overlapping sample blocks. From the blocks of samples, we generate a feature vector to be fed into a translation algorithm. The translation algorithm accepts the feature vector or sequence of consecutive feature vectors and translates them into a device command or user feedback. For efficient online implementation, the length and overlap of the sample blocks should fit the temporal dynamics of the relevant brain signal, the method

of feature extraction, the application, and the available processing power. As a general rule, one should try to not compute more feature vectors than would be necessary for a particular application. Additionally, for some applications it may be desirable to average consecutive feature vectors before feeding them into a translation algorithm to improve the signal-to-noise ratio. [47]

3.4.2 Temporal features

Peak-picking and integration

Peak-picking is a simple feature extraction method whereby the maximum or minimum amplitude is determined within a sample block and will usually be defined relative to a preceding stimulus. The value and possibly its time of occurrence will be the feature vector for the block. Alternatively, some form of averaging or integration over all or part of the time block can be used. These techniques will often be preferable to simple peak-picking because of variations in the latency of a response to a stimulus and overlying high-frequency, high-amplitude, non-relevant activity.

Correlation and template-matching

The similarity or correlation of a sample block to a template can be used as a feature. Computing the correlation of a response to a template basically involves applying an FIR filter using the template as filter weights. For a given response, segments that are highly correlated with the template will have a high output while those not highly correlated to the template will have a low output. Wavelet analysis can be considered a variation of this technique whereby the template has specific analytical properties that produce a frequency decomposition of the signal. [47]

3.4.3 Spectral Features

For BCI's that operate on oscillatory activity, such as those utilizing motor imagery, we are usually interested in frequency-domain features. Classically, the Fourier transform, which transforms a signal from the time-domain into the frequency domain, has been the chief tool for extracting this information. However in the BCI field, alternative feature extraction methods have been utilized to meet specific application constraints and objectives.

Band Power

When tracking the amplitude modulations in a specific frequency band, a straight forward solution is to first use a band-pass filter to isolate the frequencies of interest. Then, we rectify the signal to obtain all positive values by squaring the signal or taking it's absolute value. Finally, we smooth signal with a low-pass filter or by integration. One popular way of calculating band power features is to square the band-passed signal and then average all of the squared samples in the sample block:

$$BandPower(BP) = \frac{1}{K}\sum_{i=1}^{K} x(i)^2$$

Where K is the number of samples in the sample block and $x(i)$ is the band-pass filtered signal on a single channel. The band-pass filter most frequently chosen is an IIR filter because it minimizes the time lag between the input and output during asynchronous operation. A squaring filter may then be used, and averaging is most often accomplished with an FIR moving average filter. It should be noted that for band-pass filtered signals which are time-centered (zero-mean), this operation is equivalent to calculating the variance for a set a samples on a given channel. When tracking several frequency bands, it is usually preferable to use the FFT or an AR model to generate the full frequency spectrum rather than using multiple band-pass filters.

Sample Spatial Covariance Matrices

Sample spatial covariance matrices contain within them spatial information that may be useful for discriminating between SMR patterns associated with different tasks. A commonly used scheme is to band-pass filter the signals, segment the data into blocks, apply a CSP spatial filter, and calculate the variance of each spatially filtered channel. Like in PCA, CSP produces output channels that are uncorrelated. If we make the assumption that the output channels are jointly Gaussian, then their uncorrelatedness also makes them statistically independent. Statistical independence implies that for any two channels Z_i and Z_j where $i \neq j$, $COV[Z_i, Z_j] = 0$. However, the idea that these channels may not be truly Gaussian distributed, and thus not independent, and that there could still be useful information for classification in $COV[Z_i, Z_j]$, as well as the discarding of potentially useful information in CSP, is one of the primary motivations for using the sample spatial covariance matrices themselves as features.

Fast Fourier Transform

The fast Fourier transform (FFT) is an efficient computation of the discrete Fourier transform (DFT). The FFT of a digital signal yields the signal's frequency spectrum with a frequency

resolution of sample rate/number of FFT points. The number of FFT points is a user selected parameter that must be greater than or equal to the number of samples and is usually chosen to be a base-2 number for computational efficiency.

The FFT takes N sample points and produces N frequency samples, or bins, uniformly distributed over the interval of $\pm sampling\ rate/2$. However, the spectrum includes only $N/2$ unique frequency bins uniformly distributed from 0 to sampling rate along with their $N/2$ negative mirror images. The FFT produces values with a real part and and an imaginary part which can be easily converted into magnitude and phase.

For finer frequency sampling one can use the technique of zero padding whereby M zeros are appended to the sample block to produce $(M+N)/2$ frequency bins. This does not increase the spectral resolution because no additional information is being supplied, but it allows one to interpolate the spectrum with different bins. The process of segmenting the data may create false discontinuities in the signal when the segment contains a non-integer number of cycles for a frequency component. This may in turn produce ripples around the peaks in the spectrum, an effect known as spectral leakage. This effect can be mitigated by multiplying the sample block by a windowing function to taper the samples at the edges of the block. Although this smooths the spectrum, it also increases the width of the peaks lowering the spectral resolution. [47] The Hanning window is a commonly used windowing function that suffices for 95 percent of cases.

Oftentimes, researchers will talk about the power spectrum of a signal rather than the amplitude spectrum. Because the signal power is proportional to the amplitude squared, a simple estimation of the power spectrum can be obtained by squaring the FFT magnitude. To obtain a non-parametric estimate of the power spectral density, the DFT of the signal's autocorrelation function, of the signal we can use the standard periodogram:

$$S(f) = \frac{\Delta t}{N} |\sum_{n=0}^{N-1} x_n e^{-i2\pi nf}|^2, \quad -\frac{1}{2\Delta t} < f \leq \frac{1}{2\Delta t}$$

Where $S(f)$ is the PSD estimate as function of frequency and Δt is the sampling interval. More robust estimates of the power spectrum can be obtained with variations of the standard periodogram such as Welch's method or Bartlett's method.

Autoregressive Modeling

Autoregressive (AR) modeling, a special case of ARMA modeling, is a parametric method for generating the power spectrum of a signal. It is an alternative to Fourier-based techniques which has the advantages of providing a smoother and more easily interpretable power spectrum than DFT, especially in the case of short time segments, but has the disadvantage of complex model identification. In biomedical applications, AR modeling has been notably used in the analysis of heart rate variability and EEG recordings.

AR modeling makes the assumption that the signal being modeled is stationary and stochastic and can be generated by passing white noise through an IIR filter. The goal of the AR model is to predict the current value of a time series based on past values. The future dependency of the signal on past values is described by the autocorrelation function:

$$r_{xx}[k] = \frac{1}{N}\sum_{n=1}^{N-k} x[n]x[n+k]$$

where $r_{xx}[k]$ is the autocorrelation of x at delay k, N is the number of samples, and $x[n]$ is the value of the signal we wish to model at sample n.

The AR model can be regarded as a set of autocorrelation functions. It is based on the assumption that the signal can be modeled as a weighted sum of M previous samples and an error term ϵ_n:

$$x[n] = \sum_{i=1}^{M} a_i x[n-i] + \epsilon[n] \tag{3.1}$$

where $x[n]$ is the current value of the signal, a_i is the weight of ith delayed sample, M is the model order, and $\epsilon[n]$ is the one-step prediction error, i.e. the difference between the predicted value and the current value of $x[n]$. Expanding equation 3.1:

$$x[n] = a_1 x[n-1] + a_2 x[n-2] + \ldots + a_M x[n-M] + \epsilon[n]$$

or in matrix form:

$$\begin{aligned} x &= \begin{bmatrix} x[M] & x[M-1] & \ldots & x[1] \\ x[M+1] & x[M] & \ldots & x[2] \\ \vdots & \vdots & \ldots & \vdots \\ x[N-1] & x[N-2] & \ldots & x[N-M] \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} + \begin{bmatrix} \epsilon[M+1] \\ \vdots \\ \epsilon[N] \end{bmatrix} \\ &= Xa + \epsilon \end{aligned}$$

We would like to find the optimal a, a_{opt}, such that the prediction error is orthogonal to each column vector in X. Thus,

$$X^T\epsilon = X^T(x - Xa_{opt}) = 0 \Leftrightarrow X^T X a_{opt} = X^T x$$

The least-squares solution is:

$$a_{opt} = (X^T X)^{-1} X^T x \tag{3.2}$$

The direct least-squares solution is known as the covariance method.

Closer inspection of the matrix $X^T X$ and the vector $X^T x$ reveals:

$$\frac{X^T X}{N} \approx R = \begin{bmatrix} r(0) & r(-1) & \ldots & r(1-M) \\ r(1) & r(0) & \ldots & \vdots \\ \vdots & \ldots & \ldots & r(-1) \\ r(M-1) & \ldots & r(1) & r(0) \end{bmatrix}$$

$$\frac{X^T x}{N} \approx r = \begin{bmatrix} r(1) \\ r(2) \\ \vdots \\ r(M) \end{bmatrix}$$

Inserting these approximations into the equation 3.2 yields:

$$a_{opt} = R^{-1} r$$

This is called the Yule-Walker equation, and is known as the autocorrelation method. The recursive method usually used to solve the Yule-Walker equation is called the Levinson-Durbin algorithm.

We can use the covariance method or the autocorrelation method to solve for the model parameters. Alternative methods that are used to estimate the model parameters are derived from these two basic methods. Commonly used in the BCI field, the Burg algorithm is based on minimizing the forward and backward prediction error while satisfying the Levinson-Durbin recursion. The Burg algorithm has the advantage that it is guaranteed to produce a stable IIR model. Because the power spectral density estimate is encoded in the AR parameters, the model coefficients can be used directly as features in a translation algorithm. Some researchers have used adaptive AR parameters derived with a Kalman filtering algorithm as features for BCI systems.

AR spectral estimation is accomplished by considering the AR synthesis filter, equation 3.1. Once the model is fitted to the time series, it is inverted so that the prediction error sequence becomes the input and the measured time series the output to the AR synthesis filter. Properties of the AR synthesis filter are then used to determine the power spectrum of the time series.

The output of any digital filter can be determined with the discrete convolution:

$$y[n] = \sum_{k=-\infty}^{\infty} h[n]x[n-k] = h[n] * x[n]$$

where $y[n]$ is the output sequence, $h[n]$ is the filter's impulse response, $x[n]$ is the input sequence, and $*$ denotes the convolution operator. It is generally mathematically easier to handle convolutions in the frequency domain, so using the DFT:

$$y[n] = h[n] * x[n] \longleftrightarrow H(\omega)X(\omega)$$

To facilitate the handling of sequences in the frequency domain, we can make use of the z-transform:

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n}$$

The z-domain representation of the synthesis filter is:

$$X(z) = \sum_{i=1}^{M} a_i X(z) z^{-i} + W(z)$$

$$= (a_1 z^{-1} + a_2 z^{-2} + \ldots + a_M z^{-M}) X(z) + W(z)$$

$$\rightarrow X(z)(1 - a_1 z^{-1} - a_2 z^{-2} - \ldots - a_M z^{-M}) = W(z)$$

$$\frac{X(z)}{W(z)} = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2} - \ldots - a_M z^{-M}}$$

$$= \frac{1}{1 - \sum_{i=1}^{M} a_i z^{-i}} = H(z)$$

where $H(z)$ is the transfer function of the AR synthesis filter in the z-domain. Thus, the AR filter is an all-pole, IIR filter which attempts to describe the process in the frequency domain with poles only.

We often represent the amplitude and frequency response of a filter by plotting it in the complex plane. In this representation, frequencies are wrapped around the unit circle defined by:

$$z = e^{j\omega T}$$

where ω is the angular frequency and T is sampling period. At 0 rad we have 0 Hz and $\pm\pi$ we have the maximum frequency of 1/(2T) Hz. As we sweep around the unit circle, $H(z)$ peaks when we near a pole. The closer that the pole is to the unit circle, the higher and sharper the peak. In order to have a stable filter we require that all of the poles lie inside of the unit circle, while the zeros can lie anywhere in the complex plane. For real-valued signals, all of the zeros and poles are either real or show up as complex conjugate pairs.

Substituting the definition of z into $H(z)$:

$$H(\omega) = \frac{1}{1 - \sum_{i=1}^{M} a_i e^{-ji\,\omega T}}$$

Using the relation:

$$S_X(f) = |H(f)|^2 S_W(f)$$

We find the power spectral density of our modeled signal as:

$$S_X(f) = \frac{\sigma_W^2}{|1 - \sum_{i=1}^{M} a_i e^{-j2\pi f i}|^2}$$

where σ_W^2 is the prediction error variance. [43]

The main drawback of AR spectral estimation is that it's accuracy is highly dependent on model order. Too low of a model order may blur the spectrum, while too high of a model

order may produce artificial peaks. One approach used in EEG analysis is to assume that the signal is comprised of 3-6 peaks representing some combination of alpha, beta, gamma, and delta waves. Thus, each peak would be represented by a pair of M poles yielding a model order of 6-12. However, this fails to account for adjacent or overlapping bands and wide-band or narrow-band artifacts. Thus, higher model orders are often preferred. The model order should also be increased in proportion to the sampling frequency. For EEG sampled at 160 Hz, a model order of 10-20 is usually selected. [47]

3.4.4 Time-frequency features

Wavelets

Wavelet analysis produces a time-frequency representation of a signal. It aims to overcome one major problem associated with classical spectral-analysis techniques based on the power spectrum: that classical spectral techniques suffer from temporal and spectral resolution that is highly dependent on segment length, model order, and other parameters. This becomes an even more serious problem when the signal contains multiple relevant frequency components each with temporally distinct amplitude modulation characteristics. For example, for a given segment length the amplitude of a component may change significantly over the segment. Thus, the higher frequency components may change greatly from cycle to cycle while the lower frequency components may not change much over the segment because they undergo fewer cycles. Wavelet analysis basically allows one to determine how the amplitude of each frequency component changes over the sample block.

One method of realizing an FFT output is to create a set of parallel band-pass filters called a filter bank where each filter is centered at uniform frequency intervals, and to filter the entire segment through each filter. However in wavelet analysis, we choose to vary the length of a subset of the entire segment that is to be filtered based on the filter's band-pass frequency to create a time-frequency tiling as shown in figure 3.7. Thus, band-pass filters in the filter bank that pass high frequency components will process shorter segments of data, while those that pass low frequency components will process longer segments of data. Each FIR band-pass filter uses an oscillatory template called a mother wavelet, and these mother wavelets will be scaled relative to the band-pass filter. So band-pass filters passing high frequencies will tend to compress the mother wavelet, while those passing low frequencies will stretch the mother wavelet. For subsets of data that are highly correlated with the mother wavelet, there will be a large magnitude response and vice-versa.

There are many different mother wavelets each with unique time-frequency characteristics and mathematical properties. Further, we can derive application specific mother wavelets if we know something about the general pulse characteristics of the signal. Different sets of scaling and time-shifting components can be applied, and the relationships among them can be used to compute a wavelet transform. Analogous to the FFT, the discrete wavelet trans-

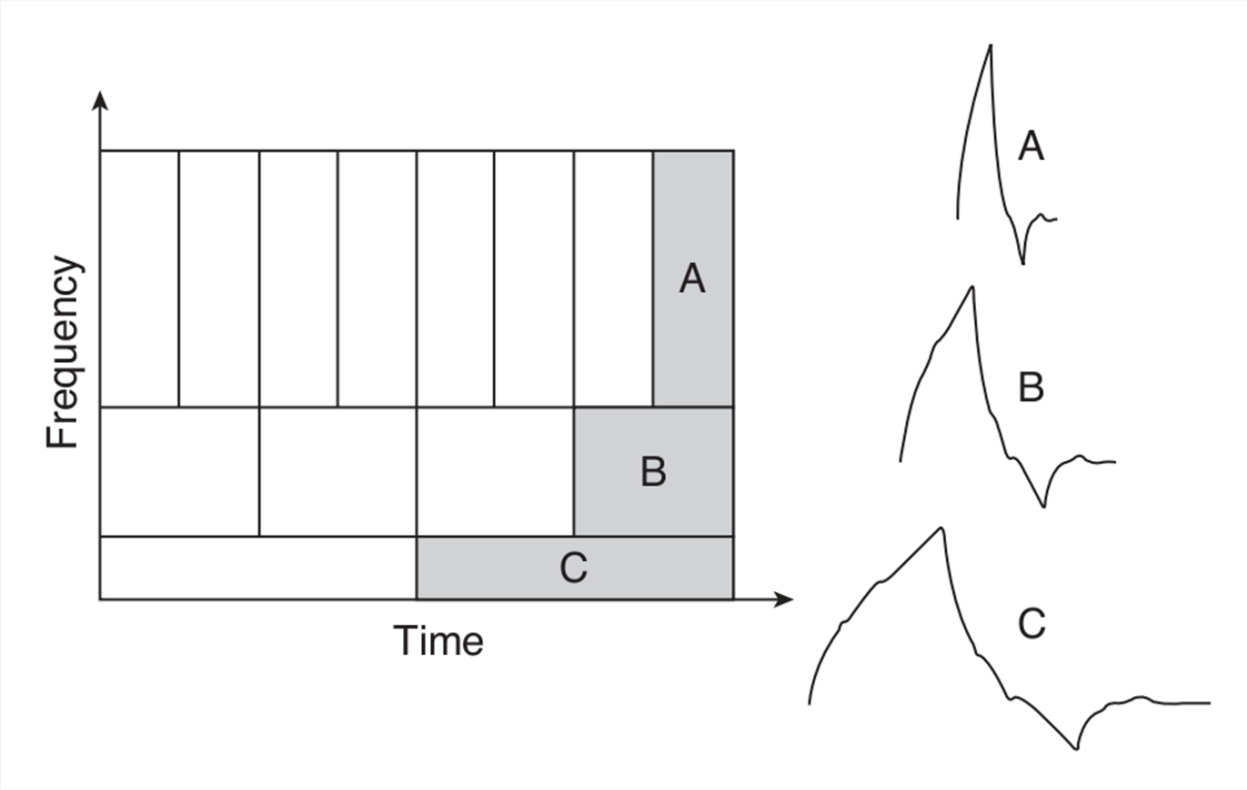

Figure 3.7: Time-frequency tiling along with an example of an associated mother wavelet for each tile. [47] [fair use]

form (DWT) provides an efficient computation of the wavelet transform which minimizes the redundancy in the time-frequency representation. [47]

3.4.5 Similarity features

Phase Locking Value

For some BCI applications, we may interested in knowing the amount of synchrony between two signals occupying the same narrow frequency band. The amount of phase coupling between the two signals can be quantified with the phase locking value (PLV). To calculate the PLV, we first apply a narrow band-pass filter to the signals originating from two different electrodes. Then we calculate the instantaneous phase phase difference using the Hilbert Transform or the phase information from the FFT as:

$$\theta(t) = \theta_{signal1}(t) - \theta_{signal2}(t)$$

where $\theta(t)$ is the instantaneous phase difference at time t, and $\theta_{signal1}(t)$ and $\theta_{signal2}(t)$ are the phases at time t of signal 1 and signal 2, respectively. Because $\theta(t)$ varies with time, we must take the average to determine the consistency of phase coupling between the signals.

However because phase is circular (i.e. $0\ rad = 2\pi\ rad$), we must first convert the phases into vectors having length 1 using the complex exponential. The PLV is then calculated as:

$$PLV = |\frac{1}{N}\sum_{t=1}^{N} e^{j\theta(t)}|$$

When all of the vectors have the same instantaneous phase difference, they will all point in the same direction and the PLV will equal one. The closer that the PLV is to one, the more consistent the phase difference between the signals, i.e. they are phase-locked. When the PLV is near zero, the instantaneous phase difference is random with a normal distribution, indicating that the signals are not phase coupled in the relevant frequency band.

Coherence

Coherence measures the correlation between amplitudes of two signals occupying the same narrow frequency band. Using the FFT or some other spectral estimation technique, we use the power spectrum to select power estimates from the relevant narrow-band. The coherence is then calculated as:

$$\gamma_{xy}^2(\omega) = \frac{|S_{xy}(\omega)|^2}{S_{xx}(\omega)S_{yy}(\omega)}$$

where $S_{xy}(\omega)$ is an average of individual power estimates of the cross spectrum between signals x and y, and $S_{xx}(\omega)$ and $S_{yy}(\omega)$ are the averages of power estimates of signal x and signal y, respectively. $\gamma_{xy}^2(\omega)$ will take on values between zero (no coherence) and one (perfect coherence). It should be noted that a large number of observations is usually needed for an accurate estimate of coherence.

Mahalanobis distance

Features can be defined by measuring the similarity of signal features to the features of a predetermined archetype distribution of these features. One possible similarity feature is the Euclidean, "straight line", distance between two points. However, when the features comprising a feature vector are possibly correlated, then it may be preferable to use an alternative similarity feature, the Mahalanobis distance. The Mahalanobis distance has the advantage that it accounts for the covariance among features and is scale invariant. The Mahalanobis distance is calculated as:

$$D_M(x) = \sqrt{(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

where x is an observation feature vector and Σ and μ are the covariance matrix and mean of the archetypal distribution, respectively.

Figure 3.8 displays a two-dimensional feature space where the two features of the observations are correlated, along with equidistance Euclidean and Mahalanobis contours from the mean of the two-dimensional feature distribution. It can be seen that the equidistance

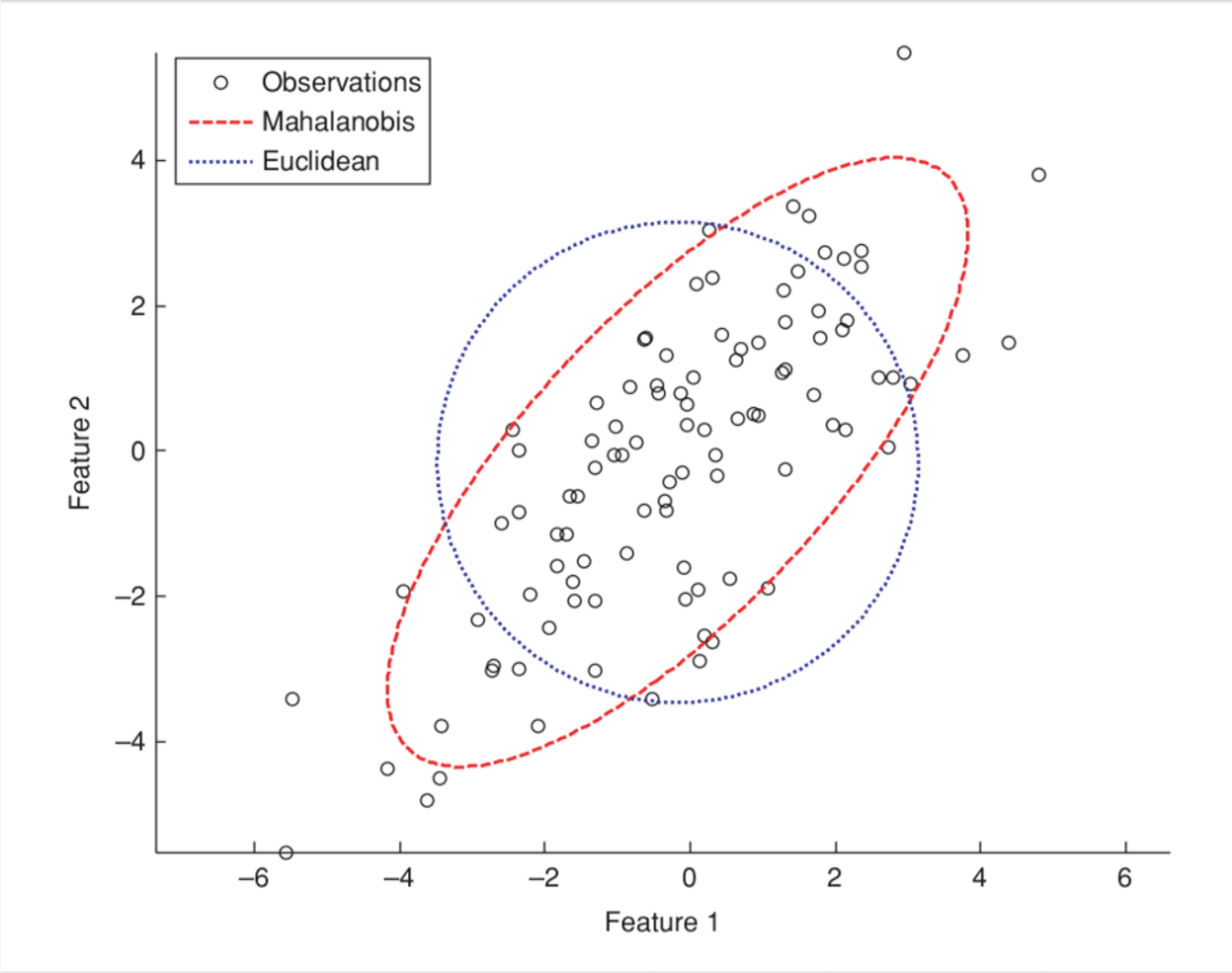

Figure 3.8: Two equidistant contours from the mean of two-dimensional feature distribution where the features are correlated. The Mahalanobis contour (red) captures the covariance (shape of the distribution) of the features while the Euclidean contour does not. [47] [fair use]

Euclidean contour does not have a consistent relationship with the joint distribution of features. However, the equidistance Mahalanobis contour does in fact correlate well with the joint distribution of features. Basically, feature vector observations that exist on equal probability contours of the joint PDF of the archetypal distribution have equal Mahalanobis distance from the mean of that distribution and are thus equally likely to belong to that class. Using training data feature vectors to define the archetypal distribution, we can use the Mahalanobis distance of a testing data feature vector from the mean of the archetypal distribution as a feature to be fed into a translation algorithm. [47]

3.5 Feature Conditioning

3.5.1 Normalization

Much like signal normalization where we aim to make the signal zero-mean and unit-variance, we can also apply these normalization procedures to features. Feature normalization may be beneficial when features differ in mean or dynamic range and those differences are not relevant to a particular BCI application. For example, if one chooses to use a regression translation algorithm for cursor control, the regression weights and thus the magnitude of response will be affected by a feature's mean value. Additionally, if the dynamic ranges of each feature differ greatly, then the translation algorithm may be biased by being more sensitive to one feature than another. However, one should be cautious when applying feature normalization because it may discard information useful for a particular application.

3.5.2 Log-Normal Transforms

Some translation algorithms such as linear discriminant analysis make the assumption that the independent variables, the features, are normally distributed. In general, features extracted by FFT or band power will not be normally distributed because they will range from zero to some upper bound. Moreover, the power in EEG drops off as 1/frequency. Thus, a quick and simple solution is to make the features more Gaussian by applying the log function:

$$y = logx$$

where x is the input feature vector and y is the output feature vector.

3.5.3 PCA and ICA

Analogous to applying PCA or ICA to data blocks in spatial filtering, we can also apply these exact same procedures to sets of feature vectors. This is often done when features are highly correlated or we would like to reduce the dimensionality of the feature vectors. Lower dimensionality feature vectors can be advantageous when the number of available observations used to train the translation algorithm are limited. In the feature domain, the matrix X in the equation $Z = W^T X$ will be comprised of the feature vectors of multiple observations.

3.6 Feature Selection

As discussed in the data-independent spatial filter section, some researchers use r^2 as a metric to determine the channels and frequency bands from which to extract features that are most relevant to the task condition. r^2 in this context is the square of the bi-serial correlation coefficient, because the dependent variable will be a binary value (e.g. representing left or right motor imagery). It is defined as the proportion of the variance of the signal feature that is accounted for by the task condition.

Suppose we have a sequence of one-dimensional observations $(x_1, y_1), \ldots, (x_K, y_K)$ of a single feature x_k (e.g. the amplitude in a given frequency band at a given channel) and a corresponding label $y_k \in [-1, 1]$. We then define X^+ and X^- to be the set of observations x_k associated with $y_k = 1$ and $y_k = -1$, respectively. The bi-serial correlation coefficient is then calculated as:

$$r_X = \frac{\sqrt{N^+N^-}}{N^+ + N^-} \frac{mean(X^-) - mean(X^+)}{std(X^+ \cup X^-)}$$

where N^+ and N^- are the number of observations belonging to class +1 and -1, respectively. One can then select the features having the highest r^2 score. Using the spatially filtered data from each electrode, a feature plot such as in figure 3.9 can be generated.

One may also like to take the model into consideration when selecting the best features to be included. Unfortunately, a simple heuristic that selects those features that are best able to predict the correct output may not yield the optimal subset of features because of the possible correlation between features or the possible suppression of error of another feature. Thus, one may instead choose a stepwise heuristic. In forward stepwise selection, the best individual feature is chosen, and then each of the other features are considered pairwise with this best features. This process is then repeated by considering the addition of another feature to this best pair, and so on. Alternatively, one can use backward stepwise selection where all of the features are initially included in the model, and then features that contribute the least to the correct prediction of the output are removed. It is also possible to combine the forward and backward approaches into forward-backward stepwise selection. The stopping rule is usually based on some criterion variable such as r^2. [47] The calculation of r^2 and its interpretation in this context will be discussed in more detail in the performance evaluation-continuous output section.

3.7 Feature Translation

Once the features have been extracted and selected, we then need to derive translation algorithms that translate these features into device control commands. The commands can be either discrete-valued or continuous-valued. Often we use machine learning approaches to derive a model from features of the training data. We then input features extracted from

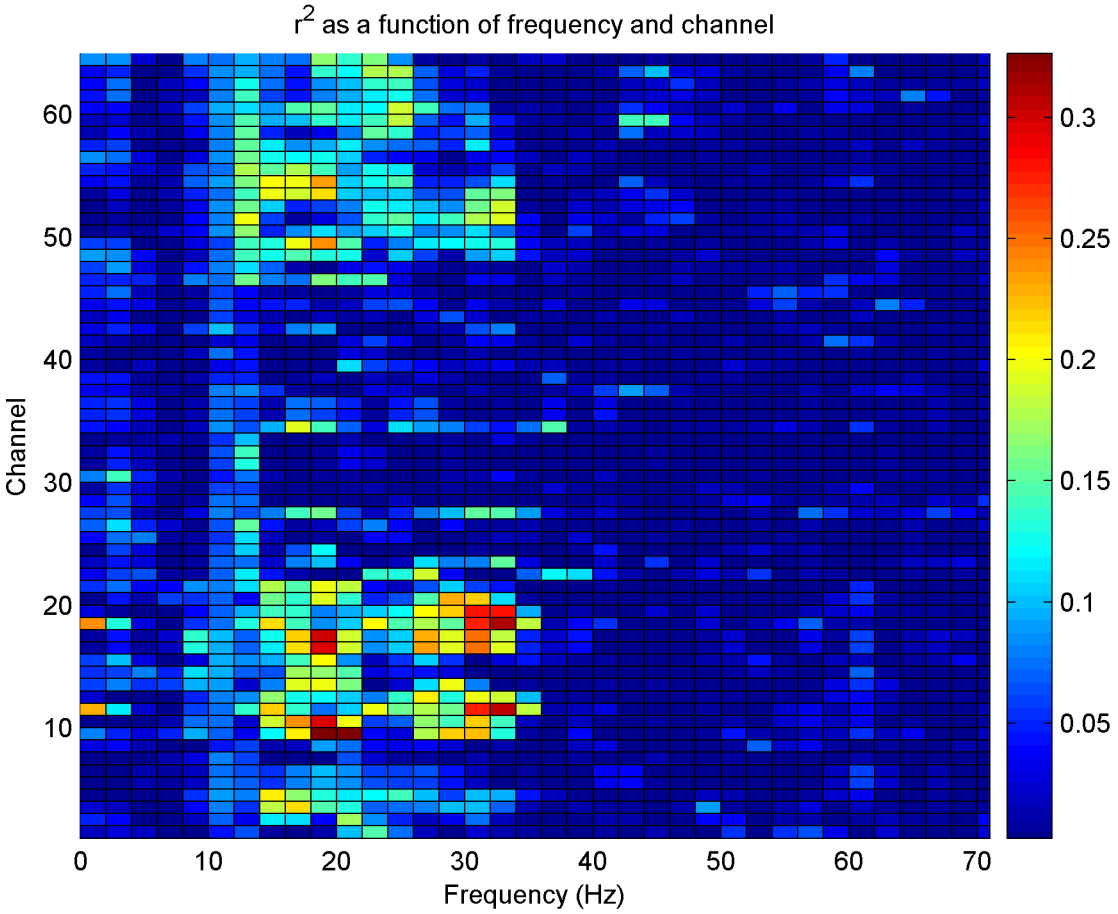

Figure 3.9: Feature plot generated using CAR spatially filtered channels while a user imagined moving his left or right hand. [40] [fair use]

testing data into this model to produce an output. The two major categories of translation algorithms are classification methods, which output discrete commands, and regression methods, which output continuous commands.

The models that we use to describe the data can be either linear or nonlinear. For the present work, only linear models will be described in detail. It should be noted that each of the classification methods described can be formulated for regression and vice versa. Furthermore, by using a sliding window over the testing data and analyzing the output of a classification prediction function at every sample or every few samples, continuous control applications (e.g. cursor control) can be realized as shown in the BCI system in figure 3.10. Gradual outputs from classification functions are also possible by considering the distance of a feature vector from a separating hyperplane as in the case of the Fisher linear discriminant method.

3.7.1 Classification Methods

Classification methods aim to derive a model from the training data with known labels to predict the correct label, or class, of testing data. By the "No Free Lunch" theorem, there is no general superiority of one classifier over all others in pattern classification, and any outperformance of a particular classifier is due to its fit to a particular application. Thus, there are a wide variety of different classifiers commonly utilized in BCI systems, most notably the Fisher linear discriminant (FLD), support vector machine (SVM), minimum distance, naive Bayes, hidden Markov model (HMM), and artificial neural network (ANN). The first two of these approaches will be discussed in the following sections. The minimum distance approach, specifically minimum Riemannian distance to Riemannian mean, will be taken up in detail in the chapter Riemannian Geometry.

Fisher Linear Discriminant

The approach of the Fisher linear discriminant, often also referred to as Linear Discriminant Analysis (LDA), is to project the higher dimensional feature vectors onto a line such that the separation of the feature vectors belonging to the two different classes becomes easier. This operation is shown for the case of two dimensional feature vectors in figure 3.11. For the case of the d-dimensional feature vector $x \in \mathbb{R}^d$, $w \in \mathbb{R}^d$ will define a line in d-dimensional space,

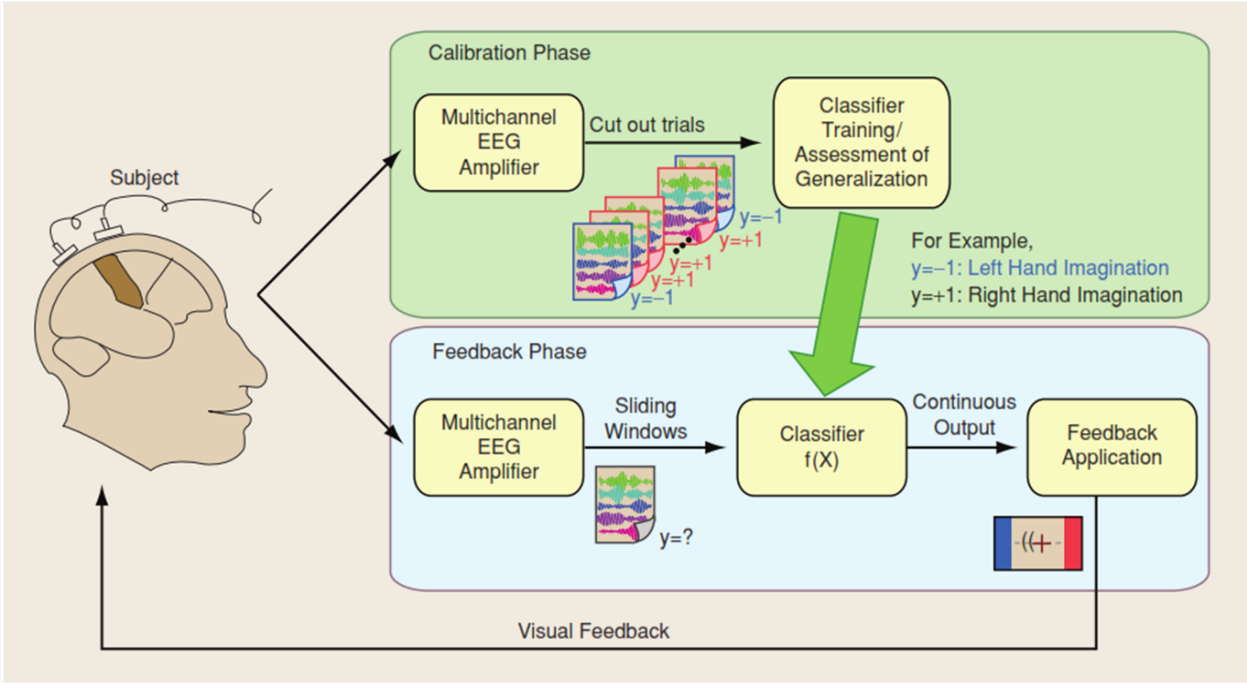

Figure 3.10: BCI system utilizing a classification prediction function along with a sliding window to realize continuous cursor control. [6] [fair use]

and we use the dot product between w and x to project the feature vector onto this line. Perpendicular to this line and located somewhere along it is what is known as the separating hyperplane. Basically, the Fisher linear discriminant aims to paramertize the hyperplane that best separates the two classes of feature vectors.

The separability of the two classes is measured by the distance between the projected class means and the amount of variance of the features in this direction. Thus we attempt to maximize the ratio of between-class scatter to within-class scatter:

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

where w is the weight vector in direction of the projected space, S_B is the between-class scatter, and S_W is the within-class scatter.

The between-class scatter is calculated as:

$$S_B = (m_1 - m_2)(m_1 - m_2)^T$$

where m_1 and m_2 are the sample mean vectors associated with class 1 and class 2 and calculated as:

$$m_\omega = \frac{1}{n_\omega} \sum_{j \in I_\omega} x_j$$

where $\omega \in [1, 2]$ represents the two classes, n_ω and I_ω are the number of samples and the set of indices, respectively, and x_j is the jth feature vector.

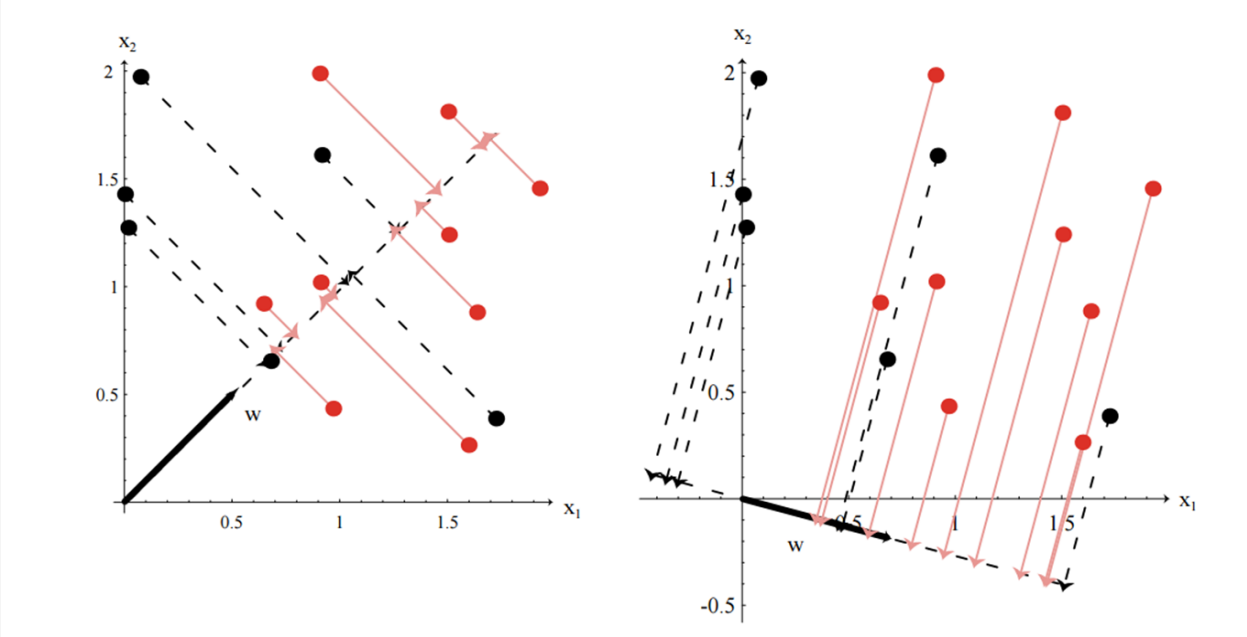

Figure 3.11: The projection of feature vectors onto to lines. It can be seen that projection onto the line on the right would allow for better separation of the red and black points. [12] [fair use]

The within-class scatter is calculated as:

$$S_W = S_1 + S_2$$

where S_1 and S_2 are the scatter matrices associated with class 1 and class 2, respectively, and calculated as:

$$S_\omega = \sum_{j \in I_\omega} (x_j - m_\omega)(x_j - m_\omega)^T$$

We can compute the optimal weight vector, w, which is one of the generalized eigenvectors that jointly diagonalizes S_B and S_W, as:

$$w = S_W^{-1}(m_1 - m_2)$$

This weight vector lies in the projected space, and the best separating hyperplane is orthogonal to it. We must also compute the bias, b, which is the distance in the direction of w from the origin of the projected space to the separating hyperplane as:

$$b = \frac{1}{2} w^T (m_1 + m_2)$$

Once we have parameterized the hyperplane, we next need to come up with a prediction function so that we can predict which class a testing feature vector belongs to. The prediction function becomes:

$$y = sign(w^T x + b)$$

if $w^T x \geq b$ then $y = 1$, and if $w^T x < b$ then $y = -1$. These can be easily converted to the predicted label, 1 or 2, as: $y_{pred} = \frac{y+3}{2}$.

Support Vector Machine

Much like the Fisher linear discriminant, the support vector machine (SVM) is a linear discriminant technique that seeks a hyperplane that best separates feature vectors belonging to different classes. However, while Fisher linear discriminant attempts to maximize the margin between the class means, SVM tries to maximize the margin of separation between the two classes. Conceptually, one can think about having a set of two-dimensional features belonging two different classes. The SVM aims to find the thickest possible line that can separate the two classes without misclassification as shown in figure 3.12. The thickness of this line is termed the margin.

Assume that we have a set of data points and labels:

$$D = \{(x_1, y_1), \ldots (x_n, y_n)\}$$

where $x_i \in \mathbb{R}^d$ are our d-dimensional feature vectors and $y_i \in \{-1, 1\}$ are our labels. In SVM, for correct classification we assume that:

$$y_i(w^T x_i + b) \geq 1 \quad \forall i \in \{1, \ldots, n\}$$

where w is the weight vector perpendicular to the separating hyperplane, b is the bias, and we have chosen greater than equal to one instead of zero so that there will be some margin between the points.

If we imagine two points, x_+ on one side of the margin and x_- on the other side of the margin, both lying on the edge of the margin directly across from one another, then we can calculate the distance between these two points as:

$$margin = \sqrt{(x_+ - x_-)^T(x_+ - x_-)}$$

Because we have chosen the two points to lie right on the edge of the margin, we can write

$$w^T x_- + b = -1$$

$$w^T x_+ + b = 1$$

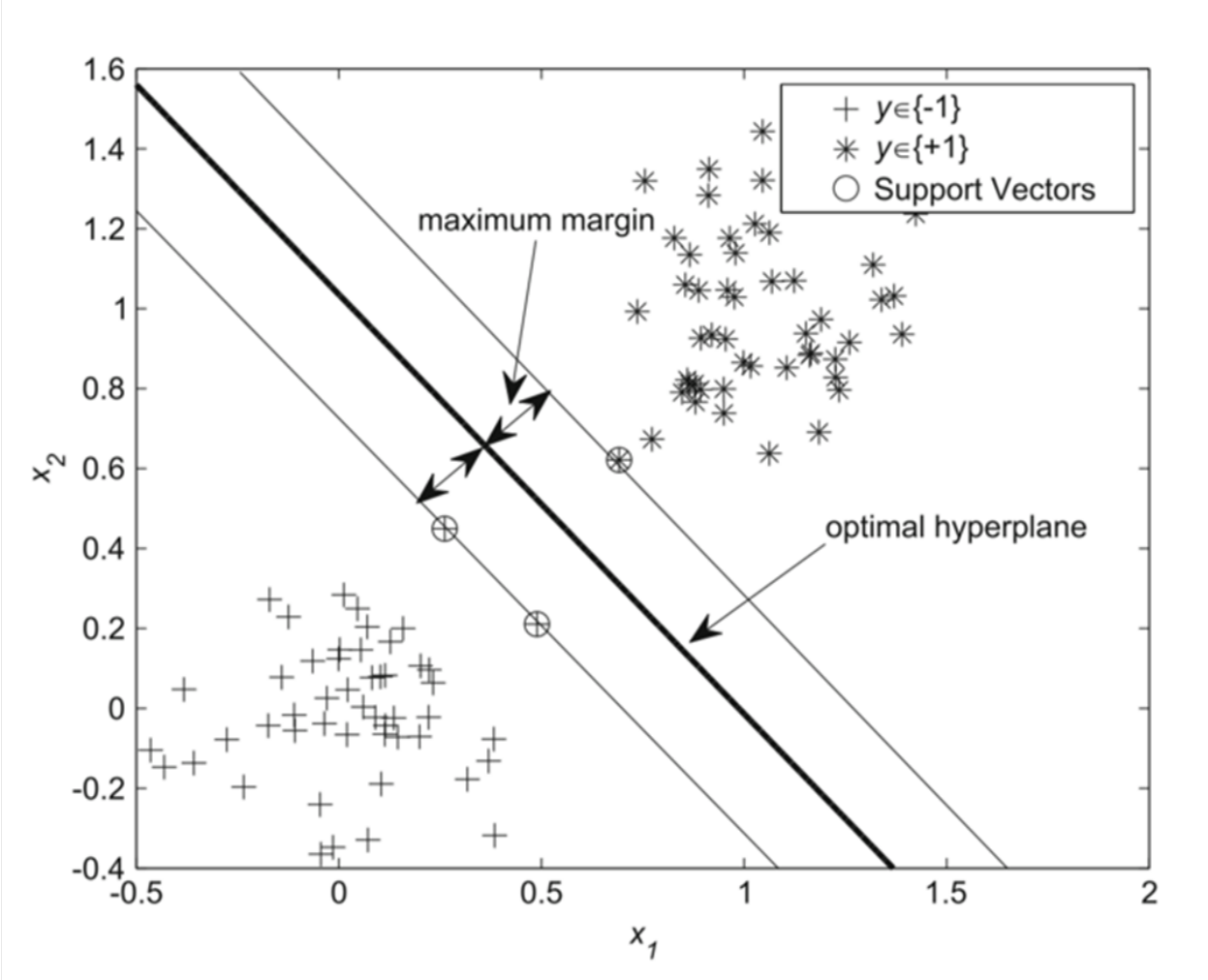

Figure 3.12: Support vector machine for two-dimensional feature vectors. [25] [fair use]

Because require that x_- and x_+ lie directly across from another, we can write:

$$x_+ - x_- = \gamma w$$

where γ is some scaling factor of w. After some simple algebraic manipulations, we find that:

$$margin = \frac{2}{\sqrt{w^T w}}$$

Thus our optimization problem can be written as:

$$\max_{w \in \mathbb{R}^d} \frac{2}{\sqrt{w^T w}}$$

$$s.t. \quad y_i(w^T x_i + b) \geq 1 \quad \forall i \in \{1, \ldots, n\}$$

This is exactly equivalent to the optimization problem:

$$\min_{w \in \mathbb{R}^d} \frac{1}{2} w^T w$$

$$s.t. \quad y_i(w^T x_i + b) \geq 1 \quad \forall i \in \{1, \ldots, n\}$$

which is the standard form for the separable SVM. This form of the optimization problem now fits into a class of problems called quadratic programs which have been well-studied and have optimized optimization algorithms for their solutions.

For the case that there are some features that cannot be correctly classified for any choice of w, we can use the technique of slackening. By slackening, we add a slack variable, $\xi_i \geq 0$, to the constraint equation as well as a slack penalty term to the function we are trying to minimize. Thus, the soft-margin optimization problem becomes:

$$\min_{\substack{w \in \mathbb{R}^d \\ \xi_i \geq 0}} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i$$

$$s.t. \quad y_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i \in \{1, \ldots, n\}$$

where C is a parameter that determines how much the optimization will have to pay for increasing the slack. As in Fisher linear discriminant, the prediction function for SVM is:

$$y = sign(w^T x + b)$$

3.7.2 Regression Method

Linear Least Squares Regression

The linear regression method aims to fit the training data with a linear model of the form:

$$\hat{y} = w^T x + b$$

where $\hat{y}$ is the continuous predicted output for some input feature vector x. The least squares method is the most common method of parameterizing this model by finding the regression weights w and bias b. We seek a function, $f(x)$, that predicts the proper output given some input x times some coefficients:

$$f(x) = w^T x$$

where $w \in \mathbb{R}^d$ is the augmented weight vector, i.e. includes the bias term, and $x \in \mathbb{R}^d$ is an augmented feature vector, i.e. has an extra 1 added as the last entry. Note that $f(x) \in \mathbb{R}$ is a predicted scalar output that we are trying to match to the actual output $y \in \mathbb{R}$.

The training objective is as follows:

$$\min_w \frac{1}{2} \sum_{i=1}^{n} (f(x_i) - y_i)^2$$

Thus, we are trying to minimize the squared error between the predicted output and the actual output for training data. In matrix form, we can write the function $f(x)$ as:

$$f(X) = X^T w$$

where $X \in \mathbb{R}^{d \times N}$ is a matrix of $d-1$ features augmented with a row of all ones of N observations. Thus, we can substitute $f(X)$ into the objective function to get:

$$J(w) = \frac{1}{2}(X^T w - y)^T (X^T w - y)$$

$$= \frac{1}{2} w^T X X^T w - y^T X^T w + \frac{1}{2} y^T y$$

To find the minimum, we take the gradient of this function with respect to w and set it equal to zero:

$$\nabla_w J = X X^T w - X y = 0$$

Thus we find that the w that minimizes the squared error can be calculated directly as:

$$w = (X X^T)^{-1} X y$$

solving this equation will yield the regression weights as well as the bias term.

Some researchers (e.g. The Wadsworth Group) focusing on cursor control applications derive regression weights using a non-augmented feature matrix X. However, it should be noted that the features in these experiments are normalized in real-time to have zero-mean (by subtracting the mean of each row in X from each entry in that row) and unit-variance (by dividing each entry in a row of X by the standard deviation of that row). Thus, the regression weights will be the same as the case that the normalized matrix X is augmented with ones and an additional intercept term is calculated.

It should also be noted that this procedure can be easily applied to non-separable classification problems in finding the minimum squared error (MSE) solution to the linear discriminant function. This solution is directly related to the Fisher linear discriminant. In fact, these two solutions are equivalent when y is a binary -1 or +1 scaled by n_i/n, where n_i is the number of samples belonging to class i and n is the total number of samples.

3.7.3 Additional Considerations

Because BCI systems must work in real-time, the key to any model used for a translation algorithm is that it must generalize well to unseen testing samples. Biological data, and EEG especially, usually displays considerable chance variation over time. This problem can often be mitigated by using large sets of training data. That is, the generalizability of a model tends to increase with increasing number of training observations. It is also important to avoid overfitting a model whereby the model is tuned so precisely to the training data that it may not generalize well to the testing data. Thus, while more complex models with more parameters tend to fit the training data better, the generalizability tends to decrease with increasing number of parameters. The process of regularization whereby additional constraints are placed on the parameters can effectively eliminate parameters from the model reducing its complexity and improving its generalizability.

Another issue with BCI feature translation is the potential nonstationarity of the data. Nonstationary data refers to data whose statistics change over time. These statistical metrics can include the mean of each feature, each feature's variance, or the relationship among features such as the linear relationship or covariance. These changes can occur with user fatigue, the user changing his or her control strategy, as a disease such as ALS progresses, etc. Nonstationarity can lead to poor generalizability because a static translation algorithm would not be able to accurately capture the changing statistics in the data. Thus, a solution to the nonstationarity problem that is frequently employed by researchers is to use adaptive feature translation algorithms.

When parameterizing a model, an important consideration is the time-frame of the data to be used, i.e. the data window. Long data windows tend to provide more accurate parameter estimates, while short data windows tend to be better able to handle nonstationarity and track the changes in data statistics. Data windows can be static, whereby the parameters are estimated from some fixed block of data and never changed, or sliding, whereby the param-

eters are estimated from some block of data and then re-estimated from a subsequent block of data. Static windows are typically used for stationary data or where the changing data statistics is not of immediate importance to the particular application, while sliding windows are typically used for nonstationary data and applications that require the translation algorithm to capture the changing data statistics, e.g. cursor control.

One commonly used sliding window method is to give more recent sample more weight in determining the model's parameters. The simplest adaptive algorithm utilizing this method to determine model parameters is the least-mean squares (LMS) algorithm. The LMS algorithm first calculates the error between the actual current output and the predicted current output:

$$e(t) = Y(t) - Y'(t)$$

where $e(t)$ is the error, $Y(t)$ is the actual output, and $Y'(t)$ is the predicted output, all at time t. The LMS algorithm then predicts the future parameters using this error and the current parameters:

$$w(t+1) = w(t) + le(t)X(t)$$

where $w(t+1)$ and $w(t)$ are the parameter vectors at time $t+1$ and t, respectively, l is the learning rate that controls the rate of adaptation, and $X(t)$ is the current feature vector.

The LMS algorithm works by adjusting the parameters a small amount in the direction that reduces the error of the current trial. The LMS algorithm will track the the least-squares solution with a properly chosen learning rate. More complex adaptive algorithms such as recursive least squares (RLS) or Kalman filter have also been used in the BCI field to parameterize models. Although the LMS or Kalman filter algorithms work well with relatively simple error surfaces, sometimes researchers will employ alternative optimization algorithms (such as genetic algorithms or particle swarm optimization) in order to avoid unintentionally getting caught in local error minima. [47]

3.8 Performance Evaluation

3.8.1 Cross-Validation

In setting parameters for use in a BCI system (e.g. frequency band for most relevant features, length of time segment to be extracted, or parameters for spatial filters or translation algorithms), we would often like to know how well a certain configuration performs so that the optimal parameters can be selected. To determine the optimal parameters, the obvious solution is collect a lot of training data. However, because this can be a demanding and tedious process for the user, there is an alternative approach that is often preferred. Using the technique of cross-validation, we can divide our training data up so that a subset becomes the new "training" data and the remainder becomes the "testing" data. By choosing differ-

ent subsets of data to be training and testing, we can better evaluate a given configuration and compare it to an alternative configuration.

In K-folds cross-validation we begin by dividing the training set D into K equal parts denoted D_j where $j = 1, \ldots, K$. For the jth fold, D_j serves as a test set, while the remaining $K - 1$ parts serve as a training set. The prediction accuracy rate r_j is then computed using the predicted labels and the true labels. We then average the r_j's for each j to obtain the average prediction accuracy rate r. We then select the set of parameters that maximize r. [25]

3.8.2 Accuracy, Sensitivity, Specificity, and Selectivity

If we imagine a binary classifier that predicts class p or class n, we can derive some definitions based on whether or not the classification is correct. If the classifier predicts p, and the true class is p, this is referred to as a true positive output (TP). However, if the classifier predicts p, but the true class is n, we refer to this as a false positive output (FP). Similarly, when the classifier predicts n, and the true class is n, we call this a true negative output (TN). Finally, if the classifier predicts n while the true class is p, then this is referred to as a false negative output (FN). If n is some state of desired inactivity while p is a desired "move-forward" command for a wheelchair, then a false positive output would be the most dangerous error for a user and what we as designers would try our best to avoid.

We can use these different types of outputs to define some important performance metrics:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Sensitivity\,(TP\,rate) = \frac{TP}{TP + FN}$$

$$Specificity\,(negative\,predictive\,value) = \frac{TN}{TN + FN}$$

$$Selectivity\,(positive\,predictive\,value) = \frac{TP}{TP + FP}$$

Accuracy is the most commonly used metric for performance evaluation of BCI systems producing discrete outputs. However, for the wheelchair situation mentioned earlier, we may want the selectivity to be as close to 1 as possible. Furthermore, accuracy alone does not account for the time it takes a classifier to make a decision or the consistency of performance as accuracy could vary greatly over the course of a day. [47]

In asynchronous BCI systems where the system must be able to classify between a state of intentional activity and a state of intentional inactivity, researchers will sometimes use ROC curves to determine the threshold that a feature must go beyond to output an "activity" class. ROC curves are plots of sensitivity vs. 1-specificity (TP rate vs. FP rate) as the threshold is varied. A lower threshold is associated with a higher sensitivity but a lower specificity.

Essentially, ROC curves attempt to describe the inherent trade-off between sensitivity and specificity. [25]

3.8.3 Confusion Matrix and Kappa Coefficient

A more sophisticated analysis than simple classification accuracy can be represented in a confusion matrix. For a 2-class classification problem, the confusion matrix shows the number of TPs, FNs, FPs, and TNs as shown in figure 3.13. Alternatively, one can display

Class	Y	N
Y	Hits (TP)	Misses (FN)
N	FA (FP)	CR (TN)

Figure 3.13: 2-class confusion matrix with FA indicating false activations and CR indicating correct rejections. [42] [fair use]

the percentages of these occurrences. Confusion matrices can also be extended to M-class classification problems. The confusion matrix shown in figure 3.14 was used to describe the

Class	1	2	3	4	Total
1	73	17	7	8	105
2	10	87	3	5	105
3	6	13	74	12	105
4	2	4	7	92	105
Total	91	121	91	117	420

Figure 3.14: Example of a 4-class confusion matrix for BCI competition III dataset IIIa indicating classifications for a given time point in 420 trials. [42] [fair use]

outputs from a 4-class, continuous classification problem posed in BCI Competition III for dataset IIIa. The competition required algorithms to be causal so that for each time point in a given trial, classification could be done using only that time point and a window of previous time points. Thus, a confusion matrix was built for each time point across every artifact-free trial.

Confusion matrices can be used to find Cohen's Kappa Coefficient. Overall accuracy p_0 of continuous classification can be found as:

$$p_0 = \frac{\sum_{i=1}^{M} n_{ii}}{N}$$

where n_{ii} is the number of samples that have been correctly classified (i.e., the diagonal elements of the confusion matrix), M is the number of classes, and $N = \sum_{i=1}^{M} \sum_{j=1}^{M} n_{ij}$ is the total number of samples. Chance agreement p_e can be found as:

$$p_e = \frac{\sum_{i=1}^{M} n_{:i} n_{i:}}{N^2}$$

where $n_{:i}$ and $n_{i:}$ are the sum of the ith column and the ith row, respectively. From these two quantities, the Kappa coefficient κ can be calculated as:

$$\kappa = \frac{p_0 - p_e}{1 - p_e}$$

The standard error $\sigma_e(\kappa)$ can be found as:

$$\sigma_e(\kappa) = \frac{\sqrt{(p_0 + p_e^2 - \sum_{i=1}^{M}[n_{:i}n_{i:}(n_{:i} + n_{i:})]/N^3)}}{(1 - p_e)\sqrt{N}}$$

[42]

3.8.4 Continuous Output Evaluation

Continuous output applications (e.g. cursor control) can be evaluated with a continuous metric. At each time a feature vector is used to produce an output, we can calculate the squared prediction error, i.e. the squared difference between the actual output and the correct output. Summing the squared prediction errors produces the χ^2 statistic which represents the variance of the output due to BCI error. Dividing by the total variance of the output yields the proportion of total variance due to error, $1 - r^2$. r^2 is then easily calculated and is the proportion of the total variance accounted for by the model. Alternatively, one may choose to evaluate the system based on the root-mean-squared (RMS) error, which is the square root of the average squared error. [47]

3.8.5 Information Transfer Rate

The information transfer rate (ITR) as defined by Shannon is the amount of information communicated per unit time. The ITR incorporates accuracy, speed, and the number of classes to give a more objective performance metric for the comparison of different BCI systems and for measuring system improvement. In bits per trial, the ITR is defined as:

$$B = \log_2 N + P \log_2 P + (1 - P) \log_2 \frac{1 - P}{N - 1}$$

where N is the number of classes and P is the classification accuracy. If the trial lasts t seconds, B/t yields the ITR in the form of the bit rate. [25]

Chapter 4

Riemannian Geometry

4.1 Introduction

Researchers in the BCI field have found that using Riemannian geometry can improve motor imagery BCI performance in both binary [1] and multi-class [2] classification problems. Using the structure of the sample spatial covariance matrix (SCM), the process of spatial filtering and classification can be combined into one unique step. The key idea is that the space of sample SCMs is a differentiable Riemannian manifold $\mathcal{M}$ in which the tangent space at each point is a finite-dimensional Euclidean space. Classical treatment of SCMs, as in the derivation of CSP spatial filters, treats them as if they naturally lie in Euclidean space. However, using Riemannian geometry provides a rich framework with which to manipulate SCMs in their natural space.

4.2 Preliminaries

We denote the space of $n \times n$ symmetric matrices in the space of square real matrices $M(n)$ as $S(n) = \{S \in M(n), S^T = S\}$. We then state that sample SCMs belong to the set of $n \times n$ symmetric positive-definite (SPD) matrices such that $P(n) = \{P \in S(n), u^T P u > 0, \forall u \in \mathbb{R}^n\}$. SPD matrices have the property that they are diagnolizable with strictly real positive eigenvalues. Finally, we denote $Gl(n)$ to be the set of $n \times n$ invertible matrices in $M(n)$.

The exponential matrix and logarithmic matrix of P is obtained by first using the eigende-composition of P:

$$P = U diag(\sigma_1, \ldots, \sigma_n) U^T$$

where U is the matrix of eigenvectors of P and $\sigma_1, \ldots, \sigma_n$ are the positive, real eigenvalues

of P. The exponential matrix can then be calculated as:

$$exp(P) = Udiag(exp(\sigma_1), \ldots, exp(\sigma_n))U^T \tag{4.1}$$

and the logarithmic matrix, the inverse operation, can be calculated as:

$$log(P) = Udiag(log(\sigma_1), \ldots, log(\sigma_n))U^T \tag{4.2}$$

. The following properties hold:

- $\forall P \in P(n), det(P) > 0$
- $\forall P \in P(n), P^{-1} \in P(n)$
- $\forall (P_1, P_2) \in P(n)^2, P_1 P_2 \in P(n)$
- $\forall P \in P(n), log(P) \in S(n)$
- $\forall S \in S(n), exp(S) \in P(n)$

The space of SPD matrices $P(n)$ is a differentiable Riemannian manifold $\mathcal{M}$ of dimension $m = n(n+1)/2$. The derivatives at a matrix P on the manifold lies in a vector space called the tangent space T_P at that point. The tangent space is also of dimension m and lies in the space $S(n)$.

Riemannian manifolds have an associated Riemannian metric, which is an inner product $<,>_P$ on each tangent space that varies smoothly from point to point on the manifold. The natural metric on the manifold of SPD matrices is the local inner product:

$$< S_1, S_2 >_P = Tr(S_1 P^{-1} S_2 P^{-1})$$

The inner product induces a norm for the tangent vectors such that $< S, S >_P = ||S||_P^2 = Tr(SP^{-1}SP^{-1})$. At the identity matrix, the norm simplifies to the Frobenius norm $< S, S >_I = ||S||_F^2$. [2]

4.3 Riemannian Distance

Let $\Gamma(t) : [0, 1] \to P(n)$ be any differentiable curve lying in the manifold connecting any two points (SPD matrices) on the manifold. The length of such a curve can be computed as:

$$L(\Gamma(t)) = \int_0^1 ||\dot{\Gamma}(t)||_{\Gamma(t)} dt$$

The minimum length curve lying in the manifold connecting two SPD matrices P_1 and P_2 in $P(n)$ is called a geodesic. The length of the geodesic is referred to as the Riemannian distance and is calculated as:

$$\delta_R(P_1, P_2) = ||log(P_1^{-1}P_2)||_F = [\sum_{i=1}^{n} log^2\lambda_i]^{1/2} \tag{4.3}$$

where $||\cdot||_F$ is the Frobenius norm of a matrix and the λ_i's are the real strictly positive eigenvalues of $P_1^{-1}P_2$. Important properties of the Riemannian distance include:

- $\delta_R(P_1, P_2) = \delta_R(P_2, P_1)$
- $\delta_R(P_1, P_2) = \delta_R(P_1^{-1}, P_2^{-1})$
- $\delta_R(P_1, P_2) = \delta_R(W^T P_1 W, W^T P_2 T) \forall W \in Gl(n)$

Inspection of the third property reveals that the space of SPD matrices is invariant to projection allowing us to manipulate the space with tools such as PCA without affecting the distance. [2]

4.4 Exponential and Logarithmic Maps

For each point in the manifold P there exists a tangent space composed of a set tangent vectors at P. Each tangent vector $S_i \in S(n)$ can be thought of as the derivative at $t = 0$ of the geodesic $\Gamma_i(t)$ connecting P and another point in the manifold P_i. Using what is known as an exponential mapping, we can project this vector S_i back onto the manifold to obtain P_i as:

$$Exp_P(S_i) = P_i = P^{1/2}exp(P^{-1/2}S_iP^{-1/2})P^{1/2}$$

Because $P^{-1/2}S_iP^{-1/2}$ is an element of $S(n)$ and not $P(n)$, we can not simply use equation 4.1 to calculate the matrix exponential. Instead, we use this quantity as a normalized tangent vector S_i and calculate the exponential map using it and P in an efficient algorithm that can be found in [16].

The inverse operation is what is known as a logarithmic map. Logarithmic mapping allows us to project any SPD matrix P_i in the manifold onto the tangent space at any SPD matrix $P \in P(n)$ as:

$$Log_P(P_i) = S_i = P^{1/2}log(P^{-1/2}P_iP^{-1/2})P^{1/2} \tag{4.4}$$

These two operations can be seen geometrically in figure 4.1. As a word of caution to the reader, S_i is a tangent vector in matrix form and a member of $S(n)$. We will later vectorize S_i to form s_i which is an $m = n(n+1)/2$ dimensional vector in the traditional sense. This is possible because S_i is symmetric and has only m unique entries that fully characterize it.

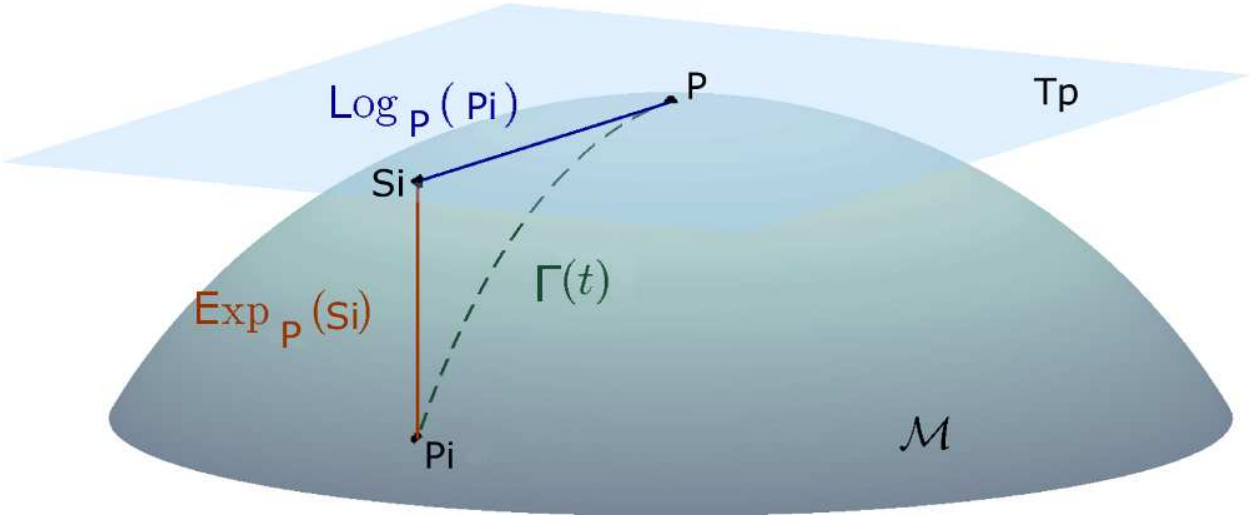

Figure 4.1: Manifold $\mathcal{M}$ and tangent space at point P T_P. The derivative of the curve $\Gamma(t)$ connecting P and P_i at $t = 0$ is a vector S_i in T_P. The exponential map maps P_i to a point in T_P, and the logarithmic map maps S_i back to $\mathcal{M}$. [2] [fair use]

Using the concept of the logarithmic map, we can derive an alternative definition for the Riemmanian distance:

$$\begin{aligned} \delta_R(P, P_i) &= ||Log_P(P_i)||_P = ||S_i||_P \\ &= ||upper(P^{-1/2} Log_P(P_i) P^{-1/2})||_2 = ||s_i||_2 \end{aligned} \tag{4.5}$$

where the $||\cdot||_P$ operator denotes the norm of a tangent vector $S_i \in S(n)$ calculated as $||S_i||_P = \sqrt{Tr(S_i P^{-1} S_2 P^{-1})}$ and $||\cdot||_2$ denotes the standard L_2 norm of a vector. The *upper* operator vectorizes the symmetric matrix by retaining the upper triangular part, applying unity weight to the diagonal elements, and applying $\sqrt{2}$ weight to the off-diagonal elements. s_i is an m dimensional vector of the normalized the tangent space.

We can derive some definitions for s_i with inspection of equation 4.5:

$$s_i = upper(P^{-1/2} Log_P(P_i) P^{-1/2}) \tag{4.6}$$

Substituting equation 4.4 into equation 4.6 yields:

$$s_i = upper(P^{-1/2} S_i P^{-1/2}) = upper(P^{-1/2} P^{1/2} log(P^{-1/2} P_i P^{-1/2}) P^{1/2} P^{-1/2})$$

using the fact that $P^{-1/2} P^{1/2} = P^{1/2} P^{-1/2} = I(n)$, where $I(n)$ is the $n \times n$ identity matrix, we can write a final expression for s_i as:

$$s_i = upper(log(P^{-1/2} P_i P^{-1/2})) \tag{4.7}$$

[2].

4.5 Riemannian Mean

Techniques such as CSP require averaging SPD matrices in order to derive spatial filters to yield maximally discriminative output channels. The average traditionally used is called the arithmetic mean or the Euclidean mean. Using the definition of Euclidean distance on $\mathcal{M}(n)$, $\delta_E(P_1, P_2) = ||P_1 - P_2||_F$, we can compute the Euclidean mean of $I \geq 1$ SPD matrices as:

$$\mathcal{A}(P_1, \ldots, P_I) = \underset{P \in P(n)}{\arg\min} \sum_{i=1}^{I} \delta_E^2(P, P_i) = \frac{1}{I} \sum_{i=1}^{I} P_i$$

Because $P(n)$ is convex, $\mathcal{A}(P_1, \ldots, P_I)$ does lie within $P(n)$. However, the arithmetic mean does not interpolate natural properties. For instance, the linear average of matrices with the same determinant can result in a matrix with a larger determinant. Furthermore in linear PCA, the straight lines defined by the modes of variation do not stay within $P(n)$ and thus do not preserve positive definiteness. This motivates the utilization of an alternative mean metric, the Riemannian mean.

The Riemannian mean is the geometric mean of a set of SPD matrices in $\mathcal{M}(n)$. We can derive an expression for the Riemannian mean of $I \geq 1$ SPD matrices using the Riemannian distance:

$$\mathcal{G}(P_1, \ldots, P_I) = \underset{P \in P(n)}{\arg\min} \sum_{i=1}^{I} \delta_R^2(P, P_i)$$

Such a local minimum is guaranteed to exist because $P(n)$ is a manifold of non-positive sectional curvature. However, no closed-form expression exists to compute this mean, so we must instead rely on iterative optimization algorithms to compute it. [2]

4.6 Classification Using Riemannian Geometry

4.6.1 Minimum Distance to Riemannian Mean

There have been several methods used for classification using Riemannian geometry. The simplest approach is called Minimum Distance to Riemannian Mean (MDRM). This is a supervised technique in which we assess the Riemannian distance of a testing sample SCM to the intra-class Riemannian mean $P_{\mathcal{G}}^{(k)}$ of the training sample SCMs belonging to class k. The unknown sample SCM is then assigned to the class k having the minimum Riemannian distance.

4.6.2 Tangent Space LDA

Although MDM classification using Riemannian distance and mean have shown promising results in the literature [2], researchers have sought to use Riemannian geometry concepts in more sophisticated classification algorithms to improve performance. Because many popular and efficient algorithms (e.g. LDA or SVM) rely on projections onto hyperplanes, they cannot be directly implemented using sample SCM features which exist on a Riemannian manifold. Using the concept of the tangent space, researchers have devised a method called tangent space LDA (TSLDA) that makes LDA amenable to these features.

The basic idea of TSLDA is to first project sample SCM's onto the tangent space at the Riemannian mean of the entire training set. Because the tangent space is a Euclidean space, the LDA approach can be implemented. As discussed in the Fisher Linear Discriminant theory section, we may then parameterize the LDA prediction function using the training tangent feature vectors. For classification, we project a test sample SPD matrix with unknown label onto the tangent space at the Riemannian mean of the training set, vectorize the resulting tangent vector matrix, and assign it the label according to:

$$y_i = sign(w^T s_i + b)$$

Overfitting

A peculiarity arises due to the fact that the tangent vectors are $m = n(n+1)/2$ dimensional. So, for example, 22×22 SPD matrices would have 253-dimensional tangent vectors. Because the dimensionality of the tangent vectors increases very rapidly with larger covariance matrices, the TSLDA approach can be problematic because the LDA criteria is very sensitive to overfitting. As discussed previously, a potential solution would be to collect more samples, but this is sometimes time-consuming and tiring to users. Furthermore, if we would like to use online datasets of previously recorded data, this solution may not be possible. Thus for large covariance matrices the number of observations is usually less than the number of variables.

Two main approaches have been suggested to address this problem. The first calls for the use of a regularized version of LDA. The second approach uses a variable selection procedure whereby the variables most useful for proper classification are selected.

The standard approach to parameterize the LDA prediction function uses the empirical calculation of sample SCMs from the tangent vectors. However, the empirical estimate of covariance becomes unstable when the number of variables is greater than the number of observations. That is, a systematic error occurs in which large eigenvalues are estimated too large and small eigenvalues are are estimated too small. This systematic error leads to reduced performance and LDA classification accuracies which are far from optimal. Therefore, the technique of shrinkage can be employed to produce better estimates of the covariance matrices.

Let $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$ be the set of d-dimensional feature vectors. Then the estimated mean is calculated in the usual way as:

$$\hat{\mu} = \frac{1}{n}\sum_{i=0}^{n} x_i$$

The empirical estimate of the covariance is then calculated as:

$$\hat{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu})(x_i - \hat{\mu})^T$$

To correct for the systematic error, we calculate the shrinkage estimate of the covariance as:

$$\tilde{\Sigma}(\gamma) = (1-\gamma)\hat{\Sigma} + \gamma\nu I \tag{4.8}$$

where $\gamma \in [0,1]$ is a tuning parameter and ν is the average eigenvalue $trace(\hat{\Sigma})/d$ where d is the feature space dimensionality. We can then exploit the positive semidefinite nature of $\tilde{\Sigma}$ to write its eigenvalue decomposition as $\tilde{\Sigma} = VDV^T$, where V are the othonormal eigevectors of $\tilde{\Sigma}$ and D is the diagonal matrix of eigenvalues. Inserting this definition into yields:

$$\tilde{\Sigma} = (1-\gamma)VDV^T + \gamma\nu I = (1-\gamma)VDV^T + \gamma\nu VIV^T = V((1-\gamma)D + \gamma\nu I)V^T$$

Thus, we see that $\hat{\Sigma}$ and $\tilde{\Sigma}$ have the same set of eigenvectors V and that the extreme eigenvalues are shrunk or elongated in the direction of the average ν. For $\gamma = 0$, we obtain the standard unregularized LDA, while $\gamma = 1$ assumes spherical covariances. The shrinkage procedure is shown geometrically in figure 4.2.

The optimal shrinkage parameter can be found using a cross-validation procedure. However, due to increased complexity and computational costs, Ledoit and Wolf [23] developed an analytical method to calculate the optimal shrinkage parameter for certain directions. The method basically aims to minimize the Frobenius norm between the shrunk covariance matrix and the unknown true covariance matrix Σ. This is accomplished by penalizing large sample-to-sample variances in the empirical covariance entries leading to stronger shrinkage.

We first define $(x_k)_i$ and $\hat{\mu}_i$ to be the $i-th$ element of the vectors x_k and μ, respectively,and define s_{ij} to be the element of the $i-th$ row and $j-th$ column of $\hat{\Sigma}$. We then define

$$z_{ij}(k) = ((x_k)_i - (\hat{\mu})_i)((x_k)_j - (\hat{\mu}_j)$$

The optimal shrinkage parameter towards identity as in equation 4.6.2 can be computed as suggested by Schäfer and Strimmer [38] as:

$$\gamma^* = \frac{n}{(n-1)^2}\frac{\sum_{i,j}^{d} var_k(z_{i,j}(k)}{\sum_{i\neq j} s_{ij}^2 + \sum_i (s_{ii} - \nu)^2} \tag{4.9}$$

Using the shrinkage estimate of covariance in the derivation of the LDA prediction function leads to what is called regularized LDA or shrinkage LDA. [5]

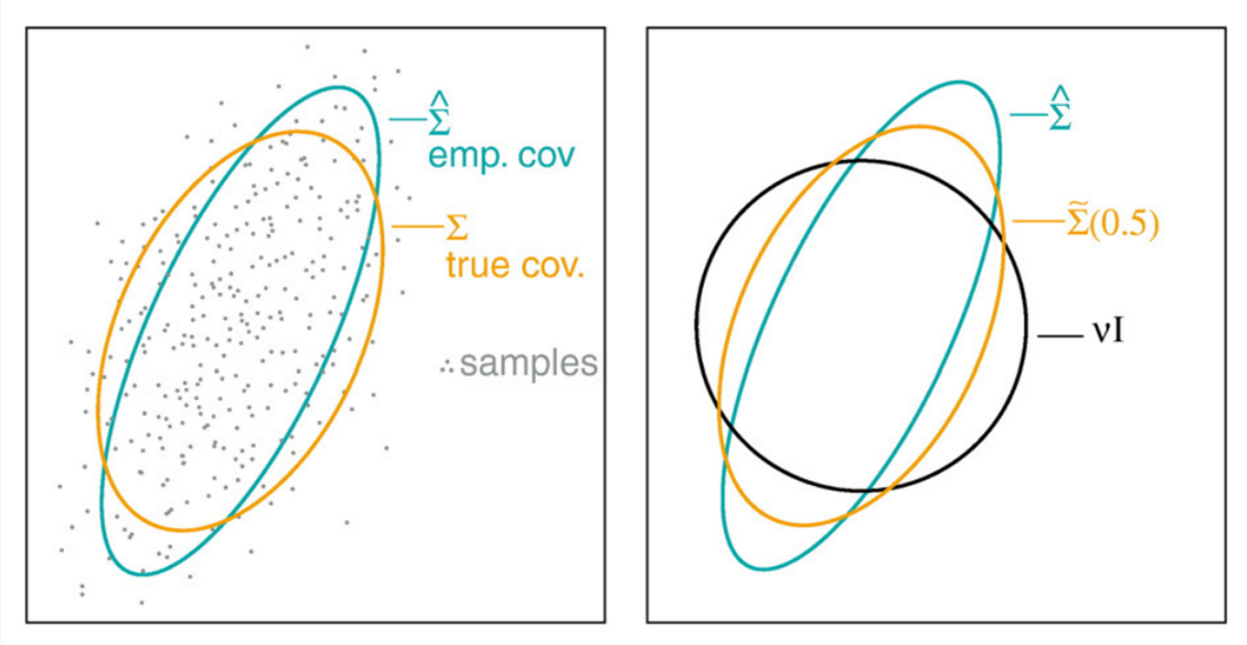

Figure 4.2: Covariance Shrinkage. [5] [fair use]

Once the shrinkage estimate of the covariance matrices associated with each class have been calculated, the standard procedure discussed previously.

The second approach to handle overfitting uses a variable selection procedure. To achieve four-class classification of a BCI competition dataset, Bacharant et al. [2] used one-way ANOVA to select the most discriminantive variables. First, they performed PCA in the tangent space to derive uncorrelated variables creating an orthogonalized tangent space S_0. Next, for each variable in S_0 they applied a one-way ANOVA. The variables could then be ranked according to their p-value and a weighted False Discovery Rate (FDR) was used to automatically select the minimal number of variables. Alternative approaches may be taken to variable selection, one of which will be explored later in the present work.

4.6.3 Fisher Geodesic Discriminant Analysis

While MDRM classification is a simple and relatively effective method, BCI performance can degrade when a test covariance matrix exists far from the two inter-class Riemannian means in a way that is not class-related. That is, the class-related information encoded in the Riemannian distance can be obscured by overlying noise. Therefore, it is preferable to perform some type of filtering operation before calculation of Riemannian distance for MDRM classification.

Fisher Geodesic Discriminant Analysis (FGDA) is supervised method of filtering which is another extension of LDA to the tangent space. In FGDA, the goal is to search for the

geodesics that support class-related information. Filtering is then performed along this line to discard of information not relevant for proper classification.

As in TSLDA, the first step is to calculate the Riemannian mean of the entire training set of sample SCMs. Next, all of the training covariance matrices are projected onto the tangent space at the Riemannian mean, and the Fisher Geodesic filters are derived using an LDA criterion. Like in TSLDA, we again face the problem of overfitting. Thus, we must either use the technique of shrinkage or some form of variable selection when the dimensionality of the tangent vectors is comparable to or exceeds the number of observations.

As discussed in the Fisher Linear Discriminant section, the Fisher LDA criterion is the maximization of the ratio of between-class scatter S_B to within-class scatter S_W. This problem can be solved with the eigenvalue decomposition of $S_W^{-1}S_B$, or the $eig(S_B, S_W)$ command in Matlab, yielding a set of eigenvectors and eigenvalues. The standard LDA approach is to retain the eigenvector corresponding to the highest eigenvalue and to use this as the weight vector in the LDA prediction function. In order to perform filtering, we instead retain up the K eigenvectors associated with the K highest eigenvalues and place them in a matrix $\tilde{W} \in \mathbb{R}^{n(n+1)/2 \times K}$. As suggested by [1], it is assumed that all class-related information is contained within the first five components so that $K \leq 5$.

Once the Fisher geodesic filter matrix is derived, we then may use it to filter test sample SCMs prior to MDRM classification. For filtering, which does not reduce the dimensionality of the problem, a least-squares estimate is used to compute the variation modes and projection. The filtering operation is defined to be:

$$\tilde{s_x} = \tilde{W}(\tilde{W}^T\tilde{W})^{-1}\tilde{W}^T s_x \tag{4.10}$$

where $\tilde{s_x}$ is the vectorized, filtered test tangent vector belonging to unknown class x and s_x is the vectorized, unfiltered test tangent vector.

After Fisher geodesic filtering, the filtered test matrix in the tangent space is projected back to the manifold. MDRM classification using intra-class Riemannian means of the training set and the filtered test matrix can then be performed as discussed the previous sections. It should be noted that retention of a single eigenvector in $\tilde{W}$ followed by filtering and MDRM classification leads to the same classification accuracies as performing LDA for classification in the tangent space without mapping back to the manifold as in the TSLDA method. The entire FGDA approach is shown geometrically in figure 4.3 for the case of 2×2 (3-dimensional SPD) covariance matrices generated using Wishart distributions for each class. [1]

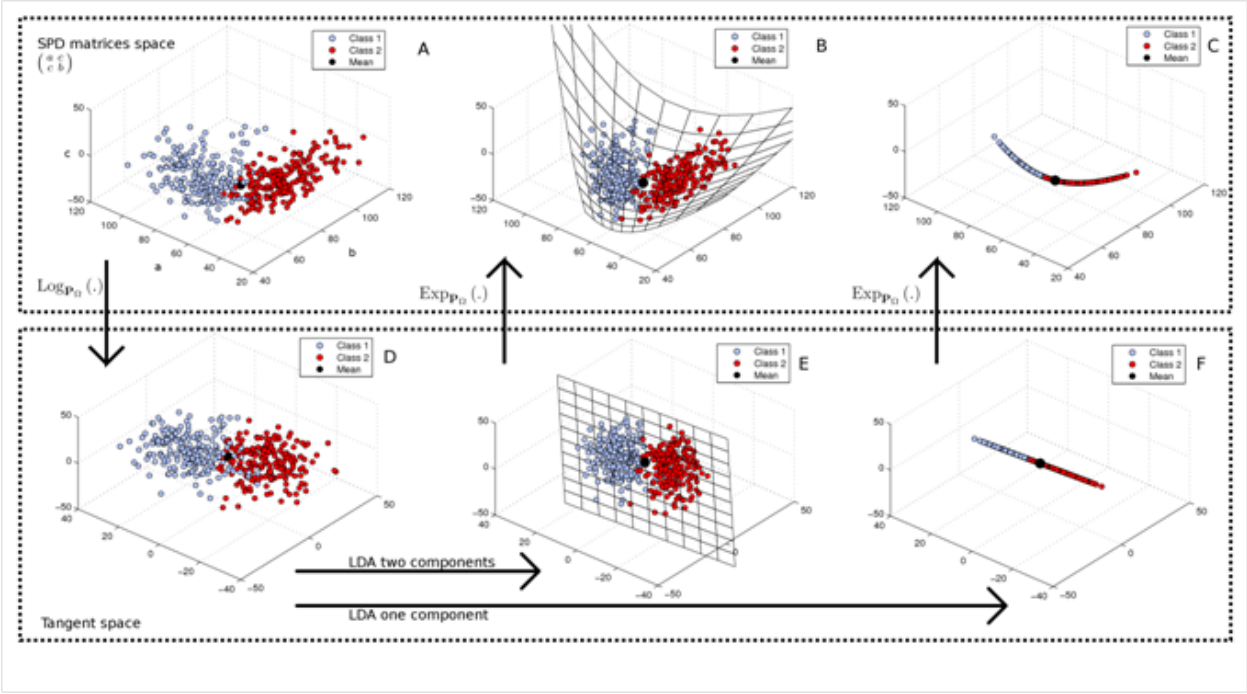

Figure 4.3: Fisher geodesic filtering operations. [1] [fair use]

Chapter 5

Numerical Experiments

5.1 Introduction

The goal was to compare several methods operating on a BCI competition dataset. The several variations of the CSP+LDA method were investigated. Additionally, several Riemannian geometry were considered. Finally, the two methods were combined to discover if BCI performance could be improved.

5.2 BCI Competition IV 2a Dataset

The goal of BCI competitions is to validate signal processing and classification methods for brain-computer interfaces. In BCI competitions, several high-quality datasets are provided for open access to the scientific community. Researchers can then apply a variety of signal processing and machine learning approaches to this data and compare their method to the methods of other researchers. During the actual competition, only the training data includes known true labels while the performance of the prediction algorithm operating on the testing data is not revealed until the end of the competition. The dataset used in the present study comes from BCI Competition IV and is the 2a dataset. [7]

The 2a dataset consists of EEG data from 9 subjects. When subjects were presented with a cue, they were asked to perform one of four motor imagery tasks: imagination of movement of left hand (class 1), right hand (class 2), both feet (class 3), or tongue (class 4). Each subject performed two sessions, on separate days, each consisting of 6 runs separated by short breaks. Each run consisted of 48 trials (12 trials of each of the four motor imagery tasks) yielding 288 trials per session. Additionally, 5 minutes of EOG recordings were collected before each session to potentially assist in artifact removal (two minutes of eyes open looking at the fixation cross, two minutes of eyes closed, and one minute with eye movements).

Subjects sat in a comfortable armchair in front of a computer screen. At the beginning of each trial (t=0s), a fixation cross appeared on the black screen and a short acoustic warning tone was presented. At t=2s, a cue in the form of an arrow pointing left, right, down, or up (corresponding to one of the four classes) was presented and stayed on the screen for 1.25s. The subject then performed the appropriate motor imagery task until the fixation cross disappeared at t=6s. A short break then ensued during which the screen was black again.

EEG data was recorded with twenty-two Ag/Ag-Cl electrodes distributed over the scalp. All channels were recorded monopolarly with the left mastoid serving as the reference and the right mastoid serving as ground. The signals were sampled at 250 Hz, bandpass filtered between 0.5 Hz and 100 Hz, and notch filtered at 50 Hz to suppress line noise. The sensitivity of the amplifier was set to 100 μV. Additionally, EOG data was recorded monopolarly from three electrodes placed around the eyes, sampled at 250 Hz, bandpass filtered between 0.5 Hz and 100 Hz, notch filtered at 50 Hz, and recorded with an amplifier sensitivity of 1 mV.

5.3 Present Study

In the competition, participants were asked to provide continuous classification (asynchronous) for each sample in form of labels 1-4. A confusion matrix would then be built for each time point across all artifact-free trials, and from the confusion matrices, the time course of the accuracy as well as the kappa coefficient would be obtained. However, departing from the original competition, for the present study only non-continuous classification (synchronous) was performed. That is, only one output was given or predicted for each trial taking into consideration only the window of data from 3.5 to 5.5 seconds after cue presentation. Furthermore, only the problem of binary classification of left hand or right hand imagined movement was considered. Thus, the goal of the present study was to compare the classification accuracy of several different methods for binary motor imagery classification of the 288 training and testing trials associated with left or right hand motor imagery collected over the two sessions.

For all of the methods tested, 22 channels of EEG training and testing data were loaded into Matlab, concatenated together, and bandpass filtered between 8 Hz and 30 Hz using a 5th-order Butterworth filter. It should be noted that any missing values were replaced with zeros prior to bandpass filtering. The bandpass filtered training data was then again separated from the bandpass filtered testing data. The training data and testing data were then epoched to extract out data 0.5 seconds to 2.5 seconds after a left or right hand cue. The sample spatial covariance matrix for each of the extracted trials was then calculated.

The performance of each method was evaluated by means of 10-fold cross-validation. That is, the covariance matrices of all 288 of the training and testing trials were concatenated together and split into 10 roughly equal folds. One fold was set aside as the test set while

the remaining nine folds were used to train the classifier. After classification accuracy was calculated, the test fold was replaced, and the process was repeated with the next fold. The process continued until each fold served as the test set. The 10 classification accuracies were then averaged to find an average classification accuracy for a given set up.

For several of the methods, CSP spatial filters were derived. The number of spatial filters to include in the spatial filter matrix were then varied and an average classification accuracy was calculated from 10 fold cross-validation for the different numbers of spatial filters. Graphs were generated which plotted the number of spatial filters as the abscissa and classification error as the ordinate. In this way, potential overfitting could be detected.

Finally, the processing time of each method was found as an additional measure of performance. It is hypothesized that prior CSP spatial filtering with Riemannian geometry classification methods will greatly reduce computation time.

5.4 CSP+LDA Reference Methods

Several variations of the CSP+LDA method were performed. In the standard method, the arithmetic mean was used to calculate the average covariance matrix for filter derivation. A second method instead used the Riemannian mean to calculate the average covariance matrix. For both of these methods, the spatial filters were selected according to either pairs of the highest and lowest eigenvalue eigenvectors or using the maximization of $|(\lambda_1)_j - 0.5|$. As a final reference method, as suggested by Blankertz et. al [6], 6 spatial filters were selected from either end of the eigenvalue spectrum, i.e. 3 eigevectors corresponding to the 3 highest eigenvalues and 3 eigenvectors corresponding to the 3 lowest eigenvalues.

5.4.1 Deriving and Selecting the CSP Spatial Filters

In the CSP+LDA method, first the training covariance matrices were separated by those associated with left hand cues and those associated with right hand cues. Then, either the arithmetic or Riemannian mean (see section 5.5.1 for details) of the set of covariance matrices was calculated for each class yielding P_1 and P_2, where P_1 is the mean of left hand covariance matrices and P_2 is the mean of right hand covariance matrices. Using the eigendecomposition as discussed in the common spatial patterns theory section (i.e., eig(P_1,P_1+P_2)), the 22-by-22 spatial filter matrix V was generated.

In the first method of spatial filter selection, the eigenvectors were ordered according to decreasing eigenvalue, and pairs of $J/2$ spatial filters corresponding to the highest and lowest eigenvalues were selected from V and placed into a 22-by-J spatial filter matrix W. In the second method of spatial filter selection, the eigenvectors were ordered according to decreasing values of $|(\lambda_1)_j - 0.5|$. The top J spatial filters were then extracted and placed

into W. Increasing numbers of spatial filters were included in W, calculating average cross-validation accuracy for each additional spatial filter or pair, until the full 22 was reached.

Thus, there were four methods for spatial filter derivation and selection. In order of increasing complexity, they were: arithmetic mean selecting pairs of from both ends of the eigenvalue spectrum, arithmetic mean selecting filters based on maximization of $|(\lambda_1)_j - 0.5|$, Riemannian mean selecting pairs of from both ends of the eigenvalue spectrum, and Riemannian mean selecting filters based on maximization of $|(\lambda_1)_j - 0.5|$.

5.4.2 Extracting the Log-Variance Features

The log-variance features of both the training epochs and the testing epochs were then to be extracted. The epochs were first multiplied by the spatial filter matrix W yielding J channel epochs. Next the variance of each of the J channels was calculated. Finally, the logarithm of these variances was taken. These operations yielded a set J-dimensional training feature vectors and testing feature vectors.

5.4.3 Training the LDA Classifier

Once the features associated with the training epochs which had known labels had been extracted, the LDA classifier was then to be trained. Using the equations discussed in the Fisher linear discriminant section, the weight vector and bias were calculated. Note that the scatter matrices S_ω were calculated by using the cov command in Matlab on the concatenated feature vectors of each class where the columns were the individual features.

5.4.4 Predicting the Labels of the Testing Data

The features belonging to the testing data were then input into a prediction function parameterized with the training data. The predicted labels were then compared to the true labels, and the classification accuracy was calculated.

5.5 Riemannian Geometry Reference Methods

In all of the Riemannian geometry reference methods, no CSP spatial filters were derived or used. Several different reference methods were explored in the present work, namely minimum Riemannian distance to Riemannian mean (MDRM), tangent space LDA (TSLDA), and minimum Riemannian distance to Riemannian mean following Fisher geodesic filtering (FGMDRM) using different numbers of Fisher geodesic (FG) filters. Because the TSLDA

and FGMDRM methods use an LDA criterion, they are very prone to overfitting. Thus, shrinkage (regularization) LDA was used to address this problem. Later in the proposed method, variable selection via CSP spatial filtering was used instead to prevent overfitting allowing the use of the standard LDA procedure.

5.5.1 Minimum Distance to Riemannian Mean

Calculating the Intra-class Riemannian Means

The first step in the MDRM method was to calculate the intra-class Riemannian mean covariance matrices for the training data. The training sample SCMs were separated by those associated with class 1 (left hand cues) and those associated with class 2 (right hand cues). The Riemannian mean of class k was first initialized to be the arithmetic mean. Next, each covariance matrix was mapped to the tangent space at the current mean and vectorized using equation 4.7 where the logarithmic matrix was computed with equation 4.2. These operations formed a set of s_i's, i.e. one for each trial in class k. The stopping rule was checked, and, if not met, the arithmetic mean s of the s_i's was calculated. This is a valid operation because the tangent space is a Euclidean space.

The estimated mean s was un-vectorized to form $S \in S(n)$, and, using the exponential map, was mapped back to the manifold. The process was repeated until the stopping rule was satisfied. The stopping rule used in the present study was that the absolute value of the squared Frobenius norm of the matrix T, with the set of s_i's as columns, minus the squared Frobenius norm of the previous T divided by squared Frobenius norm of previous T was less than some tolerance $\epsilon = 10^{-5}$.

Minimum Distance to Riemannian Mean Classification

Once the Riemannian means of class 1 and of class 2 were calculated, minimum Riemannian distance classification was performed. For each testing covariance matrix, the Riemannian distance from each of the Riemannian means was calculated using equation 4.3. The testing covariance matrix was assigned the label corresponding to the shorter Riemannian distance to the Riemannian mean of class 1 or 2.

5.5.2 Tangent Space LDA

The first step in the tangent space LDA (TSLDA) method was to calculate the Riemannian mean of the entire set of training covariances matrices belonging to the two classes. This was accomplished in an analogous manner as calculation of the intra-class Riemannian means in the MDRM method. That is, the Riemannian mean of the entire training set was initialized

as the Euclidean mean, mapped onto the tangent space at that mean, vectorized, averaged using the Euclidean mean, and mapped back to the manifold. The process was repeated until the same stopping rule was satisfied.

Once the Riemannian mean of the entire training set was calculated, every training covariance matrix was mapped onto the tangent space at the Riemannian mean and vectorized forming a set of 253-dimensional vectors. The class-related Euclidean means of the tangent vectors were then calculated forming m_1 and m_2. The Euclidean mean of m_1 and m_2 was then calculated forming m_{tot}. The between-class scatter S_B was then calculated in a similar manner to that discussed in the Fisher Linear Discriminant theory section.

In order to calculate the within-class scatter S_W, the class-related covariance matrices of the tangent space vectors needed to be estimated. Because the tangent space vectors had $p = 253$ variables with only about 130 observations, empirical estimators of covariance would be very unstable. Therefore, the method of shrinkage estimation was used [38] to form two 253×253 class-related covariance matrices. The within-class scatter was then calculated by taking the Euclidean mean of the two covariance matrices.

The next step was to parameterize the TSLDA prediction function. To derive the optimal weight vector w, the generalized eigenvalue decomposition of S_B and S_W with the $eig(S_B, S_W)$ command in Matlab. The eigenvector corresponding to the maximum eigenvalue was selected as the weight vector w. The bias b was calculated as $w^T m_{tot}$.

The testing covariance matrices were then projected onto the tangent space at the Riemannian mean of the training covariance matrices. They were then vectorized and concatenated forming a set of 253-dimensional testing tangent vectors with unknown label. The testing tangent vectors were then input into the prediction function to generate a set of predicted labels. The predicted labels were compared to the true labels and the classification accuracy was computed.

5.5.3 Fisher Geodesic Discriminant Analysis

The steps used in Fisher Geodesic Discriminant Analysis (FGDA) were exactly the same as in the TSLDA method up to the the generalized eigenvalue decomposition of S_B and S_W. In the FGDA method, however, different numbers of the eigenvectors (FG filters) corresponding to the highest eigenvalues were retained. In the present study, between two and five eigenvectors were retained and placed in a FG filter matrix $\tilde{W}$. The case of a single eigenvector in $\tilde{W}$ was not considered in the present study because it was verified to produce identical classification results as the TSLDA method.

The newly built FG filter matrix could then be used to filter test covariance matrices in the tangent space. To accomplish this task, the test covariance matrices with unknown labels were first projected into the tangent space at the Riemannian mean of the entire training set and vectorized. Equation 4.10 was then used to filter the test covariance matrices. The

filtered test covariance matrices were then mapped back to the manifold. Using the standard MDRM procedure, each filtered test covariance matrix was assigned the label corresponding to the shorter Riemannian distance to the Riemannian mean of class 1 or class 2.

5.6 Proposed Method: CSP Spatial Filtering with Riemannian Geometry Classification

5.6.1 Introduction

In the proposed method, CSP spatial filtering was done prior to calculation of the sample spatial covariance matrices (SCMs) that were classified using Riemannian geometry methods. The CSP spatial filtering operation acts as a sort of variable selection procedure because the CSP method entails a selection of a subset of spatial filters. Thus, the sample SCMs are of lower dimensionality as are their associated tangent vectors. This allows the use of the standard LDA method so that shrinkage is no longer used in the proposed methods.

It should be noted that this form of variable selection is not continuous, but rather makes increasing jumps of $J+1$ as additional spatial filters are retained. Thus, the tangent vector dimensionality varies from 1, 3, 6, 10, 15, 21,...,253 as the number of spatial filters varies from 1 to 22. It should also be noted that because the sample SCMs can be of lower dimensionality, processing speed may be greatly reduced when fewer numbers of spatial filters are used.

5.6.2 Deriving and Selecting the CSP Spatial Filters

Two methods of deriving CSP spatial filters were completed for comparison. In the first method, the average sample SCM for each class was calculated using the arithmetic mean as in reference method. For the second method, the Riemannian mean of the sample SCMs for each class was calculated. Using these means, P_1 and P_2, the *eig* function in Matlab was used to calculate full W matrices for each method.

Furthermore, two methods of spatial filter selection were used. In the first method, pairs of spatial filters corresponding to the largest and smallest eigenvalues were selected for inclusion in W. In the second method, spatial filters with corresponding eigenvalues that maximized $|(\lambda_1)_j - 0.5|$ were selected first. Spatial filters were added to W until the full 22-by-22 matrix was constructed. As discussed previously, these represented the four different methods of CSP spatial filter derivation and selection.

5.6.3 Extracting the Sample Spatial Covariance Matrix Features

Once the spatial matrix W was constructed, the band pass filtered training and testing epochs were multiplied by it potentially reducing the dimensionality of the data. The training and testing sample SCMs were then calculated from the spatially filtered epochs. For spatially filtered data that consisted of six spatially filtered time series, for example, this operation yielded 6×6 sample SCMs. Without the spatial filtering step, the unfiltered 22 channel epochs yield 22×22 sample SCMs. Thus, the additional spatial filtering step allows one to manipulate smaller dimension sample SCMs reducing the computational cost.

5.6.4 Minimum Distance to Riemannian Mean with CSP Spatial Filtering

The intra-class Riemannian mean of the training covariance matrices of each class was then calculated using the methods discussed in the Riemannian Geometry Method section. The Riemannian distance of each testing covariance matrix to each intra-class Riemannian mean was then calculated using the methods described previously. The testing covariance matrix was assigned the label corresponding to the shorter Riemannian distance.

5.6.5 Tangent Space LDA with CSP Spatial Filtering

Following CSP spatial filtering, the Riemannian mean of the entire training set of $J \times J$ sample SCMs was found. Next, the training sample SCMs were projected onto the tangent space at the Riemannanian mean to generate a set of feature vectors associated with class 1 or class 2. The standard LDA approach as discussed in the Fisher Linear Discriminant theory section was then followed. It should be noted that shrinkage calculation of the covariance matrices needed for the within-class scatter was not done as it was assumed that overfitting would not be a problem for lower dimensionality feature vectors. Thus, only the empirical calculation of covariance was used.

5.6.6 MDRM Classification with Fisher Geodesic Filtering and CSP Spatial Filtering

As before, the steps in the FGMDRM method with CSP spatial filtering follow the TSLDA method very closely up to the classification step. Again using the empirical estimate of covariance, the within-class scatter was calculated, and the subsequent eigendecomposition of $S_W^{-1}S_B$ yielded a set of $n(n+1)/2$ eigenvectors where n is the size of the sample SCMs.

The eigenvector associated with the highest eigenvalue served as the optimal weight vector

in the TSLDA method, but in the FGMDRM method, filtering using several of these eigenvectors was desired. Thus, different numbers of eigenvectors were included in $\tilde{W}$ from 2 to 5. It should be noted that for the case that $n = 1$, FGMDRM using 2 to 5 eigenvectors was not possible. Likewise for $n = 2$, only FGMDRM using 2 or 3 eigenvectors was possible.

Next, the testing sample SCMs were projected onto the tangent space at the Riemannian mean of the entire training set and vectorized. The testing tangent vectors were filtered with the least-squares operation as in equation 4.10. The FG filtered testing tangent vectors were then projected back to the manifold where standard MDRM classification was completed.

5.6.7 Mean Classification Time

Riemannian classification methods have been traditionally associated with longer computation times which may obviate there use for modest increases in mean classification accuracy for binary classification beyond the simpler CSP+LDA methods. Thus, it is of interest to investigate whether a significant reduction in computation time could be realized by prior CSP filtering.

For an average performer, subject 1, the mean time to classify a single test trial was calculated for several methods. The time taken to bandpass filter the testing data and extract the testing epochs was not factored in as it was required for all methods and not relevant for comparison. Thus, the number of CSP spatial filters using the Riemannian mean and the $|(\lambda_1)_j - 0.5|$ criterion for selection was varied and the mean single trial classification time was calculated for the final fold using the same 10-fold cross-validation set up.

5.7 Results

5.7.1 Evaluating for Overfitting

To begin, the accuracy of each classification algorithm on the training set and the testing set was plotted against the number of CSP spatial filters to evaluate for overfitting as shown in figure 5.1. Overfitting was determined by a training accuracy that was trending upwards towards 100% and a testing accuracy trending downward so that the two began to diverge. The same general trends were noted for all methods of CSP spatial filter derivation and selection for all subjects. The CSP+LDA methods tended to slightly overfit with increasing numbers of CSP spatial filters, while the CSP+TSLDA and the CSP+FG filtering+MDRM methods faced much greater problems with overfitting for increasing numbers of CSP spatial filters. The CSP+MDRM methods did not appear to overfit with increasing numbers of CSP spatial filters.

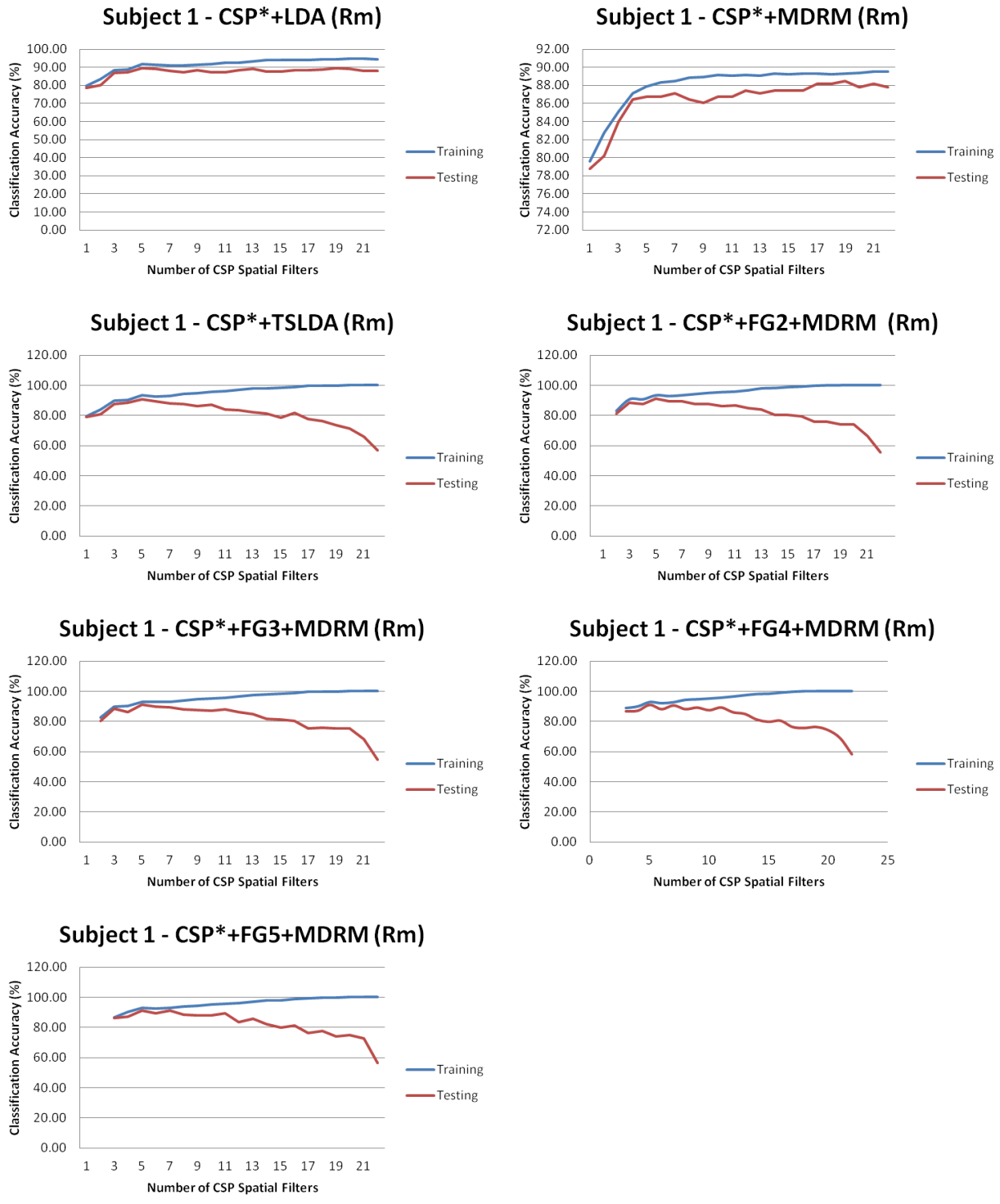

Figure 5.1: Testing for Overfitting

5.7.2 Classification Accuracy

The performance of each method was evaluated by 10-fold cross-validation. In all of the following tables in this section, CSP spatial filter derivation using the arithmetic mean is designated by (Ar) while derivation using the Riemannian mean is designated by (Rm). Spatial filter selection using the maximization of $|(\lambda_1)_j - 0.5|$ is designated by CSP* while the alternative selection criterion is designated CSP. Furthermore, for all methods using CSP spatial filtering, the optimal number of CSP spatial filters for a particular subject and a particular will be shown in parenthesis beside the classification accuracy.

The results of the CSP+LDA methods can be seen in table 5.1. Using the optimal number of spatial filters found by cross-validation outperformed the reference method of using 6 spatial filters derived using the arithmetic mean for all methods. The criteria of spatial filter selection using the maximization of $|(\lambda_1)_j - 0.5|$ outperformed the criteria of using pairs of highest and lowest eigenvalues for both derivation with the arithmetic mean and the Riemannian mean. It was found that the best mean classification accuracy in 10-fold cross validation could be achieved using the optimal number of spatial filters for each subject, the Riemannian mean, and the $|(\lambda_1)_j - 0.5|$ criteria.

Table 5.1: Comparison of CSP+LDA methods.

CSP+LDA Methods - Average 10-Fold Cross-Validation Classification Accuracy					
User	CSP(6)+LDA (Ar)	CSP+LDA (Ar)	CSP*+LDA (Ar)	CSP+LDA (Rm)	CSP*+LDA (Rm)
S1	88.47	91.65 (12)	91.27 (13)	89.53 (8)	89.54 (6)
S2	55.97	60.1 (20)	58.66 (12)	62.54 (20)	61.5 (16)
S3	96.9	97.59 (4)	97.59 (6)	97.57 (14)	97.57 (14)
S4	69.05	72.19 (10)	75.31 (9)	72.2 (12)	72.19 (9)
S5	56.91	63.53 (22)	65.26 (21)	62.5 (2)	64.57 (6)
S6	62.52	71.9 (22)	72.6 (16)	74.62 (2)	75.32 (5)
S7	83.33	83.66 (4)	84.03 (6)	83.34 (4)	83.68 (5)
S8	98.28	98.28 (4)	98.28 (8)	98.28 (6)	98.28 (6)
S9	94.73	94.74 (4)	94.38 (3)	94.4 (2)	95.07 (7)
Avg	78.46	81.51	81.93	81.66	81.97

Table 5.2 shows the results of the MDRM experiments. All methods of CSP spatial filter derivation and selection outperformed the reference method when the subject-specific optimal number of CSP spatial filters was used. Using the Riemannian mean for filter derivation outperformed using the arithmetic mean. For both the arithmetic mean and Riemannian mean methods for filter derivation, filter selection by taking pairs of filters from either end of the eigenvalue spectrum outperformed the alternative selection method.

The results of the TSLDA methods are shown in table 5.3. All of the combined CSP TSLDA methods outperformed the standard regularized TSLDA. Both methods of Riemannian mean filter derivation outperformed both methods using the arithmetic mean. The $|(\lambda_1)_j - 0.5|$ criteria of filter selection outperformed the alternative method for filter selection for both the Riemannian and arithmetic mean.

The results for using FG filtering prior to MDRM classification are shown in tables 5.4, 5.5, 5.6, and 5.7 where the different tables represent using 2 through 5 eigenvectors derived using

Table 5.2: Comparison of MDRM methods.

MDRM Methods - Average 10-Fold Cross-Validation Classification Accuracy					
User	MDRM Ref	CSP+MDRM (AR)	CSP*+MDRM (AR)	CSP+MDRM (Rm)	CSP*+MDRM (Rm)
S1	87.8	88.83 (16)	88.14 (16)	88.14 (18)	88.49 (19)
S2	56.95	58.33 (16)	58.68 (20)	58.62 (4)	59.66 (3)
S3	93.74	96.17 (8)	95.83 (5)	94.78 (2)	94.78 (2)
S4	67.64	69.41 (8)	69.03 (8)	70.06 (6)	68.33 (21)
S5	57.96	61.11 (20)	61.06 (9)	63.51 (2)	61.76 (4)
S6	68.73	71.87 (12)	69.42 (10)	73.93 (2)	71.51 (4)
S7	79.17	80.9 (10)	80.55 (4)	82.66 (4)	82.32 (4)
S8	95.85	96.19 (10)	96.19 (9)	96.54 (2)	96.54 (2)
S9	89.17	94.05 (2)	93.69 (1)	94.05 (2)	93.69 (1)
Avg	77.45	79.65	79.18	80.25	79.67

Table 5.3: Comparison of TSLDA methods.

TSLDA Methods - Average 10-Fold Cross-Validation Classification Accuracy					
User	Reg TSLDA	CSP+TSLDA (AR)	CSP*+TSLDA (AR)	CSP+TSLDA (Rm)	CSP*+TSLDA (Rm)
S1	90.59	89.89 (6)	89.91 (7)	89.54 (6)	90.94 (5)
S2	57.56	62.46 (8)	62.09 (15)	64.89 (6)	65.92 (6)
S3	98.28	95.85 (4)	95.85 (5)	97.23 (4)	97.23 (3)
S4	69.75	73.56 (4)	75.32 (8)	72.14 (4)	72.89 (9)
S5	64.93	63.52 (12)	63.88 (15)	66.26 (14)	66.61 (5)
S6	70.50	66.31 (12)	71.86 (11)	75.31 (4)	73.57 (4)
S7	79.78	86.08 (6)	85.74 (9)	85.73 (6)	85.41 (5)
S8	96.55	97.59 (2)	97.59 (2)	97.93 (2)	97.93 (2)
S9	94.41	94.4 (2)	93.69 (12)	94.4 (2)	93.69 (1)
Avg	80.26	81.07	81.77	82.60	82.69

the LDA criterion. For all of the different numbers of FG filters, all combined CSP filtering methods outperformed the standard regularized FG filtering method. The Riemannian mean derivation of the CSP spatial filters outperformed the arithmetic mean for all methods, and the $|(\lambda_1)_j - 0.5|$ criterion for spatial filter selection outperformed the alternative selection method for all methods.

Table 5.4: Comparison of MDRM methods with prior Fisher geodesic filtering using 2 FG filters.

FG2 MDRM Methods - Average 10-Fold Cross-Validation Classification Accuracy					
User	Reg FG2	CSP+FG2+MDRM (AR)	CSP*+FG2+MDRM (AR)	CSP+FG2+MDRM (Rm)	CSP*+FG2+MDRM (Rm)
S1	90.59	88.85 (8)	90.25 (7)	89.53 (6)	91.31 (5)
S2	57.55	62.46 (8)	61.06 (9)	65.58 (6)	65.58 (6)
S3	97.93	95.85 (4)	96.19 (5)	96.18 (2)	97.23 (3)
S4	69.41	73.57 (4)	75.67 (8)	73.19 (4)	72.91 (8)
S5	64.57	63.52 (12)	63.87 (12)	67.65 (14)	67.32 (5)
S6	69.80	67 (12)	71.18 (10)	74.62 (4)	73.23 (4)
S7	79.78	85.41 (4)	85.05 (9)	86.08 (6)	85.07 (5)
S8	96.55	97.93 (2)	97.93 (2)	97.59 (2)	98.62 (2)
S9	94.06	94.4 (2)	93.36 (3)	94.05 (2)	93.69 (2)
Avg	80.03	81	81.62	82.72	82.77

A summary of the results of several reference methods as well as the top performing proposed methods is displayed in table 5.8. The standard MDRM method without CSP spatial filtering or Fisher geodesic filtering performed the worst with a mean classification accuracy of 77.45% in 10-fold cross-validation. The CSP method using 6 spatial filters (3 from either end of the eigenvalue spectrum) derived using the arithmetic mean, which is one of the most popular methods in the BCI field, performed the second worst with 78.46% accuracy. The regularized TSLDA approach and regularized Fisher geodesic filtering approach with 2 Fisher geodesic filters and MDRM classification had a similar performance of around 80%. Although the

Table 5.5: Comparison of MDRM methods with prior Fisher geodesic filtering using 3 FG filters.

FG3 MDRM Methods - Average 10-Fold Cross-Validation Classification Accuracy					
User	Reg FG3	CSP+FG3+MDRM (AR)	CSP*+FG3+MDRM (AR)	CSP+FG3+MDRM (Rm)	CSP*+FG3+MDRM (Rm)
S1	90.25	89.2 (8)	89.22 (8)	89.89 (6)	91.31 (5)
S2	58.24	64.2 (8)	60.09 (16)	64.57 (8)	64.88 (6)
S3	97.93	95.85 (4)	96.19 (5)	96.88 (14)	96.19 (3)
S4	69.41	73.24 (4)	76.01 (8)	71.81 (4)	73.94 (9)
S5	64.57	63.15 (10)	62.86 (9)	67.99 (14)	67.34 (5)
S6	69.80	66.98 (12)	71.18 (10)	74.29 (4)	73.24 (4)
S7	79.78	85.75 (8)	85.04 (6)	85.04 (6)	85.05 (7)
S8	96.55	97.24 (6)	97.23 (8)	97.59 (6)	98.28 (3)
S9	94.06	94.05 (2)	93.69 (1)	94.05 (2)	93.36 (10)
Avg	80.07	81.07	81.28	82.46	82.62

Table 5.6: Comparison of MDRM methods with prior Fisher geodesic filtering using 4 FG filters.

FG4 MDRM Methods - Average 10-Fold Cross-Validation Classification Accuracy					
User	Reg FG4	CSP+FG4+MDRM (AR)	CSP*+FG4+MDRM (AR)	CSP+FG4+MDRM (Rm)	CSP*+FG4+MDRM (Rm)
S1	90.60	89.89 (6)	89.57 (7)	89.89 (6)	90.97 (5)
S2	58.94	62.14 (8)	61.07 (15)	64.9 (8)	64.51 (4)
S3	97.93	95.14 (12)	96.54 (5)	96.87 (14)	96.18 (3)
S4	70.80	72.89 (4)	76.01 (8)	71.82 (4)	73.58 (8)
S5	64.91	63.5 (10)	63.19 (9)	67.65 (14)	67.7 (5)
S6	70.16	66.98 (12)	70.14 (11)	73.24 (4)	75 (4)
S7	80.12	85.75 (10)	84.72 (10)	85.73 (6)	84 (6)
S8	96.55	97.59 (6)	97.59 (7)	97.59 (6)	98.28 (3)
S9	94.06	93.35 (4)	93.71 (11)	93.35 (4)	93.36 (10)
Avg	80.45	80.8	81.39	82.34	82.62

Table 5.7: Comparison of MDRM methods with prior Fisher geodesic filtering using 5 FG filters.

FG5 MDRM Methods - Average 10-Fold Cross-Validation Classification Accuracy					
User	Reg FG5	CSP+FG5+MDRM (AR)	CSP*+FG5+MDRM (AR)	CSP+FG5+MDRM (Rm)	CSP*+FG5+MDRM (Rm)
S1	90.25	88.89 (6)	89.89 (7)	88.87 (10)	91.27 (7)
S2	58.95	61.83 (6)	60.71 (15)	63.88 (8)	64.56 (6)
S3	97.93	95.83 (12)	96.19 (5)	95.84 (14)	95.84 (3)
S4	70.80	72.19 (4)	74.24 (5)	71.82 (4)	73.57 (8)
S5	64.22	63.85 (10)	64.59 (17)	67.99 (14)	67.01 (5)
S6	70.85	67.33 (12)	72.2 (10)	73.24 (4)	73.62 (4)
S7	80.47	85.06 (4)	85.74 (10)	86.07 (8)	85.75 (7)
S8	96.90	97.57 (6)	97.59 (7)	97.24 (6)	97.59 (3)
S9	93.72	93.69 (4)	93.71 (11)	93.69 (6)	93.37 (11)
Avg	80.45	80.7	81.65	82.07	82.51

regularized Fisher geodesic filtering approach using 4 or 5 filters performed slightly better than that using 2, the 2 filter approach is shown for a more fair compairson with the top performing Fisher geodesic proposed method. The CSP+LDA method using the Riemannian mean and $|(\lambda_1)_j - 0.5|$ criterion performed the best of the reference methods with 81.97% accuracy.

Of the proposed methods, CSP+MDRM performed the worst and performed worse than the CSP*+LDA using the Riemannian mean and regularized TSLDA reference methods with a mean classification accuracy of 80.25%. The CSP*+TSLDA and the CSP*+FG2+MDRM, both using the Riemannian mean, outperformed all of the reference methods with mean classification accuracies of 82.69% and 82.77%, respectively.

Table 5.8: Comparison of several reference methods with some of top performing proposed methods.

	Reference Methods					Proposed Methods		
User	CSP(6)+LDA (Ar)	CSP*+LDA (Rm)	MDRM	Reg TSLDA	Reg FG2	CSP+MDRM (Rm)	CSP*+TSLDA (Rm)	CSP*+FG2 (Rm)
S1	88.47	89.54 (6)	87.8	90.59	90.59	88.14 (18)	90.94 (5)	91.31 (5)
S2	55.97	61.5 (16)	56.95	57.56	57.55	58.62 (4)	65.92 (6)	65.58 (6)
S3	96.9	97.57 (14)	93.74	98.28	97.93	94.78 (2)	97.23 (3)	97.23 (3)
S4	69.05	72.19 (9)	67.64	69.75	69.41	70.06 (6)	72.89 (9)	72.91 (8)
S5	56.91	64.57 (6)	57.96	64.93	64.57	63.51 (2)	66.61 (5)	67.32 (5)
S6	62.52	75.32 (5)	68.73	70.50	69.80	73.93 (2)	73.57 (4)	73.23 (4)
S7	83.33	83.68 (5)	79.17	79.78	79.78	82.66 (4)	85.41 (5)	85.07 (5)
S8	98.28	98.28 (6)	95.85	96.55	96.55	96.54 (2)	97.93 (2)	98.62 (2)
S9	94.73	95.07 (7)	89.17	94.41	94.06	94.05 (2)	93.69 (1)	93.69 (2)
Avg	78.46	81.97	77.45	80.26	80.03	80.25	82.69	82.77

The same information in table 5.8 is summarized in figure 5.2 with ± 1 standard deviation. All of the methods had a mean classification accuracy of around 80%. The popular CSP(6)+LDA (AR)method had the largest standard deviation, while the CSP*+TSLDA (Rm) and CSP*+FG2 (Rm) had the smallest standard deviations.

For an average performer, subject 1, the mean time to classify a single test trial was calculated for several methods as the number of CSP spatial filters was varied as shown in figure 5.3. The mean classification time for the CSP+LDA method stayed relatively constant with increasing numbers of CSP filters. However, the mean classification time for CSP+MDRM and CSP+TSLDA both increased at about the same rate when additional spatial filters were used. The CSP+FG2+MDRM method saw the greatest effects as the computation time increased rapidly with additional spatial filters. The reference methods that did not use any CSP spatial filtering are plotted as well, and seemed to have mean classification times of their CSP-related methods when all CSP spatial filters were used.

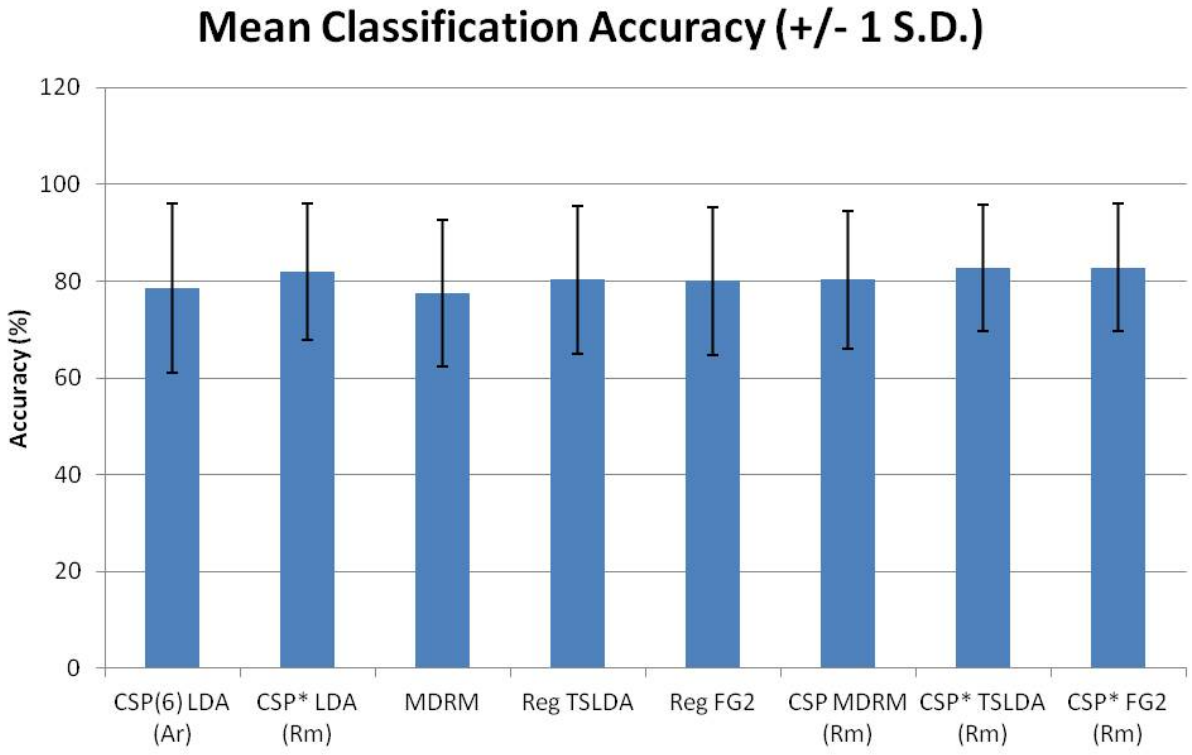

Figure 5.2: Mean classification accuracy in 10-fold cross-validation with ±1 standard deviation.

5.8 Discussion

5.8.1 Overfitting

When evaluating for overfitting, it was observed that the CSP+LDA methods tended to overfit the training data slightly as more CSP spatial filters were added. This result was expected as it has been previously described in [6]. The effect was slightly more pronounced when the Riemannian mean was used instead of the arithmetic mean. In general, beyond a certain number of spatial filters (about 6 in subject 1), no additional performance could be gained with additional spatial filters and the test accuracy remained relatively constant.

The MDRM methods did not tend to overfit with increasing numbers of spatial filters were used. For some users, the training accuracy did not improve with increase with increasing numbers of spatial filters but instead remained relatively constant or even decreased in some cases. In all cases, however, the testing accuracy tracked the training accuracy fairly closely in general.

The CSP+TSLDA and CSP+FG filtering+MDRM methods tended to be the most subject to overfitting with increasing numbers of spatial filters. This can be tracked back to the use

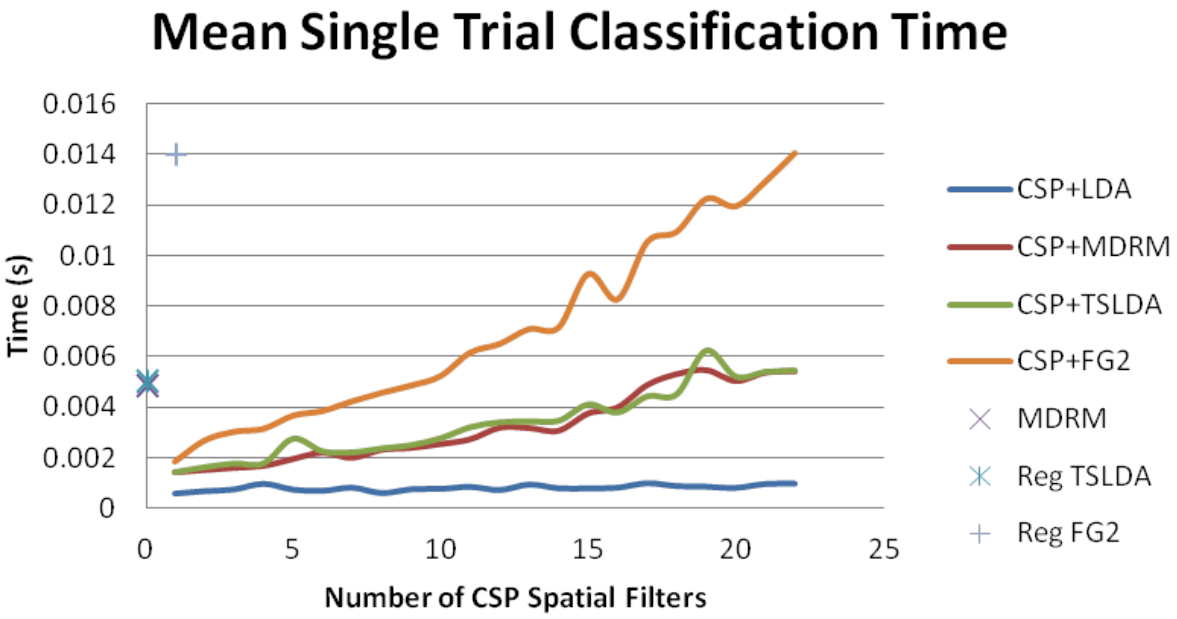

Figure 5.3: Mean single trial classification time as the number of CSP spatial filters is varied.

of an LDA criterion which is very prone to overfitting. The results were expected because all of these methods used the empirical calculation of covariance to calculate the within-class scatter. As more spatial filters were added, the dimensionality of the tangent vectors grew quickly, and once the dimensionality became comparable to or exceeded the number of observations, the empirical estimate of covariance became unstable and unreliable. A subsequent reduction in generalization capabilities was thus observed in classification of the test set.

5.8.2 Classification Accuracy

For all of the methods tested, the best classification accuracies were achieved using the Riemannian mean to derive the CSP spatial filters. This result was expected as the Riemannian mean of a set of covariances matrices is more precise as we treat them in their natural Riemannian space instead of as Euclidean objects. For all methods except the MDRM method, the best classification accuracies were achieved using the maximization of $|(\lambda_1)_j - 0.5|$. This result was expected because it allows designer to choose odd numbers of spatial filters which provides greater flexibility in the filter selection process.

The best performing reference method used CSP spatial filters derived with the Riemannian mean, selection by maximization of $|(\lambda_1)_j - 0.5|$, and standard LDA classification of the log-variance features. It achieved a very good mean performance of 81.97%. Although

the Riemannian mean is not typically used in most BCI systems, it's use in deriving CSP spatial filters and the subsequent improvement in classification accuracy has been reported previously in [3].

Using CSP spatial filtering prior to MDRM classification greatly improved the classification accuracy of the MDRM method from 77.45% to 80.25. Thus this proposed method performed better than the standard 6 CSP spatial filters with LDA classification of log-variance. Furthermore, it had comparable performance to the regularized TSLDA method and even improved on the regularized FG filtering method for certain numbers of FG filters.

The CSP*+TSLDA approach performed second best in terms of mean classification accuracy of all of the tested methods with 82.69%. It outperformed the standard 6 CSP spatial filter method by over 4% and improved the CSP*+LDA method by nearly 1%. Because the nature of variable selection by CSP spatial filtering was discontinuous, theoretically this method could be improved even more by selecting the most significant variables in the tangent space.

The CSP+FG filtering using 2 FG filters+MDRM method performed the best in terms of mean classification accuracy. Like the CSP*+TSDA method, it improved on the standard method by over 4% and improved on the CSP*+LDA method by nearly 1%. It had a comparable mean classification accuracy to the CSP*+TSLDA method improving upon it by only about 0.1%. Because this method required that we project the geodesic filtered tangent vectors back to the manifold for MDRM classification, the tangent vectors needed to stay in the space of symmetric matrices. Thus, no additional variable selection in the tangent space would be possible for this method.

Examining figure 5.2, it can be seen that the CSP(6)+LDA method had the largest standard deviation while the CSP*+TSLDA and CSP*+FG2 had the smallest. Looking back to table 5.8, this can be explained by the fact that the CSP*+TSLDA and CSP*+FG2 helped the poorest performing subjects significantly beyond the reference methods while there was a very slight drop in performance from some of the reference methods for the very best performing subjects. These are promising results as BCI illiteracy, the inability of some BCI users to achieve adequate performance, is a major unsolved problem in the BCI field.

5.8.3 Timing trials

The first thing to note about timing trials is that there was in fact a significant reduction in processing time in the Riemannian methods when combined with CSP spatial filtering. If we were to use six spatial filters for each of the Riemannian methods, we would observe a reduction of around 55%, 55%, and 73% in single trial classification time for the MDRM, TSLDA, and FG2+MDRM methods, respectively. It should be further noted by looking at 5.8 that we could still potentially do pretty well with even fewer CSP spatial filters in the Riemannian methods entailing an even larger reduction in classification time. It should

however be noted that all of the Riemannian methods using any number of CSP spatial filters did have an increased classification time beyond the CSP+LDA approach.

The next obvious feature noticed in the timing trials was that there is an increase in mean single trial classification time with increasing numbers of spatial filters for the combined CSP and Riemannian methods. The classification time CSP+TSLDA method and the CSP+MDRM method rose at approximately the same rate for additional CSP spatial filters. This was somewhat expected because both methods required similar operations; that is, a matrix inversion and diagonalization [2] to calculate Riemannian distance for MDRM or to project a test covariance matrix onto the tangent space for TSLDA. The classification time for the CSP+FG2+MDRM method was the highest of all methods tested for time trials. This was expected because this method required filtering of the test covariance matrix via a least-squares criterion in addition to the calculation of Riemannian distance for MDRM classification.

5.8.4 Conclusion

If one were to judge performance solely on classification accuracy, the CSP+FG2+MDRM method would be the recommended method very slightly outperforming the CSP+TSLDA. However, if classification time is the metric of choice in determining the best algorithm, one ought to choose the CSP*+LDA (Rm) method. Taking both of these factors into account, the author recommends the use of the CSP+TSLDA method because it offers the best compromise between improved classification accuracy and a shorter classification time than the FG filtering+MDRM methods. Furthermore, it should be noted that it may still be possible to squeeze more performance out of the TSLDA method if an additional variable selection procedure was used in the tangent space.

Chapter 6

Future Directions

Future work includes solving the problem of discontinuous numbers of variables selected for use in the TSLDA approach. One possible solution would be to first use cross-validation to discover when additional spatial filters leads to overfitting. Then, the approach of Bacharant et. al [2] may be followed by using a one-way ANOVA to rank variables according to p-value, and then using a weighted False Discovery Rate to automatically select the most discriminative variables.

Future work also includes moving beyond the intellectual curiosity of classifying discrete body part movement imagery onto real applications. The author envisions a system in which users could use a robotic BCI in a semi-autonomous fashion where high-level control would be taken over by the system. Incorporating a vision system could theoretically allow a user using a robotic arm to imagine picking up an object and classification of that command, possibly via a predicted end-effector trajectory, using techniques such as those derived in the present work. Further incorporation of encoders, strain gauges, and tactile sensors could allow real-time feedback to the BCI system to correct for any perturbations in the environment to accomplish the desired task. Although somewhat of an ambitious vision, the possibilities of such advancements in BCI technology are exciting and intriguing.

Bibliography

[1] Alexandre Barachant, Stéphane Bonnet, Marco Congedo, and Christian Jutten. Riemannian geometry applied to bci classification. In *Latent Variable Analysis and Signal Separation*, pages 629–636. Springer, 2010.

[2] Alexandre Barachant, Stéphane Bonnet, Marco Congedo, and Christian Jutten. Multiclass brain–computer interface classification by riemannian geometry. *Biomedical Engineering, IEEE Transactions on*, 59(4):920–928, 2012.

[3] Alexandre Barachant, Stphane Bonnet, Marco Congedo, and Christian Jutten. Common spatial pattern revisited by riemannian geometry. In *Multimedia Signal Processing (MMSP), 2010 IEEE International Workshop on*, pages 472–476. IEEE, 2010.

[4] Hans Berger. Über das elektrenkephalogramm des menschen. *European Archives of Psychiatry and Clinical Neuroscience*, 87(1):527–570, 1929.

[5] Benjamin Blankertz, Steven Lemm, Matthias Treder, Stefan Haufe, and Klaus-Robert Müller. Single-trial analysis and classification of erp componentsa tutorial. *NeuroImage*, 56(2):814–825, 2011.

[6] Benjamin Blankertz, Ryota Tomioka, Steven Lemm, Motoaki Kawanabe, and Klaus-Robert Muller. Optimizing spatial filters for robust eeg single-trial analysis. *Signal Processing Magazine, IEEE*, 25(1):41–56, 2008.

[7] C Brunner, R Leeb, G Müller-Putz, A Schlögl, and G Pfurtscheller. Bci competition 2008–graz data set a. *Institute for Knowledge Discovery (Laboratory of Brain-Computer Interfaces), Graz University of Technology*, page 16, 2008.

[8] Febo Cincotti, Donatella Mattia, Fabio Aloise, Simona Bufalari, Gerwin Schalk, Giuseppe Oriolo, Andrea Cherubini, Maria Grazia Marciani, and Fabio Babiloni. Non-invasive brain–computer interface system: towards its application as assistive technology. *Brain research bulletin*, 75(6):796–803, 2008.

[9] Jennifer L Collinger, Brian Wodlinger, John E Downey, Wei Wang, Elizabeth C Tyler-Kabara, Douglas J Weber, Angus JC McMorland, Meel Velliste, Michael L Boninger,

and Andrew B Schwartz. High-performance neuroprosthetic control by an individual with tetraplegia. *The Lancet*, 381(9866):557–564, 2013.

[10] An H Do, Po T Wang, Christine E King, Sophia N Chun, and Zoran Nenadic. Brain-computer interface controlled robotic gait orthosis. *Journal of neuroengineering and rehabilitation*, 10(1):1, 2013.

[11] Emanuel Donchin, Kevin M Spencer, and Ranjith Wijesinghe. The mental prosthesis: assessing the speed of a p300-based brain-computer interface. *Rehabilitation Engineering, IEEE Transactions on*, 8(2):174–179, 2000.

[12] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[13] Thomas Elbert, Brigitte Rockstroh, Werner Lutzenberger, and Niels Birbaumer. Biofeedback of slow cortical potentials. i. *Electroencephalography and Clinical Neurophysiology*, 48(3):293–301, 1980.

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

[15] Eberhard E Fetz. Operant conditioning of cortical unit activity. *Science*, 163(3870):955–958, 1969.

[16] P Thomas Fletcher and Sarang Joshi. Principal geodesic analysis on symmetric spaces: Statistics of diffusion tensors. In *Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis*, pages 87–98. Springer, 2004.

[17] Elisabeth VC Friedrich, Dennis J McFarland, Christa Neuper, Theresa M Vaughan, Peter Brunner, and Jonathan R Wolpaw. A scanning protocol for a sensorimotor rhythm-based brain–computer interface. *Biological psychology*, 80(2):169–175, 2009.

[18] Bernhard Graimann, Brendan Allison, and Gert Pfurtscheller. Brain–computer interfaces: A gentle introduction. In *Brain-Computer Interfaces*, pages 1–27. Springer, 2009.

[19] Chengalvarayan Radhakrishnamurthy Hema, MP Paulraj, Sazali Yaacob, Abdul Hamid Adom, and R Nagarajan. Asynchronous brain machine interface-based control of a wheelchair. In *Software Tools and Algorithms for Biological Systems*, pages 565–572. Springer, 2011.

[20] Aleksander Kostov and Mark Polak. Parallel man-machine training in development of eeg-based cursor control. *Rehabilitation Engineering, IEEE Transactions on*, 8(2):203–205, 2000.

[21] Christian Kothe. Introduction to modern brain-computer interface design - lecture 7: Oscillatory processes, June 2014.

[22] Karl LaFleur, Kaitlin Cassady, Alexander Doud, Kaleb Shades, Eitan Rogin, and Bin He. Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain–computer interface. *Journal of neural engineering*, 10(4):046003, 2013.

[23] Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2):365–411, 2004.

[24] Robert Leeb, Doron Friedman, Gernot R Müller-Putz, Reinhold Scherer, Mel Slater, and Gert Pfurtscheller. Self-paced (asynchronous) bci control of a wheelchair in virtual environments: a case study with a tetraplegic. *Computational intelligence and neuroscience*, 2007, 2007.

[25] Yuanqing Li, Kai Keng Ang, and Cuntai Guan. Digital signal processing and machine learning. In *Brain-Computer Interfaces*, pages 305–330. Springer, 2009.

[26] Tangient LLC. Phantom sensations and perceptions, April 2012.

[27] Dennis J McFarland, Dean J Krusienski, William A Sarnacki, and Jonathan R Wolpaw. Emulation of computer mouse control with a noninvasive brain-computer interface. *Journal of neural engineering*, 5(2):101, 2008.

[28] Dennis J McFarland, Lynn M McCane, Stephen V David, and Jonathan R Wolpaw. Spatial filter selection for eeg-based communication. *Electroencephalography and clinical Neurophysiology*, 103(3):386–394, 1997.

[29] Dennis J McFarland, Gregory W Neat, Richard F Read, and Jonathan R Wolpaw. An eeg-based method for graded cursor control. *Psychobiology*, 21(1):77–81, 1993.

[30] Dennis J McFarland, William A Sarnacki, and Jonathan R Wolpaw. Brain–computer interface (bci) operation: optimizing information transfer rates. *Biological psychology*, 63(3):237–251, 2003.

[31] Dennis J McFarland, William A Sarnacki, and Jonathan R Wolpaw. Electroencephalographic (eeg) control of three-dimensional movement. *Journal of Neural Engineering*, 7(3):036007, 2010.

[32] Klaus-Robert Müller and Benjamin Blankertz. Toward noninvasive brain-computer interfaces. *IEEE Signal Processing Magazine*, 23(5):125–128, 2006.

[33] G Onose, C Grozea, A Anghelescu, C Daia, CJ Sinescu, AV Ciurea, T Spircu, A Mirea, I Andone, A Spânu, et al. On the feasibility of using motor imagery eeg-based brain–computer interface in chronic tetraplegics for assistive robotic arm control: a clinical test and long-term post-trial follow-up. *Spinal Cord*, 50(8):599–608, 2012.

[34] G Pfurtscheller, GR Müller-Putz, A Schlögl, B Graimann, R Scherer, R Leeb, C Brunner, C Keinrath, F Lee, G Townsend, et al. 15 years of bci research at graz university of technology: current projects. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(2):205–210, 2006.

[35] Gert Pfurtscheller, C Guger, G Müller, G Krausz, and C Neuper. Brain oscillations control hand orthosis in a tetraplegic. *Neuroscience letters*, 292(3):211–214, 2000.

[36] Gert Pfurtscheller, Gernot R Müller, Jörg Pfurtscheller, Hans Jürgen Gerner, and Rüdiger Rupp. thought–control of functional electrical stimulation to restore hand grasp in a patient with tetraplegia. *Neuroscience letters*, 351(1):33–36, 2003.

[37] Audrey S Royer, Alexander J Doud, Minn L Rose, and Bin He. Eeg control of a virtual helicopter in 3-dimensional space using intelligent control strategies. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 18(6):581–589, 2010.

[38] Juliane Schäfer and Korbinian Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical applications in genetics and molecular biology*, 4(1), 2005.

[39] Gerwin Schalk and Jürgen Mellinger. Brain sensors and signals. In *A Practical Guide to Brain–Computer Interfacing with BCI2000*, pages 9–35. Springer, 2010.

[40] Gerwin Schalk and Jürgen Mellinger. *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, 2010.

[41] Reinhold Scherer, Alois Schloegl, Felix Lee, Horst Bischof, Janez Janša, and Gert Pfurtscheller. The self-paced graz brain-computer interface: methods and applications. *Computational intelligence and neuroscience*, 2007, 2007.

[42] Alois Schlogl, Julien Kronegg, Jane E Huggins, and Steve G Mason. 19 evaluation criteria for bci research. *Toward brain-computer interfacing*, 2007.

[43] Reijo Takalo, Heli Hytti, and Heimo Ihalainen. Tutorial on univariate autoregressive spectral analysis. *Journal of clinical monitoring and computing*, 19(6):401–410, 2005.

[44] Meel Velliste, Sagi Perel, M Chance Spalding, Andrew S Whitford, and Andrew B Schwartz. Cortical control of a prosthetic arm for self-feeding. *Nature*, 453(7198):1098–1101, 2008.

[45] Jacques J Vidal. Real-time detection of brain events in eeg. *Proceedings of the IEEE*, 65(5):633–641, 1977.

[46] Jean-Jacques Vidal. Toward direct brain-computer communication. *Annual review of Biophysics and Bioengineering*, 2(1):157–180, 1973.

[47] Jonathan Wolpaw and Elizabeth Winter Wolpaw. *Brain-computer interfaces: principles and practice*. OUP USA, 2012.

[48] Jonathan R Wolpaw, Niels Birbaumer, Dennis J McFarland, Gert Pfurtscheller, and Theresa M Vaughan. Brain–computer interfaces for communication and control. *Clinical neurophysiology*, 113(6):767–791, 2002.

[49] Jonathan R Wolpaw and Chadwick B Boulay. Brain signals for brain–computer interfaces. In *Brain-Computer Interfaces*, pages 29–46. Springer, 2009.

[50] Jonathan R Wolpaw and Dennis J McFarland. Multichannel eeg-based brain-computer communication. *Electroencephalography and clinical Neurophysiology*, 90(6):444–449, 1994.

[51] Jonathan R Wolpaw and Dennis J McFarland. Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans. *Proceedings of the National Academy of Sciences of the United States of America*, 101(51):17849–17854, 2004.

[52] Jonathan R Wolpaw, Dennis J McFarland, Gregory W Neat, and Catherine A Forneris. An eeg-based brain-computer interface for cursor control. *Electroencephalography and clinical neurophysiology*, 78(3):252–259, 1991.

[53] Han Yuan and Bin He. Brain–computer interfaces using sensorimotor rhythms: current state and future perspectives. *Biomedical Engineering, IEEE Transactions on*, 61(5):1425–1435, 2014.