Master's Thesis

Comparison and Combination of Feature Extraction Methods for the Application in Brain-Computer Interfaces

Patrick Ofner

Institute for Knowledge Discovery
Graz University of Technology
Head of Institute: Univ.-Prof. Dr.phil. Christa Neuper

Supervisor: Dipl.-Ing. Dr.techn. Clemens Brunner

Graz, November 2010

Abstract

Classification accuracies of feature extraction methods – as used in sensory motor rhythm (SMR) based brain-computer interfaces (BCIs) – in combination with a two-class linear discriminant analysis classifier are compared in this master's thesis. An SMR BCI uses electroencephalographic (EEG) signals to detect movement imagination and controls certain devices (e.g. wheelchair, entertainment electronics). The following methods were compared: adaptive autoregressive parameters, band-power, phase locking value, time domain parameters (TDP), and Hjorth parameters. Furthermore, it was analyzed if a certain method combination improves the classification accuracy. A genetic algorithm was used to find optimal method parameters and an optimal method combination. For that purpose, a MATLAB toolbox was developed. TDP with a bipolar spatial filter yielded the best classification accuracies; an subject-specific (individual) method optimization and classifier training is superior to a general optimization/training; a combination of methods has no advantage in contrast to using only TDP.

Kurzfassung

In dieser Masterarbeit wurden die Klassifikationsgenauigkeiten von Merkmalsextraktionsmethoden, wie sie bei auf sensorimotorischen Rhythmen (SMR) basierten Brain-Computer Interfaces (BCIs) verwendet werden, verglichen. Als Klassifikator wurde eine lineare Diskriminanzanalyse für zwei Klassen eingesetzt. Ein SMR-BCI verwendet die Elektroenzephalografie (EEG) um Bewegungsvorstellungen zu erkennen und steuert damit bestimmte Geräte (z.B. Rollstuhl, Unterhaltungselektronik). Folgende Methoden wurden miteinander verglichen: adaptive autoregressive Parameter, Bandleistung, Phase Locking Value, Time Domain Parameters (TDP) und Hjorth Parameter. Weiters wurden diese Methoden miteinander kombiniert um zu untersuchen, ob dadurch die Klassifikationsgenauigkeit verbessert werden kann. Die optimalen Parameter der einzelnen Merkmalsextraktionsmethoden und die optimale Methodenkombination wurden mittels eines genetischen Algorithmus gefunden. Dafür wurde eigens eine MATLAB-Toolbox entwickelt. TDP (bipolar abgeleitet) lieferte die höchste Klassifikationsgenauigkeit. Eine individuelle Optimierung und ein individuelles Training führten zu höheren Klassifikationsgenauigkeiten als eine generelle Optimierung und ein generelles Training. Eine Kombination von Methoden brachte gegenüber TDP keine Vorteile.

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....
date (signature)

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ....
(Unterschrift)

Acknowledgments

I would like to thank my parents for their great support during my studies. Furthermore, I would like to thank Dipl.-Ing. Dr.techn. Clemens Brunner for fruitful discussions and hints, and I appreciate that he has spent a lot of his time supporting me with my master's thesis.

Contents

Chapter 1

Introduction

A brain-computer interface (BCI) is a system which allows a person to control a device with the brain [18]. This device could be e.g. a PC running a computer game or a fighter aircraft. However, disabled people gain the largest benefits from such a system: patients with the locked-in syndrome are able to communicate with people around them, paraplegics with no function in their four limbs (tetraplegia) can steer a wheelchair or artificial arms can be moved by a BCI system. Of course, nowadays all these things are only available in a prototype stage and used solely in research. Nevertheless, BCI research is progressing quickly and the mentioned applications will be used outside the labs eventually.

A BCI must fulfill the following criteria [13]:

- a BCI must use signals recorded directly from the brain
- at least one of these signals must be intentionally modulated by the user to effect goal-directed behavior
- the system must react in real time (low latency)
- the system must provide feedback to the user (closed-loop system)

There are many methods to record brain activity: electroencephalography (EEG), magnetoencephalography (MEG), electrocorticography (ECoG), functional magnetic resonance imaging (fMRI), near-infrared spectroscopy (NIRS) and positron emission tomography (PET). EEG is the most widely used method for a BCI, because of low costs, good temporal resolution and simple application. EEG is a non-invasive method where electrodes are placed on the scalp. With these electrodes, electrical activities of neurons from the cortex are measured.

If a person moves a body part, corresponding areas on the primary motor cortex are activated. The basic idea behind a specific type of BCI system – the so called event-related desynchronization (ERD) BCI or sensory motor rhythm (SMR) BCI – is that these areas are also active when the person only *imagines* a movement. This activity can be measured through event-related synchronization/desynchronization (ERS/ERD) effects in the EEG. If a neural population processes information, the activity becomes desynchronized and the mean power over this area decreases, for further information see [12] and [14]. A ERD-BCI system recognizes different types of imagined movements and transforms these in control commands for a particular device.

In general, an EEG-based BCI system consists of:

- electrode cap

- measurement amplifier
- signal preprocessing
- feature extraction
- classifier
- the device to be controlled

The basic functionality of a BCI system is shown in figure 1.1. The measurement amplifier

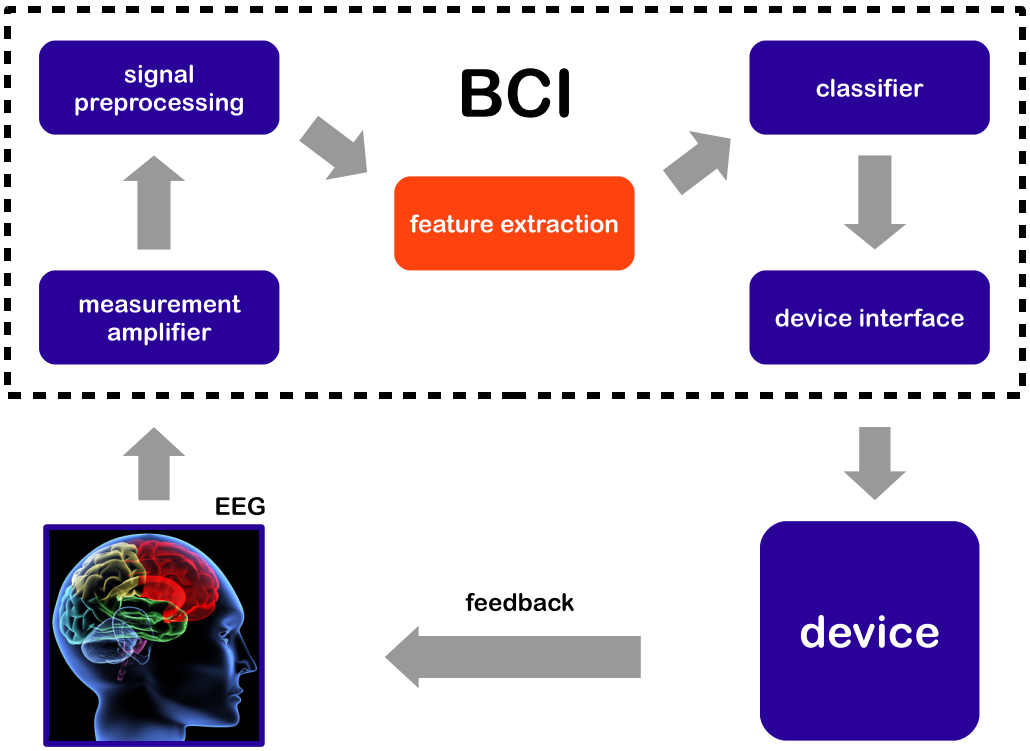


Figure 1.1: basic components of an EEG-based BCI system

records EEG signals (about 10–100 µV) from the electrodes placed on the scalp. These signals are amplified and digitized. The signal preprocessing unit tries to improve the signal-to-noise (SNR) ratio. The SNR is usually low, because there are artifacts resulting from e.g. muscle movements, and volume conduction within the skull blurs the sources. Therefore, spatial filters like bipolar, common average reference (CAR) or Laplace filter are often applied. This master's thesis places emphasis on the feature extraction unit. Its purpose is to extract relevant information from noisy EEG signals for the subsequent classifier. A feature extraction unit applies one or more feature extraction methods to the EEG signal from the signal preprocessing unit. Afterwards, the classifier classifies the extracted features and a control signal (based on the detected class) is sent to the device. The BCI user gets feedback from the device, e.g. observes the steering direction of a wheelchair.

Many feature extraction methods were developed in the past, but no comprehensive comparison of these methods was done yet. In the first part of this master's thesis, such a comparison was performed. In the second part, a combination of feature extraction methods which performs better than the individual methods was searched. A single-trial offline setup with a two-class linear discriminant analysis (LDA) classifier was used. The following feature extraction methods were evaluated and combined, respectively:

- band-power (BP)
- adaptive autoregressive (AAR) parameters
- bilinear adaptive autoregressive (BAAR) parameters
- multivariate adaptive autoregressive (MVAAR) parameters
- phase locking value (PLV)
- Hjorth (HJORTH) parameters
- time domain parameters (TDP)

Each of these feature extraction methods has parameters which need to be set. The concrete values of these parameters have an effect on how well the information from the EEG signal is extracted. Therefore, it is crucial to carefully tune these parameters, otherwise a potentially good method with a bad parameter set could be compared to a bad method with a good parameter set. Such a comparison would be unfair and no meaningful evaluation result could be expected. Therefore, a genetic algorithm was employed to tune the parameter sets of the feature extraction methods to get meaningful evaluation results. To facilitate this comparison approach, the toolbox "TestFEM", which is based on MATLAB (The MathWorks, Inc.[1]) was written. Beside parameter tuning and method evaluation, this toolbox was also used to find the best combination of feature extraction methods.

Motivation and Aim of this Work Generally one want to apply a feature extraction method which yields a high classification accuracy. Also, the optimal parameters of such a feature extraction method are interesting. Furthermore, it is possible that a certain combination of feature extraction methods yields better classification accuracies then the best individual method. A comprehensive method comparison is essential to provide answers to these points, but no such comparison (with parameter optimization) has been found by the author. The aim of this work was to carry out the necessary comparisons. Therefore, the following tasks were performed:

- a comparison of classification accuracies of feature extraction methods (see Chapter 4.2)
- a search for a combination of feature extraction methods with the best classification accuracy (see Chapter 4.3)

[1] `http://www.mathworks.com`

- development of a test toolbox (see Chapter 6)

Chapters 2 and 3 explain basics of the methods used. Chapter 4 and 5 deal with the performed tests. Chapter 6 explains the TestFEM toolbox, this chapter is mostly independent from the other ones.

Chapter 2

Feature Extraction and Combination Methods

A feature extraction method is applied to raw EEG signals after the preprocessing step. These preprocessed signals are typically bandpass and/or spatially filtered. Features are extracted and afterwards classified through a classifier. In this work, the signals were band-pass filtered in the frequency range from $startFrq$ to $stopFrq$ before a feature extraction method was applied. In the case of band-power this band-pass filter was not applied, because the filter is integrated in the method itself. The exact values of $startFrq$ and $stopFrq$ were determined by a genetic algorithm. The mathematical principles of the used feature extraction methods are presented in this chapter. Afterwards, two feature combination methods are shown.

2.1 Adaptive Autoregressive Parameters

Adaptive autoregressive (AAR) parameters are obtained from an adaptive autoregressive model. This adaptive model is based on the autoregressive model (note the missing word adaptive) and some parameter estimation algorithm (here: Kalman filter). An extension to a multivariate AAR (MVAAR) model or a bilinear AAR (BAAR) model can be made. The autoregressive model, the Kalman filter and the application of the Kalman filter on the autoregressive model to estimate the AAR model is described here. Also, the multivariate and bilinear extensions are shown. All explanations are based on [15] and [2].

2.1.1 Autoregressive Model

An autoregressive (AR) model is used for the description or modeling of time series. This is the definition of an AR model of order p:

$$x_k = \sum_{i=1}^{p} a_i x_{k-i} + \varepsilon_k \tag{2.1}$$

p ... model order
x_k ... one-dimensional signal value at time k
a_i ... i-th autoregressive parameter
ε_k ... white noise process

The signal value of x at time k consists of the linear combination of the p previous values of x and a white noise term ε_k. This model can be interpreted as a linear infinite impulse response filter with white noise input. The noise term ε_k describes the signal part which could not be explained by the AR parameters a_i. Thus, ε_k can also be seen as an error term.

The AR parameters are fixed in time and only stationary processes can be described. However, EEG signals are non-stationary, and therefore, an extension to the AR model must be made: the AR parameters are allowed to vary in time. The resulting model is shown below:

$$x_k = \sum_{i=1}^{p} a_{i,k} x_{k-i} + \varepsilon_k \tag{2.2}$$

There are many estimation methods for the parameters $a_{i,k}$ [15]. The most commonly used method is the Kalman filter, which is explained in the next section.

Multivariate Autoregressive Model If the signal x has more than one dimension (channels in the context of EEG signal processing), a multivariate model can be defined:

$$x_{m,k} = \sum_{i=1}^{p} \sum_{j=1}^{M} a_{j,i,m} x_{j,k-i} + \varepsilon_{m,k} \tag{2.3}$$

p ... model order
M ... number of dimensions (channels)
$x_{m,k}$ ... signal value of channel m at time k
$a_{j,i,m}$ ... autoregressive parameters
$\varepsilon_{m,k}$ ... white noise process

vector/matrix notation:

$$\mathbf{x}_k = \sum_{i=1}^{p} \mathbf{A}_i \mathbf{x}_{k-i} + \epsilon_k \tag{2.4}$$

$$\mathbf{x}_k = \left[x_{1,k}, \ldots, x_{m,k}, \ldots, x_{M,k}\right]^T \tag{2.5}$$

$$\mathbf{A}_i = \begin{bmatrix} a_{1,i,1} & \ldots & a_{j,i,1} & \ldots & a_{M,i,1} \\ \vdots & & & & \\ a_{1,i,m} & \ldots & a_{j,i,m} & \ldots & a_{M,i,m} \\ \vdots & & & & \\ a_{1,i,M} & \ldots & a_{j,i,M} & \ldots & a_{M,i,M} \end{bmatrix} \tag{2.6}$$

$$\epsilon_k = \left[\varepsilon_{1,k}, \ldots, \varepsilon_{m,k}, \ldots, \varepsilon_{M,k}\right]^T \tag{2.7}$$

Bilinear Autoregressive Model Linear models are easy to handle, but they cannot express non-linearities. It is likely that EEG data contains non-linearities, therefore the

bilinear autoregressive model has been devised. As in [2], a slightly simplified version of this bilinear model was used:

$$x_k = \sum_{i=1}^{p} a_i x_{k-i} + \varepsilon_k + \sum_{i=1}^{q_1} \sum_{j=1}^{q_2} b_{i,j} x_{k-i} \varepsilon_{k-j} \tag{2.8}$$

p ... autoregressive model order
q_1, q_2 ... model orders of the bilinear term
x_k ... one-dimensional signal value at time k
a_i ... autoregressive parameters
$b_{i,j}$ ... bilinear parameters
ε_k ... white noise process

A combination of the multivariate and bilinear autoregressive model has not been used, because the resulting model would have had too many parameters a, b, and therefore would not have been feasible. Subsequently, as with the standard AR model, the parameters a and b are allowed to vary in time.

2.1.2 Kalman Filter

The Kalman filter [9] provides a means to estimate the internal state of a discrete linear dynamic system. Note that only indirect measurements of this state can be made. This filter is optimal in a way that it minimizes the mean of the squared error. One advantage of the Kalman filter is its recursive operation. It is not necessary to store all past values and allows an easy implementation on a computer system. The explanations provided are obtained from [17] and [2].

State Space Model

In the state space, all measurements, inputs and outputs of the system are combined into a feasible structure. However, in the case of AAR models, there is no input to this structure, so this term is discarded.

The first state space equation is the *system equation* (a linear stochastic difference equation):

$$\mathbf{y}_k = \mathbf{\Phi}_{k-1} \mathbf{y}_{k-1} + \mathbf{w}_{k-1} \tag{2.9}$$

Equation (2.9) describes the system dynamics: the state transition from state $\mathbf{y}_{k-1}$ to the next state $\mathbf{y}_k$. $\mathbf{y}_k$ is a vector and consists of the state parameters at time k. $\mathbf{\Phi}_k$ is the state transition matrix, which relates two successive states. In general, this matrix is time-depended, but here (AAR model) it is assumed to be constant. $\mathbf{w}_{k-1}$ is the process noise.

The second state space equation is the *measurement equation*:

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{y}_k + \mathbf{v}_k \tag{2.10}$$

Equation (2.10) shows the relation between the system state $\mathbf{y}_k$ and the measurement $\mathbf{z}_k$ at time k. $\mathbf{H}_k$ is called measurement matrix and $\mathbf{v}_k$ measurement noise.

The process noise $\mathbf{w}_k$ and the measurement noise $\mathbf{v}_k$ are assumed to be independent of each other. Both are Gaussian distributed with zero mean and covariance matrices $\mathbf{Q}_k$ respectively $\mathbf{R}_k$:

$$\mathbf{w}_k = \mathcal{N}(0, \mathbf{Q}_k) \tag{2.11}$$
$$\mathbf{v}_k = \mathcal{N}(0, \mathbf{R}_k) \tag{2.12}$$

Table 2.1 shows all state space variables, their names and their dimensions. In summary, the Kalman filter takes the measurement $\mathbf{z}_k$ and calculates the system state $\mathbf{y}_k$. $\mathbf{\Phi}_k$, $\mathbf{H}_k$, $\mathbf{Q}_k$ and $\mathbf{R}_k$ must be known, they can be constant or change in time.

symbol	**name**	**dimension**
$\mathbf{y}_k$	state vector	$n \times 1$
$\mathbf{\Phi}_k$	state transition matrix	$n \times n$
$\mathbf{w}_k$	process noise	$n \times 1$
$\mathbf{Q}_k$	covariance matrix of $\mathbf{w_k}$	$n \times n$
$\mathbf{z}_k$	measurement vector	$m \times 1$
$\mathbf{H}_k$	measurement matrix	$m \times n$
$\mathbf{v}_k$	measurement noise	$m \times 1$
$\mathbf{R}_k$	covariance matrix of $\mathbf{v_k}$	$m \times m$

Table 2.1: Kalman filter state space variables

Equations

Let $\hat{\mathbf{y}}_k^-$ be the *a priori* state estimate of $\mathbf{y}_k$ and $\hat{\mathbf{y}}_k^+$ the *a posteriori* state estimate of $\mathbf{y}_k$. The a priori state estimate has only knowledge about the system prior to step k, whereas the a posteriori state estimate has knowledge about the system given the measurement $\mathbf{z}_k$.

A priori and a posteriori estimate errors can be defined as:

$$\mathbf{e}_k^- = \mathbf{y}_k - \hat{\mathbf{y}}_k^- \tag{2.13}$$
$$\mathbf{e}_k^+ = \mathbf{y}_k - \hat{\mathbf{y}}_k^+ \tag{2.14}$$

The a priori estimate error covariance and the a posteriori estimate error covariance are then:

$$\mathbf{P}_k^- = \mathrm{E}\langle \mathbf{e}_k^- \mathbf{e}_k^{-T} \rangle \tag{2.15}$$
$$\mathbf{P}_k^+ = \mathrm{E}\langle \mathbf{e}_k^+ \mathbf{e}_k^{+T} \rangle \tag{2.16}$$

The a posteriori estimate error covariance in equation (2.16) has to be minimized. First, the a posteriori state estimate $\hat{\mathbf{y}}_k^+$ must be expressed through a linear combination of the

a priori state estimation $\hat{\mathbf{y}}_k^-$ and a weighted difference between $\mathbf{z}_k$ (actual measurement) and $\mathbf{H}_k\hat{\mathbf{y}}_k^-$ (predicted measurement):

$$\hat{\mathbf{y}}_k^+ = \hat{\mathbf{y}}_k^- + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{y}}_k^-\right) \tag{2.17}$$

The term $\left(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{y}}_k^-\right)$ in equation (2.17) represents the difference between the actual measurement and the predicted measurement and is called measurement innovation or residual. If the residual is small (estimation is good), $\hat{\mathbf{y}}_k^-$ is already a good estimation of $\mathbf{y}_k$ and will be changed less than when the residual is large (estimation is bad).

$\mathbf{K}_k$ is the Kalman gain factor and is responsible that the a posteriori estimate error covariance in equation (2.16) will be minimized. To obtain the gain factor $\mathbf{K}_k$, equation (2.17) has to be substituted into equation (2.14). The resulting expression is substituted into equation (2.16) and the expectation operator is applied. The trace from the resulting matrix is derived, set to zero and solved for $\mathbf{K}_k$ (which is an $n \times m$ matrix):

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} \tag{2.18}$$

Algorithm

The algorithm can be split in two equation groups: *time update* and *measurement update* equations.

Time update equations:

$$\hat{\mathbf{y}}_k^- = \mathbf{\Phi}_{k-1}\hat{\mathbf{y}}_{k-1}^+ \tag{2.19}$$

$$\mathbf{P}_k^- = \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1}^+\mathbf{\Phi}_{k-1}^T + \mathbf{Q}_{k-1} \tag{2.20}$$

These equations project the state estimation and the estimate error covariance ahead in time (from $k-1$ to k).

Measurement update equations:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} \tag{2.21}$$

$$\hat{\mathbf{y}}_k^+ = \hat{\mathbf{y}}_k^- + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{y}}_k^-\right) \tag{2.22}$$

$$\mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_k^- \tag{2.23}$$

Here, the current Kalman gain is calculated and the current measurement is included. Subsequently, the a posteriori state estimation and the a posteriori estimate error covariance are calculated.

The time update equations are predictor equations, whereas the measurement update equations are corrector equations: a prediction of the current state is made, which is corrected afterwards. At every time step k, the prediction and correction steps are executed, whereas only data from the preceding step and the current measurement need to be known. This gives the Kalman filter its recursive character and makes it feasible even with large amounts of data. The initial values of $\hat{\mathbf{y}}_k^+$, $\mathbf{P}_k^+$, $\mathbf{Q}_k$ and $\mathbf{R}_k$ must be known or chosen appropriately. It must also be defined how $\mathbf{Q}_k$ and $\mathbf{R}_k$ are updated. These things depend on the actual application of the Kalman filter.

2.1.3 Adaptive Autoregressive Model

The AR model was extended in section 2.1.1 with time depended AR Parameters $a_{i,k}$ (see equation (2.2)). These parameters should now be estimated in every time step with the Kalman filter. The AR model is integrated into the Kalman filter as follows:

$$\mathbf{y}_k = \begin{bmatrix} a_{1,k}, \ldots, a_{i,k}, \ldots, a_{p,k} \end{bmatrix}^T \tag{2.24}$$

$$z_k = x_k \tag{2.25}$$

The dimensions are given below (cf. table 2.1):

$$n = p \tag{2.26}$$

$$m = 1 \tag{2.27}$$

Note: z_k, v_k and R_k are scalars because x_k is a one-dimensional signal. The state vector of the Kalman filter represents the AAR parameters. The measurement vector or scalar, respectively, corresponds to the EEG signal.

The real state transition matrix is not known and a good approximation to the current state is simply the preceding state. Thus, the state transition matrix $\mathbf{\Phi}_k$ is set to the identity matrix $\mathbf{I}$. As can be seen from equation (2.9), the current state is then derived from the preceding state.

$$\mathbf{\Phi}_k = \mathbf{I} \tag{2.28}$$

The measurement matrix $\mathbf{H}_k$ must be constructed so that equation (2.10) has the shape of equation (2.2). To accomplish this, $\mathbf{H}_k$ is set to a vector consisting of the p previous EEG signal values:

$$\mathbf{H}_k = \begin{bmatrix} x_{k-1}, x_{k-2}, \ldots, x_{k-p} \end{bmatrix} \tag{2.29}$$

The state space model adapted for AAR models is shown below:

System equation (compare with equation (2.9)):

$$\begin{bmatrix} a_{1,k} \\ a_{2,k} \\ \vdots \\ a_{p,k} \end{bmatrix} = \begin{bmatrix} a_{1,k-1} \\ a_{2,k-1} \\ \vdots \\ a_{p,k-1} \end{bmatrix} + \mathbf{w}_{k-1} \tag{2.30}$$

Measurement equation (compare with equation (2.10)):

$$x_k = \begin{bmatrix} x_{k-1}, x_{k-2}, \ldots, x_{k-p} \end{bmatrix} \cdot \begin{bmatrix} a_{1,k} \\ a_{2,k} \\ \vdots \\ a_{p,k} \end{bmatrix} + v_k \tag{2.31}$$

The initial values of the variables are chosen as follows:

$$\hat{\mathbf{y}}_0^+ = \left[0, \ldots, 0\right]^T \text{ or } a_{i,0} = 0 \tag{2.32}$$

$$\mathbf{P}_0^+ = \mathbf{I} \tag{2.33}$$

$$\mathbf{Q}_0 = \mathbf{I} \cdot \mathrm{UC} \tag{2.34}$$

$$R_0 = 1 \tag{2.35}$$

The newly introduced variable UC is the update coefficient. It controls how fast the AAR model can adapt to changes in non-stationary signals. With a small UC the AAR model can not follow all changes, but a large UC results in a high variance of the estimated parameters. Usually, UC is set to a value between 10^{-6} and 10^{-1}.

It also has to be defined how the covariance matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$ are updated. There are many update methods and only a few of them have been used in this work. Table 2.2 shows the update methods for the process noise covariance matrix $\mathbf{Q}_k$, Table 2.3 for the measurement noise covariance matrix $\mathbf{R}_k$. Here, $\mathbf{R}_k$ is (standard AAR model) a scalar, but for the purpose of generalization, it is considered as a matrix. An overview and a comparison of more update methods can be found in [15]. According to [15], the $\mathbf{Q}_k$ update method with mode 2 and the $\mathbf{R}_k$ update method with modes 0 and 1 lead to good results and are favorable in the set of all compared update methods. In other words, the best update methods are included in the subset used in this work.

Q mode	**update method**
0	$\mathbf{Q}_k = \mathbf{Q}_{k-1}$
1	$\mathbf{Q}_k = \mathrm{diag}\left(\mathbf{P}_k^+\right) \cdot \mathrm{UC}$
2	$\mathbf{Q}_k = \mathbf{I} \cdot \mathrm{UC} \cdot \mathrm{trace}\left(\mathbf{P}_k^+\right)/n$

Table 2.2: update methods for the process noise covariance matrix $\mathbf{Q}_k$

R mode	**update method**
0	$\mathbf{R}_k = \mathbf{R}_{k-1}$
1	$\mathbf{R}_k = (1 - \mathrm{UC})\,\mathbf{R}_{k-1} + \mathrm{UC} \cdot \mathbf{e}\mathbf{e}^T,\ \mathbf{e} = \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{y}}_k^-$

Table 2.3: update methods for the measurement noise covariance matrix $\mathbf{R}_k$

The purpose of this whole framework is to extract AAR parameters $a_{i,k}$ at each time step k from the EEG signal (respectively $a_{j,i,m,k}$ in the case of a MVAAR model and $a_{i,k}$, $b_{j,i,k}$ with a BAAR model). These are the calculated features of this feature extraction method and used for further classification.

Multivariate Autoregressive Model The integration of the multivariate autoregressive model into the Kalman filter is similar to the AAR model and only differences are described here. Now, the parameters $a_{j,i,m,k}$ from equation (2.3) are estimated. Note, the time dependency of the parameters on k is not explicitly shown in equation (2.3). These

are the values of the state and measurement vectors:

$$\mathbf{y}_k = \left[a_{1,1,1,k}, \ldots, a_{M,1,1,k}, a_{1,2,1,k}, \ldots, a_{M,p,1,k}, a_{1,1,2,k}, \ldots, a_{j,i,m,k}, \ldots, a_{M,p,M,k}\right]^T \quad (2.36)$$

$$\mathbf{z}_k = \left[x_{1,k} \ldots x_{m,k} \ldots x_{M,k}\right]^T \quad (2.37)$$

The dimensions are given below (cf. Table 2.1):

$$n = MMp \quad (2.38)$$

$$m = M \quad (2.39)$$

Note that M specifies the number of channels.

$\mathbf{H}_k$ is constructed differently when considering an MVAAR model. $\mathbf{H}_k$ is now the Kronecker tensor product between the identity matrix with dimensions $m \times m$ and a vector of previous EEG signal values:

$$\mathbf{H}_k = \mathbf{I} \otimes \left[x_{1,k-1}, \ldots, x_{M,k-1}, x_{1,k-2}, \ldots, x_{m,k-i}, \ldots, x_{M,k-p}\right] \quad (2.40)$$

$$\mathbf{H}_k = \begin{bmatrix} x_{1,k-1}, \ldots, x_{M,k-1}, x_{1,k-2}, \ldots, x_{M,k-p} & 0 \ldots & \ldots 0 \\ 0 \ldots 0 & x_{1,k-1}, \ldots, x_{M,k-1}, x_{1,k-2}, \ldots, x_{M,k-p} & 0 \ldots 0 \\ \vdots & \vdots & \vdots \end{bmatrix} \quad (2.41)$$

Because $\mathbf{R}_k$ is a matrix in the MVAAR model, $\mathbf{R}_0$ is initialized with the identity matrix:

$$\mathbf{R}_0 = \mathbf{I} \quad (2.42)$$

Bilinear Autoregressive Model Only the differences to the standard AAR model are shown. The parameters $a_{i,k}$ and $b_{j,i,k}$ from equation (2.8) are estimated through the Kalman filter. Note, the time dependency of the parameters on k is not explicitly shown in equation (2.8). This is the state vector:

$$\mathbf{y}_k = \left[a_{1,k}, \ldots, a_{p,k}, b_{1,1,k}, b_{1,2,k}, \ldots, b_{1,q_2,k}, b_{2,1,k}, \ldots, b_{q_1,q_2,k}\right]^T \quad (2.43)$$

The dimensions are given below (cf. Table 2.1):

$$n = p + q_1 q_2 \quad (2.44)$$

$$m = 1 \quad (2.45)$$

To reflect the structure in equation (2.8), $\mathbf{H}_k$ must be changed as follows:

$$\mathbf{H}_k = \left[x_{k-1}, \ldots, x_{k-p}, x_{k-1}\varepsilon_{k-1}, x_{k-1}\varepsilon_{k-2}, \ldots, x_{k-1}\varepsilon_{k-q_2}, x_{k-2}\varepsilon_{k-1}, \ldots, x_{k-q_1}\varepsilon_{k-q_2}\right] \quad (2.46)$$

ε_k is calculated with:

$$\varepsilon_k = z_k - \mathbf{H}_k \hat{\mathbf{y}}_k^- \quad (2.47)$$

2.2 Band-Power

To calculate the band-power BP in the frequency range from a to b of a continuous signal $x(t)$, the following steps have to be executed. First, $x(t)$ must be band-pass filtered with lower cutoff frequency a and upper cutoff frequency b:

$$x_1(t) = \text{bandpass}(x(t), a, b) \tag{2.48}$$

Second, the signal is squared:

$$x_2(t) = x_1^2(t) \tag{2.49}$$

Third, a moving average over a time window with length T is calculated:

$$x_3(t) = \frac{1}{T} \int_{t-T}^{t} x_2(\tau) d\tau \tag{2.50}$$

Fourth, the logarithm of the signal is calculated. See Section 7.4 for further notes on this log-transformation.

$$\text{BP}(t) = \log(x_3(t)) \tag{2.51}$$

2.3 Hjorth Parameters

The Hjorth parameters were introduced by Hjorth [7]. In [16] the parameters were defined as follows:

$$\text{Activity} = \text{var}(x(t)) \tag{2.52}$$

$$\text{Mobility} = \sqrt{\frac{\text{Activity}\left(\frac{dx(t)}{dt}\right)}{\text{Activity}\left(x(t)\right)}} \tag{2.53}$$

$$\text{Complexity} = \frac{\text{Mobility}\left(\frac{dx(t)}{dt}\right)}{\text{Mobility}\left(x(t)\right)} \tag{2.54}$$

Activity is the signal power, Mobility is the mean frequency, and Complexity is the change in frequency.

For EEG classification, these parameters are not extracted from the whole signal, rather within short time segments of length T. Usually, T is about a second. E.g. the Activity of a continuous signal x at time t is calculated as:

$$\text{Activity}(t) = \text{var}(x(\bar{t})), \quad \bar{t} = [t - T, t] \tag{2.55}$$

These definitions use a continuous input signal $x(t)$ and can easily be adapted to discrete signals as well (the derivation is calculated as the difference between two sequent sample values).

2.4 Time Domain Parameters

Time Domain Parameters TDP are a generalization of the Hjorth parameters and described in [16]. p derivations of the original signal are calculated:

$$\mathrm{TDP}_i(t) = \log\left(\mathrm{var}\left(\frac{d^i x(t)}{dt^i}\right)\right), \quad i = 0, \ldots, p \tag{2.56}$$

The logarithm ensures approximately a Gaussian distribution of the p TDP features, see Section 7.4 for a discussion on that. As in Section 2.3, the TDP features are calculated within a short time segment of length T and not from the whole signal.

2.5 Phase Locking Value

The explanations are based on [4] and [6]. ϕ_x and ϕ_y are phases of the signals $x(t)$ and $y(t)$. The phase locking value PLV is a measure how stable the phase difference between ϕ_x and ϕ_y is within a specific time window of length T. In other words, it is a measure for the synchrony between areas on the cortex. The PLV at time t is calculated as shown:

$$\mathrm{PLV}(t) = \left|\left\langle e^{j\Delta\phi(\bar{t})}\right\rangle\right|, \quad \bar{t} = [t - T, t] \tag{2.57}$$

$$\Delta\phi(t) = \phi_x(t) - \phi_y(t) \tag{2.58}$$

The expectation operator is usually applied over the trials, but this is not possible in a single-trial setup. Accordingly, $e^{j\Delta\phi(\bar{t})}$ is averaged over the time window. If the phase difference $\Delta\phi(t)$ is constant across the time window, all complex vectors $e^{j\Delta\phi(\bar{t})}$ point in the same direction and the magnitude of the expectation value is 1. If the phase differences are randomly distributed from 0 to 2π, the expectation value of all complex vectors is 0, and therefore also the magnitude.

First, to calculate the PLV, the instantaneous phases $\phi_x(t)$ and $\phi_y(t)$ have to be computed. $x(t)$ and $y(t)$ are real-valued. The imaginary parts are found with the Hilbert transformation. Together with the real parts, the instantaneous phases can be calculated.

The real part is already known:

$$x_{real}(t) = x(t) \tag{2.59}$$

The imaginary part is computed with the Hilbert transform ($p.v.$ denotes the Cauchy principal value):

$$x_{imag}(t) = \frac{1}{\pi} p.v. \int_{-\infty}^{\infty} \frac{x_{real}(\tau)}{t - \tau} d\tau \tag{2.60}$$

The instantaneous phase is given as:

$$\phi_x(t) = \arctan\left(\frac{x_{imag}(t)}{x_{real}(t)}\right) \tag{2.61}$$

[6] provides a mathematically equivalent, but in practice faster algorithm how to compute $e^{j\Delta\phi(\bar{t})}$ from $x_{real}(t)$ and $x_{imag}(t)$.

Of course, the procedure is analogous for $y(t)$, as well as for discrete signals.

2.6 Feature Combination Methods

If more than one feature extraction method has to be used, one must use a feature combination method. Two methods from [5] are presented: CONCAT and META. In the test setup described in Section 4.3, feature extraction methods were combined solely with META.

CONCAT CONCAT concatenates simply all feature vectors of all feature extraction methods. This concatenated feature vector is afterwards classified with a LDA classifier.

META META uses two levels of classifiers. Each feature vector of a feature extraction method is classified with one separate classifier. The *continuous* output from these classifiers is classified afterwards by a meta LDA classifier. A customized classifier (linear, non-linear) could be used for each feature vector, however only LDA classifiers were used. According to [5] META leads in practice to a better classification accuracy as CONCAT.

Chapter 3

Genetic Algorithm

3.1 The Genetic Algorithm

A genetic algorithm (GA) is a stochastic optimization method which is based on natural evolution. An individual in nature is subjected to natural selection because resources (e.g. food, living space) are limited. Each individual has a genome which contains the genes. Genetic changes which lead to an advantage – with respect to reproduction and surviving – for an individual in its environment are likely to become common in the population. These changes result from mutations and recombinations of genes during sexual reproduction. A GA uses these principles and applies it to optimization problems. See [11] for additional notes on this topic and further references; this chapter is based on this reference.

First, it has to be asked how the genes of an individual (hypothesis) are represented. Mostly, they are a bit string or a vector of numbers. The interpretation of this e.g. bit string depends on the application. Second, a so-called *"fitness function"* evaluates how well a specific individual with its genes is working. This is the analog to the reproduction and surviving advantage of real individuals in nature. It is the goal of the GA to alter the genes of an individual in a way that *minimizes* (or maximizes, that depends on the actual implementation of a GA) the value of the fitness function. For example, imagine that a GA should find out if someone can do outdoor sport depending on the weather (sunny, cloudy, rainy). For that problem, an individual is represented with a bit string of 3 bits. Each bit corresponds to one of the three weather conditions and the bits are combined with the "or" operator: for instance, "001" means it is rainy, "110" means it is sunny or cloudy. The fitness function *interprets* the genetic code and assigns a *fitness score* to an individual. "110" would get a low (good) fitness score (because you can do outdoor sport when it is sunny or cloudy), "001" would get a high fitness score (it would be a bad idea to do outdoor sports when it is raining), and e.g. "100" would get a moderate fitness score (it is right that you can do outdoor sport when the sun shines, but you can also do so when it is cloudy). The genetic code and the fitness function are highly application-dependent, and there could be many other representations and fitness functions to model this problem.

Next, three operations – *crossover, mutation and selection* – have to be defined. These operations are in general *not* application-dependent as long as the individual is represented in some "standard" form, e.g. a bit string. The *crossover operation* combines two parents and creates one child with genes from each of the two parents. The *mutation operation* alters in a random manner the genes of an individual, e.g. bits. The *selection operation* selects probabilistically individuals from the population. Individuals with a low (good)

fitness score are selected with a higher probability than individuals with a higher fitness score.

The sequence of a GA is as follows:

1. *Initialization:* create a start population; all individuals are initialized with random values
2. *Selection:* select probabilistically individuals from the population
3. *Crossover:* combine some selected individuals and create children
4. *Mutate:* mutate some selected individuals
5. children and mutated individuals yield the new population
6. go to step 2.

This sequence is depicted in Figure 3.1. The explanations are somewhat vague (e.g. will the children also be mutated?), because the various GA implementations differ from each other in detail.

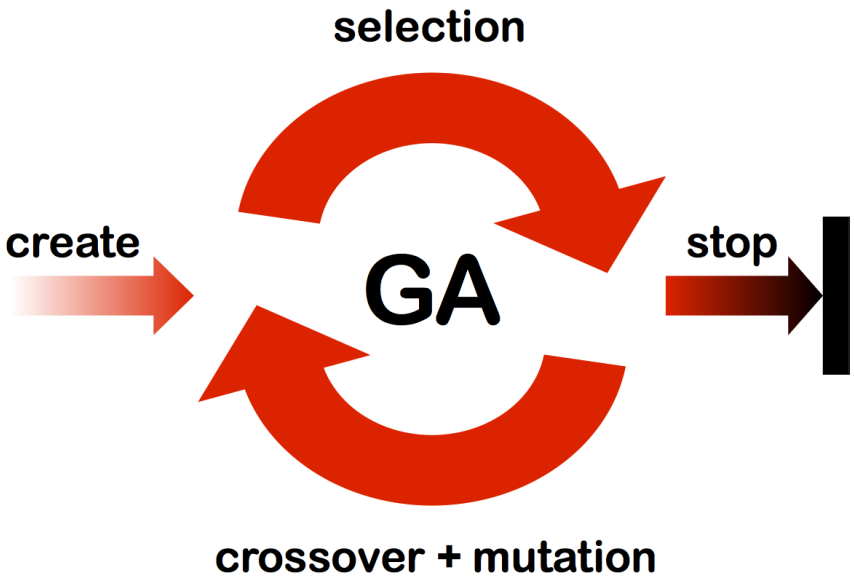


Figure 3.1: GA sequence

Each cycle in the procedure listed above creates a new generation. Generations are created until some stop criterion is reached. A stop criterium could be when:

- the maximum number of generations is reached
- a certain best fitness score is reached

- the best fitness score stalled for some generations

The genetic code of the individual with the best fitness score in the last generation is the solution of the optimization problem. However, this is a *stochastic* optimization method and there is no guarantee that a global optimum has been found.

The idea behind the crossover operation is that from two good parents an even better child could arise (which accelerates the GA), and mutation leads to diversity in the gene pool. If the diversity is too small in the population, all individuals would test nearly the same hypothesis and slow down the progress of the GA. As a result of a slow progress, the GA could get stuck in a local optimum if the GA is stopped too early. As with a *higher* probability the *better* individuals become combined and mutated, the GA optimizes the fitness function. The schema theorem [8] is an approach to characterize the evolution of the population.

Often, a small extension is made to the GA: "elitest selection". In addition to individuals resulting from crossover and mutation operations, also some percentage of the best individuals from a generation are passed unaltered to the next generation. This guarantees that the best fitness score of a generation never increases.

The use of GA is encouraged by some points:

- natural evolution works well, and so should GA
- a GA can optimize hypotheses with complex interacting parts
- no derivation of a error function is needed
- it is straightforward to parallelize GAs

Selection Two prominent selection concepts are presented: *roulette wheel (fitness proportionate) selection* and *tournament selection*.

Roulette wheel selects an individual proportional to its fitness score. The probability that an individual I_i with fitness score $\text{Fitness}(I_i)$ in a population of size p is selected is given by (note: the fitness score is *maximized* here, because the associated formula is more intuitive):

$$P(I_i) = \frac{\text{Fitness}(I_i)}{\sum_{j=1}^{p} \text{Fitness}(I_j)} \tag{3.1}$$

An issue is the range of the fitness score. If this range is too large, individuals with a good fitness score are mostly selected. This narrows the diversity. If this range is too small, all individuals will be selected with almost same probability, slowing down the progress of the GA. Often, a *fitness scaling function* is used to mitigate this problem. This function alters the raw fitness score before the selection function is applied. A fitness scaling function could be used, which gives a fitness score proportionate to the *rank* regarding the raw fitness score of an individual.

Tournament selection chooses randomly two or more individuals from the whole population. From this subset the individual with the best fitness function is finally selected. Tournament selection often results in a population with higher diversity than roulette wheel selection [11].

Crossover The crossover operator depends on the actual representation of the code (e.g. bit string or number vector). In the case of a bit string three crossover operations can be defined:

- *single-point crossover:* the bit string of the child is a concatenation of two substrings from the parents: the first n bits are chosen from the first parent, the remaining bits are chosen from the second parent
- *two-point crossover:* the bit string of the child is a concatenation of three substrings from the parents: the first n bits are chosen from the first parent, the next m bits are chosen from the second parent, the remaining bits are chosen from the first parent
- *uniform crossover:* the individual bits of the child are chosen randomly from the first or second parent

In the case of a real-numbered vector representation, a child could be created as the arithmetic mean value of its parents.

Mutation In the case of bit strings, a point mutation could be realized. A randomly chosen bit in the bit string is inverted with a certain probability. In the case of number vectors with real values, one possibility is to add a value sampled from a Gaussian distribution with mean 0 and standard variation σ. Typically, σ is decreased with every generation.

3.2 Application of the Genetic Algorithm to Feature Extraction Methods

Individual A genetic algorithm has been used to optimize feature extraction methods. Each feature extraction method has a different number and different types of parameters (e.g. integers, real numbers, bit strings). Furthermore, it should be possible to combine various feature extraction methods. A simple bit string or a vector of real values would be insufficient to fulfill these requirements. Therefore, an individual consists here of one or more structures (a list of structures). Each structure represents a feature extraction method and has the entries shown in Table 3.1. The "parameters" entry is again a structure and contains the parameters of a method. These parameters could be of different types and depend on the used feature extraction method. See Table 3.2 for a list of possible parameter types. For implementation details see Chapter 6.

entry	**description**
method	corresponding feature extraction method
channels	list of channels on those this method is applied
parameters	structure which contains the method parameters

Table 3.1: structure entries describing a method

parameter type
integer
nominal number (categorical number)
real number
bit string

Table 3.2: possible types of parameters

Fitness Score The features are extracted from the EEG data set with the methods, the parameters and the channels specified by the individual. The EEG data set used is divided in trials. These trials are divided by a 10×5-fold cross-validation in $10 \cdot 5 = 50$ sets of train trials and test trials. For each set,

- a classifier is trained with the features from the train trials in this set,
- and afterwards, the classes of the samples within the test trials in this set are determined and compared against the true classes.

A classification accuracy is calculated over corresponding samples of all test trials (see Figure 3.2). The fitness score of an individual is 1 minus the 90% quantile of the classification accuracies between cue and end of trial. To improve performance, not every sample or feature vector, respectively, within a trial is used for training and testing the classifier.

If an individual uses more than one feature extraction method, the features are combined with "CONCAT" or "META" (see Section 2.6).

The GA aims to minimize the fitness score of the individuals, and therefore optimizes the parameters of each feature extraction method *and* the combination of methods an individual uses.

Crossover Because an individual consists of a more complicated structure as a bit string, modified crossover and mutation operations must be applied. An individual has two levels of information which need to be crossed over: the used methods itself and their parameters/channels. First, the two parents are searched for methods which exist in only one parent. Each of these methods is copied unmodified to the new child with a chance of 50% (but at least one of these methods is copied). Methods of the same type which exist in both parents are definitely copied to the child, but their parameters and channels are crossed over. Table 3.3 shows the crossover operations for the different parameter types. The channels are selected randomly from channels of the parents.

Mutation This operation mutates the individual on two levels: methods are added/deleted and method parameters/channels are altered. One method is added/deleted from an individual with a probability of 30%. All parameters of all methods of an individual are mutated according to Table 3.4. For the k-th generation out of a total of G generations, scale_k is calculated as:

$$\text{scale}_k = \text{scale}_0 \left(1 - \text{shrink}\frac{k}{G}\right) \tag{3.2}$$

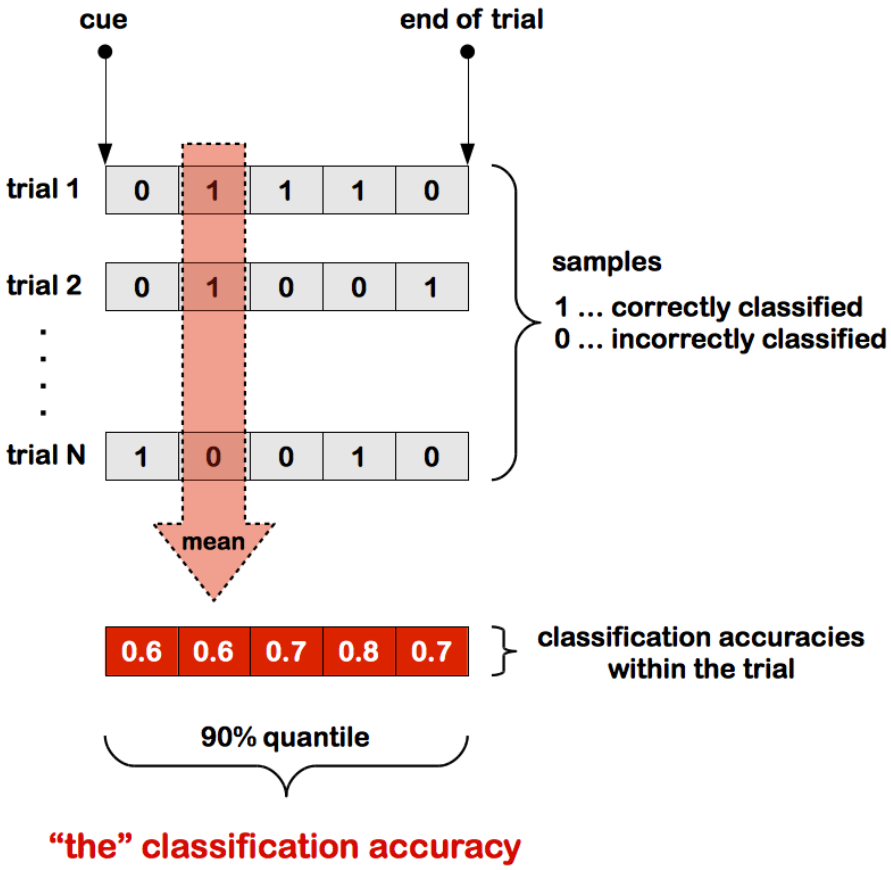


Figure 3.2: calculation of the classification accuracy; the fitness score is 1 minus the classification accuracy

parameter type	crossover operation
integer	rounded mean value
nominal number	value from one parent
real number	mean value
bit string	uniform crossover

Table 3.3: crossover operations for the parameter types

scale_0 and shrink were set as follows:

$$\text{scale}_0 = 0.5 \tag{3.3}$$

$$\text{shrink} = 0.7 \tag{3.4}$$

scale_k decreases linearly from 0.5 to 0.15. σ_k in Table 3.4 is calculated as follows:

$$\sigma_k = \text{scale}_k \frac{\text{upper} - \text{lower}}{2} \tag{3.5}$$

upper and lower are the upper and lower limits of the corresponding parameter. At the beginning, there is a high mutation rate and the GA makes large steps in the search space. At the end, there is a low mutation rate and the GA converges toward a solution. Channels are deleted or added with a probability of 30%.

parameter type	**mutation operation**
integer	add Gaussian distributed noise (mean 0, std. dev. σ_k) and round
nominal number	a new value is chosen with a probability of 30%
real number	add Gaussian distributed noise (mean 0, std. dev. σ_k)
bit string	change each bit with a probability of $scale_k$

Table 3.4: mutation operations for the parameter types

Chapter 4

Test Setup

4.1 EEG Data

All feature extraction methods were applied to prerecorded data (offline analysis) from the BCI competition IV [3]. EEG data from 9 subjects (test persons), all with 2 sessions recorded on different days, were available. A session consisted of 6 runs, each with 48 trials. In each trial, a cue appeared on a screen and the subject had to perform a cue-associated mental task: motor imagery of the left hand (class 1), right hand (class 2), both feet (class 3) or of the tongue (class 4). The cue-based paradigm is illustrated in Figure 4.1. A cross was shown on the screen at the beginning of a trial, and simultaneously a short signal tone was played. After 2 seconds, the cue appeared for 1.25 seconds in the form of an arrow showing left (class 1), right (class 2), down (class 3) or up (class 4). The subject had to imagine the corresponding movement, as long as the cross was shown (until second 6 in a trial). A short break with a blank screen followed. No feedback was provided during the trials. Data from the first session is from now on called **"optimization data"**, and data from the second session **"evaluation data"** .

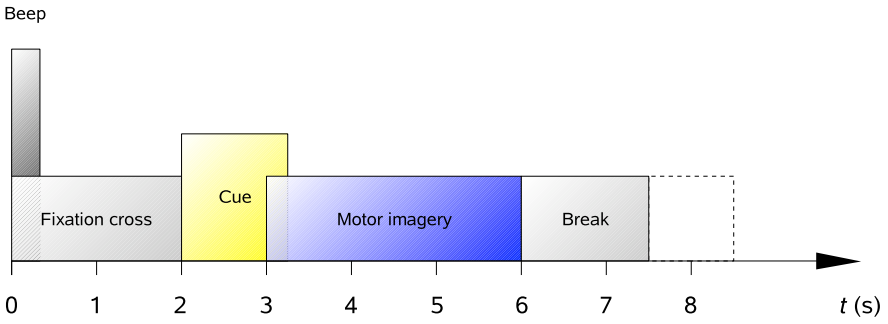


Figure 4.1: cue-based paradigm (this figure was copied with permission from [3])

The EEG was recorded monopolarly (the left mastoid was used as reference, the right as ground) with Ag/AgCl electrodes, which were arranged as shown in Figure 4.2. The signals were sampled at 250 Hz and afterwards band-pass filtered from 0.5 Hz to 100 Hz. In addition, a 50 Hz notch filter was applied. All trials were inspected by an expert and marked if they contained artifacts. [3]

In the test setups, a 2-class classifier was employed, thus only the classes 1 and 2 of the

EEG data were used. For further data reduction, the signals were lowpass filtered with a Butterworth filter of order 5 with a cut-off frequency of 55 Hz and afterwards downsampled to 125 Hz. It was also necessary to calculate Laplace, common average reference (CAR) and bipolar filtered data from the monopolar data. The channel positions of the spatially filtered signals are illustrated in Figure 4.3.

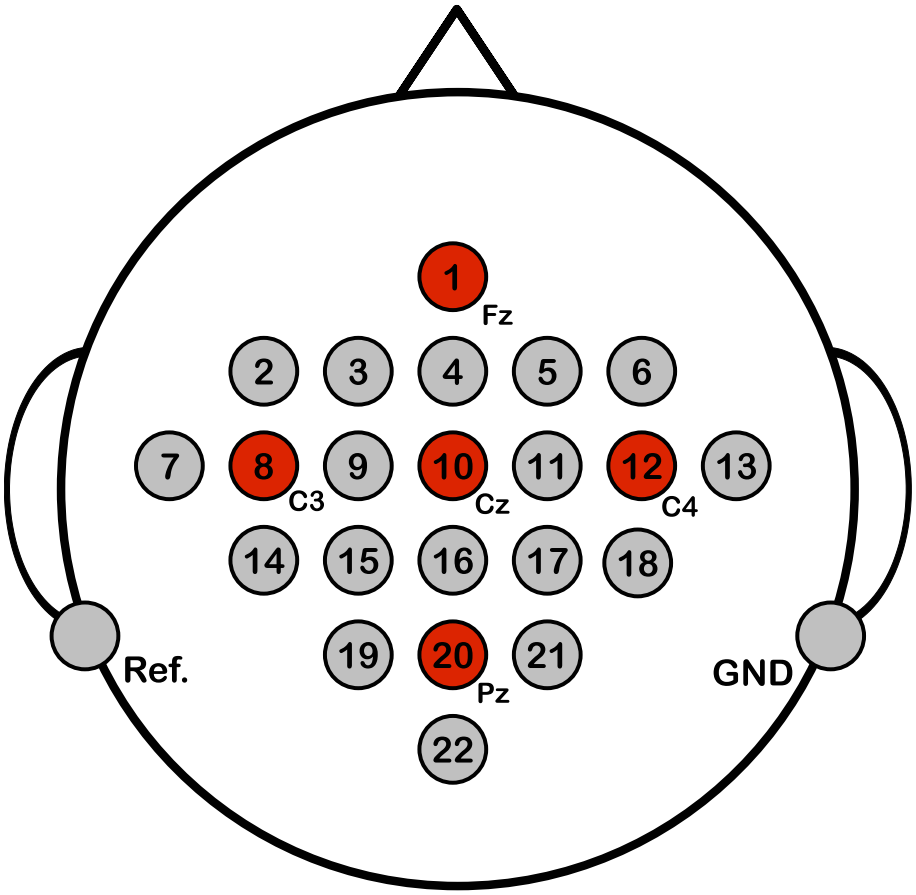


Figure 4.2: The montage of the electrodes. Electrode positions which correspond to electrodes in the international 10–20 system are labeled.

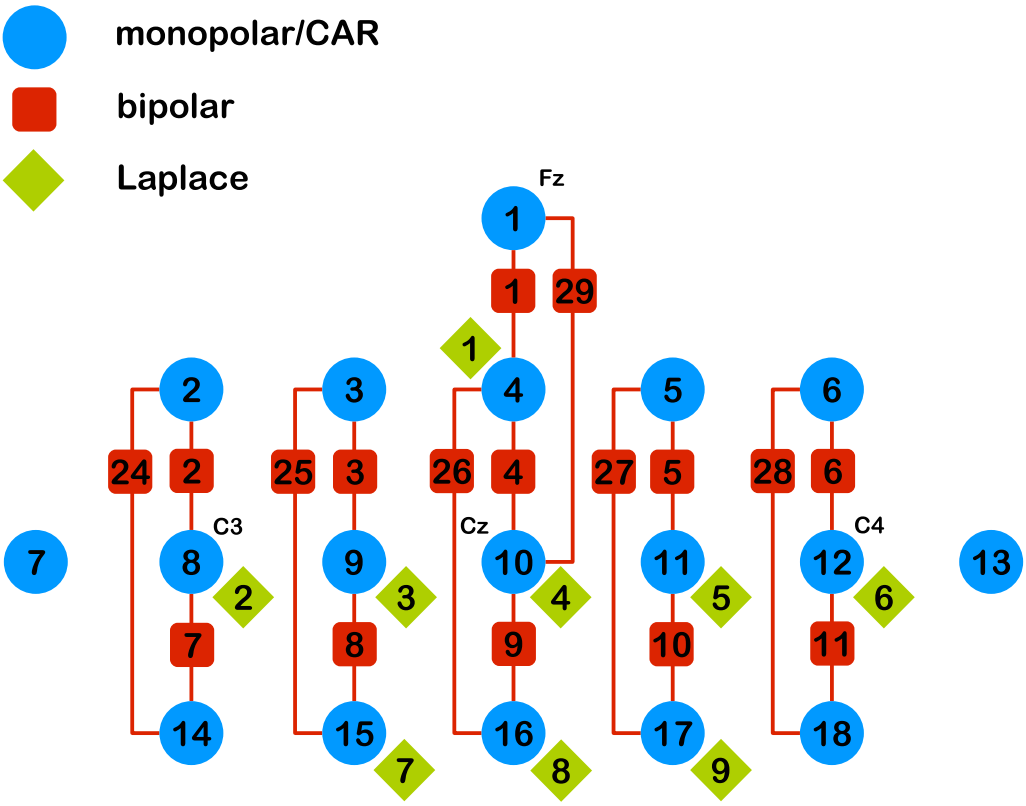


Figure 4.3: channel positions and numbers of the spatially filtered signals (monopolar, bipolar, CAR, Laplace); bipolar channels indicate with a line between which monopolar channels the difference was calculated

4.2 Comparison of Feature Extraction Methods

The feature extraction methods were compared in two different ways:

1. Individual comparison: A certain method was optimized using the optimization data of a subject, and evaluated against the evaluation data of the same subject.

2. General comparison: The optimization data from all 9 subjects were combined into one large optimization data set. A method was optimized using this large data set, and then evaluated separately against the evaluation data of the subjects.

4.2.1 Individual Comparison

The individual comparison approach optimized each method for a subject, and then evaluated the classification accuracy. With this per-subject optimization, the real potentials of the feature extraction methods were found and compared.

All feature extraction methods used data from C3 and C4 electrodes (international 10–20 system) with their corresponding spatial filters. AAR, BAAR, BP, HJORTH and TDP compute features separately from each channel (in contrast to MVAAR). Therefore, computed features (from C3 and C4) of the aforementioned feature extraction methods had to be concatenated.

There is strong evidence that no phase coupling occurs on primary motor cortex areas between the left and right hemisphere. Instead, phase couplings were observed between central and lateral positions on the primary motor cortex, as well as between the primary motor cortex and the premotor area/supplementary motor area [4]. Due to that finding, in addition to 2 channels (C3, C4), also 3 and 4 different channels were used for the PLV feature extraction method (see Tables 4.2 and 4.3). To get PLV information from both hemispheres, it was necessary to use 2 PLV feature extraction methods (one for each hemisphere). The feature values of all PLV feature extraction methods at each time step were simply concatenated into a feature vector.

There is another special case: the BP feature extraction method. As the frequency bands with discriminative information are more or less specific to a hemisphere, two BP feature extraction methods (one for each hemisphere) with their *own frequency band setup* were employed. Again, the computed features from both hemispheres were concatenated.

Thus, AAR, BAAR, HJORTH and TDP concatenated features from both hemispheres. MVAAR computed features considering all input channels, and therefore, it was unnecessary to concatenate features. BP and PLV used separate feature extraction methods for each hemisphere with their own parameter setup.

Depending on the feature extraction method and the spatial filter, several channel sets were tested (see Tables 4.1, 4.2 and 4.3). The optimization step and the evaluation step were executed for all combinations of subjects, methods and spatial filters with their channel sets. These two main steps were implemented in the MATLAB toolbox TestFEM. For details on the toolbox TestFEM see Chapter 6.

Figure 4.4 shows the basic procedure how the classification accuracy for a feature extraction method was determined.

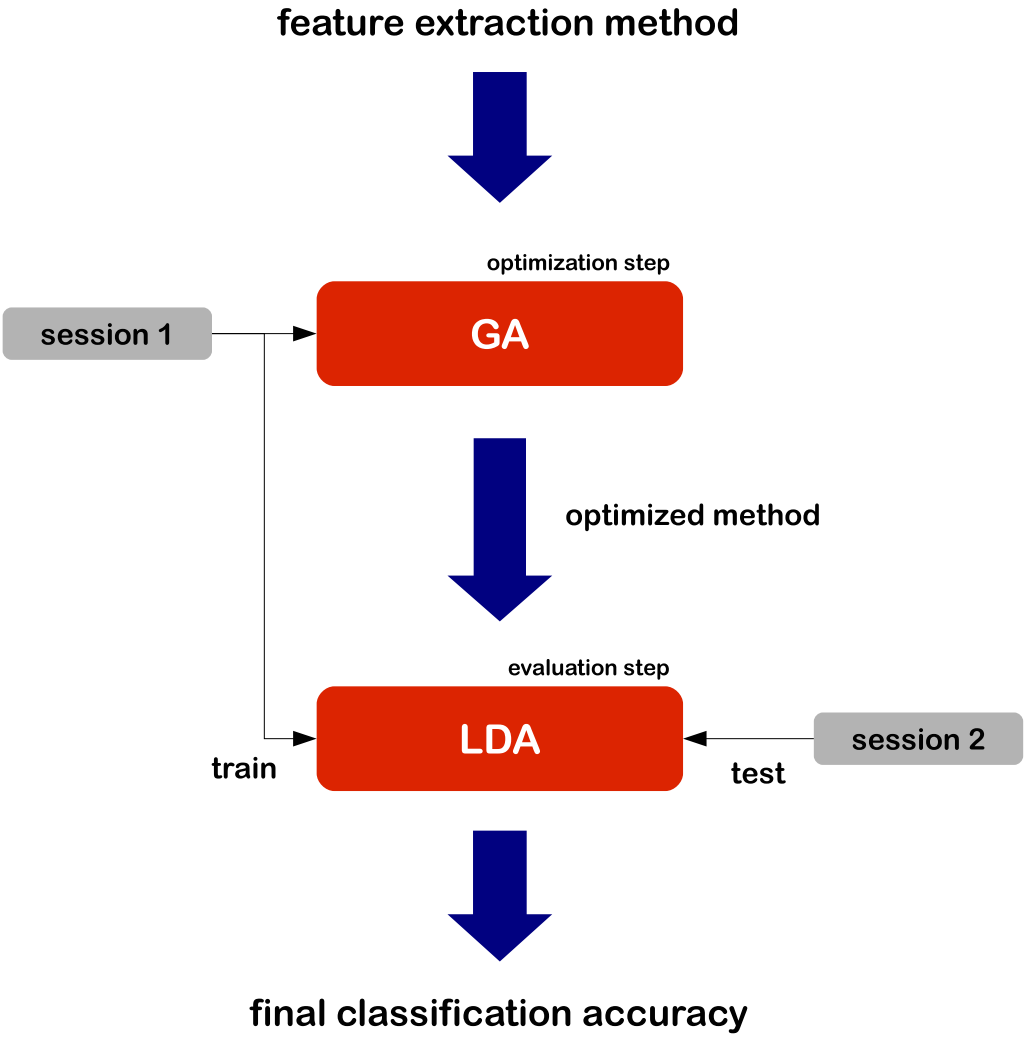


Figure 4.4: procedure for determining the classification accuracy of a feature extraction method

spatial filter	channels		
monopolar	8 & 12		
bipolar	2 & 6	7 & 11	24 & 28
Laplace	2 & 6		
CAR	8 & 12		

Table 4.1: channel sets used for all feature extraction methods; the channel positions are depicted in Figure 4.3

spatial filter	channels			
monopolar	8 & 10, 10 & 12	8 & 4, 4 & 12	8 & 1, 1 & 12	
bipolar	2 & 1, 1 & 6	7 & 4, 4 & 11	7 & 1, 1 & 11	
	2 & 4, 4 & 6	7 & 9, 9 & 11	24 & 29, 29 & 28	24 & 26, 26 & 28
Laplace	2 & 1, 1 & 6		2 & 4, 4 & 6	
CAR	8 & 10, 10 & 12	8 & 4, 4 & 12	8 & 1, 1 & 12	

Table 4.2: channel sets with 3 different channels, used additionally for PLV; the comma separates channels by their hemisphere (each hemisphere has a PLV feature extraction method, see text above)

spatial filter	channels		
monopolar	8 & 3, 5 & 12	8 & 2, 6 & 12	8 & 9, 11 & 12
bipolar	7 & 3, 5 & 11	2 & 8, 6 & 10	2 & 3, 5 & 6
	7 & 8, 10 & 11	24 & 25, 27 & 28	7 & 2, 6 & 11
Laplace	2 & 3, 5 & 6		
CAR	8 & 3, 5 & 12	8 & 2, 6 & 12	8 & 9, 11 & 12

Table 4.3: channel sets with 4 different channels, used additionally for PLV; the comma separates channels by their hemisphere

Optimization Step In this step, the parameters of a feature extraction method were optimized – specifically for a subject, spatial filter and channel set – with a genetic algorithm (see Chapter 3). An individual consisted of one feature extraction method, except in the cases of BP and PLV (see text above). The size of the population was 50, and 300 generations were generated. The fitness score of an individual, which was minimized by the genetic algorithm, was calculated as 1 subtracted by the 90% quantile of the classification accuracy of an LDA classifier over a trial (see Section 3.2, particular Figure 3.2). This classifier was trained with features extracted in an interval of 300 ms between second 3 and 6 within a trial, and tested against features extracted in an interval of 300 ms between second 2 and 6 within a trial. Train and test trials were selected through a 10×5-fold cross validation. Only data from the first session was used (optimization data). To reduce data, pauses between trials were cut out. Trials which contained artifacts were marked by an expert and removed [3].

Evaluation Step The optimized feature extraction method was used to extract features from the optimization and evaluation data, and the whole data were used (including pauses and artifacts). The LDA classifier was trained with features from the optimization data and tested against features from the evaluation data. Train features were extracted in an interval of 40 ms between second 3 and 6 within a trial, and test features were extracted in an interval of 8 ms (every sample) between second 2 and 6 within a trial. Again, the 90% quantile of the classification accuracy over a trial was taken as the final classification accuracy (similar as in Figure 3.2).

To simplify results and their interpretation, only the channel set with the highest resulting classification accuracy for a certain subject/method/spatial filter combination was considered for further analysis. Because this is an optimization, the fitness score of the best individual from the optimization step (and not the final classification accuracy from the evaluation step!) was used to select the best channel set.

It is important to emphasize that only unseen data were used for testing the classifier in the evaluation step.

4.2.2 General Comparison

In this approach, all optimization data from all 9 subjects were concatenated temporally consecutively. This concatenated data set was used in the optimization step and for training in the evaluation step. Hence, all of the optimization data was used at once for the optimization, the genetic algorithm could not have optimized a feature extraction method for a specific subject. Rather, the feature extraction methods were optimized in a general way.

Optimization Step This step was primarily the same as in Section 4.2.1, except that the data from all subjects were combined before.

Evaluation Step The LDA classifier was trained with features extracted from the concatenated optimization data, and tested *separately* against the evaluation data of each subject. All other steps were the same as described in Section 4.2.1.

Again, to simplify results, only the channel set with the highest resulting classification accuracy for a certain subject/method/spatial filter combination was further used. The fitness score of the best individual from the optimization step was used to select the best channel set. Because subjects with the same method/spatial filter/channel set combination used the same optimization step in this setup (and therefore had the same final fitness score), all subjects with the same method and spatial filter selected the same channel set for further analysis.

4.3 Combination of Feature Extraction Methods

In this test setup, combinations of feature extraction methods which yield the highest possible classification accuracies for subjects were searched. Basically it was the same process as in Section 4.2, but now a combination of methods has been optimized and afterwards evaluated.

The methods were not restricted to the C3 and C4 electrodes. All electrodes – except electrodes over the parietal lobe (electrodes 19–22, Figure 4.2) – could be used.

The feature vectors resulting from the feature extraction methods were combined with the META method, explained in Section 2.6.

Optimization Step This optimization step was executed for each subject. Not only the parameters of a method were optimized with the genetic algorithm, but also the combination of methods itself. This was a generalization of the optimization step from

the previous section: an individual consisted of one or more methods instead of exactly one. An individual could have had more methods from one type, e.g. 2 BP methods with different frequency bands for different channels. Because it was not necessary, BP and PLV methods were not handled in some special way as in the previous section. Each method used an own selection of available channels (= all electrodes with all spatial filters, but no electrodes from the parietal lobe). The population size was increased to 100 individuals and 300 generations were evolved. Because of finite computing power, some limitations must have been made: each method could not have used more than 6 channels at once (however, the PLV method used exactly two channels, but more PLV methods could be used by one individual); an individual could have used maximally one BAAR and one MVAAR method, but more methods from other types; an individual could have had maximally 6 feature extraction methods.

Evaluation Step The LDA classifiers (here: META) were trained with features from the optimization data and tested against features from the evaluation data of a subject. This test procedure was already explained in Section 4.2.1.

Chapter 5

Results

5.1 Individual Comparison Results

A feature extraction method was optimized for one person and evaluated against the same person. For the optimization and evaluation step, EEG data sampled at 125 Hz from two different sessions have been used. As final classification accuracy, the 90% quantile of the classification accuracies over the time within a trial was taken (see Figure 3.2). Several channel combinations were tested for each spatial filter, but only the channel combination with the best classification accuracy was used for that person and that spatial filter. The test procedure is explained in Section 4.2.1 in more detail. Section 5.1.1 compares the classification accuracies (descriptive statistics, significance tests), and Section 5.1.2 analyses the parameters of the feature extraction methods found by the genetic algorithm.

5.1.1 Classification Accuracies

Descriptive Statistics

A summary of the classification results of all 9 subjects is presented here. Table 5.1 shows the means, Table 5.2 the standard deviations, and Table 5.3 the medians of the classification accuracies broken down by feature extraction method and spatial filter. Table 5.4 and 5.5 show mean values, standard deviations and medians of feature extraction methods and spatial filters, respectively. Figure 5.1 shows a box-and-whisker plot. The rank of a feature extraction method (regarding the mean of the classification accuracies), the best spatial filter for a method, and the approximate distribution of the evaluation results can be read from this plot. TDP with a bipolar spatial filter gives the best classification results. Figure 5.2 shows the mean values and standard deviations of the classification accuracies from all feature extraction methods and their spatial filters. The effect of the spatial filter on the classification accuracies can be seen clearly in this figure. Except with PLV, bipolar and Laplace spatial filters work better than CAR and monopolar spatial filters. Furthermore, bipolar filters lead almost always to a better mean accuracy than Laplace filters. With PLV, exactly the opposite is the case.

	bipolar	CAR	Laplace	monopolar
AAR	0.74	0.65	0.71	0.61
BAAR	0.68	0.62	0.70	0.60
BP	0.77	0.69	0.74	0.64
HJORTH	0.76	0.69	0.73	0.63
MVAAR	0.74	0.66	0.69	0.62
PLV	0.63	0.67	0.65	0.66
TDP	0.78	0.70	0.74	0.63

Table 5.1: mean values of the classification accuracies

	bipolar	CAR	Laplace	monopolar
AAR	0.12	0.09	0.11	0.09
BAAR	0.15	0.07	0.13	0.11
BP	0.11	0.10	0.13	0.10
HJORTH	0.12	0.10	0.11	0.10
MVAAR	0.13	0.10	0.09	0.09
PLV	0.07	0.09	0.06	0.12
TDP	0.11	0.11	0.11	0.10

Table 5.2: standard deviations of the classification accuracies

	bipolar	CAR	Laplace	monopolar
AAR	0.80	0.68	0.73	0.58
BAAR	0.63	0.64	0.68	0.54
BP	0.77	0.66	0.69	0.60
HJORTH	0.77	0.67	0.71	0.58
MVAAR	0.83	0.60	0.71	0.58
PLV	0.63	0.65	0.61	0.61
TDP	0.82	0.67	0.72	0.59

Table 5.3: medians of the classification accuracies

	mean	sd	median
AAR	0.68	0.11	0.66
BAAR	0.65	0.12	0.60
BP	0.71	0.12	0.67
HJORTH	0.70	0.11	0.69
MVAAR	0.68	0.11	0.63
PLV	0.65	0.08	0.63
TDP	0.71	0.12	0.69

Table 5.4: mean values, standard deviations and medians of the classification accuracies of feature extraction methods

	mean	sd	median
bipolar	0.73	0.12	0.71
CAR	0.67	0.09	0.65
Laplace	0.71	0.11	0.70
monopolar	0.63	0.10	0.58

Table 5.5: mean values, standard deviations and medians of the classification accuracies of spatial filters

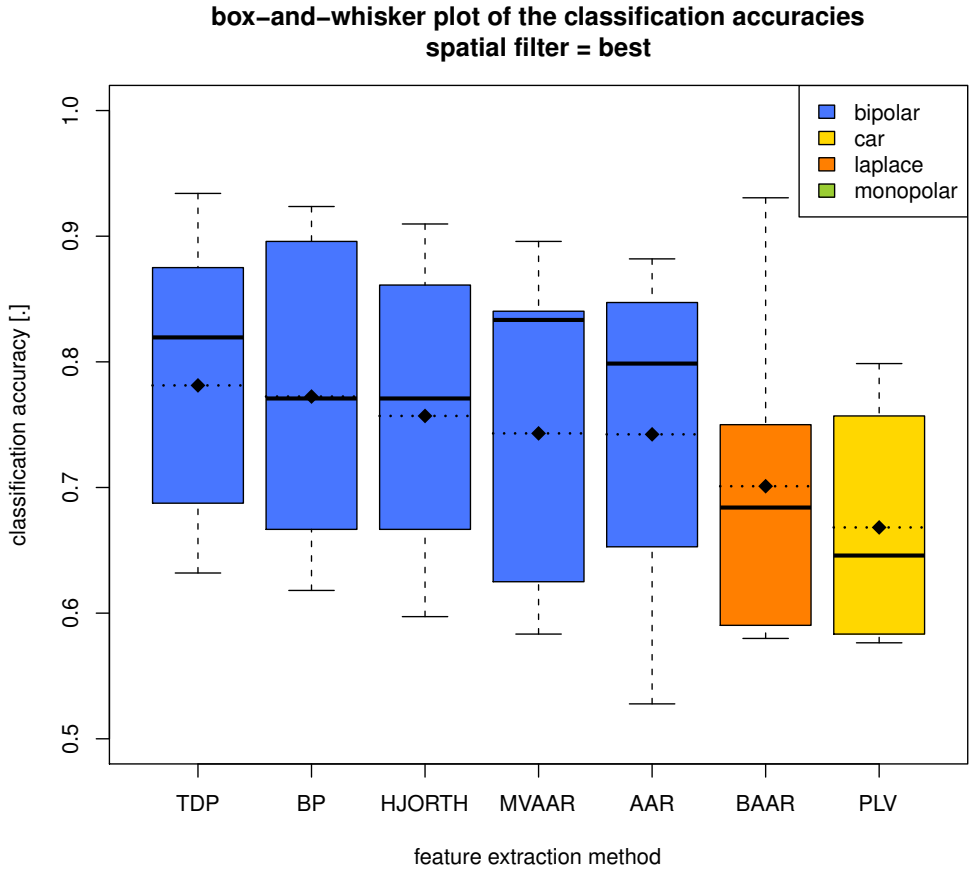


Figure 5.1: Box-and-whisker plot of the mean classification accuracies when the spatial filter with the highest mean accuracy for each feature extraction method was used. The feature extraction methods are sorted by the mean value of their classification accuracies. The thicker black solid line shows medians, the dotted line with the square marks mean classification accuracies of feature extraction methods.

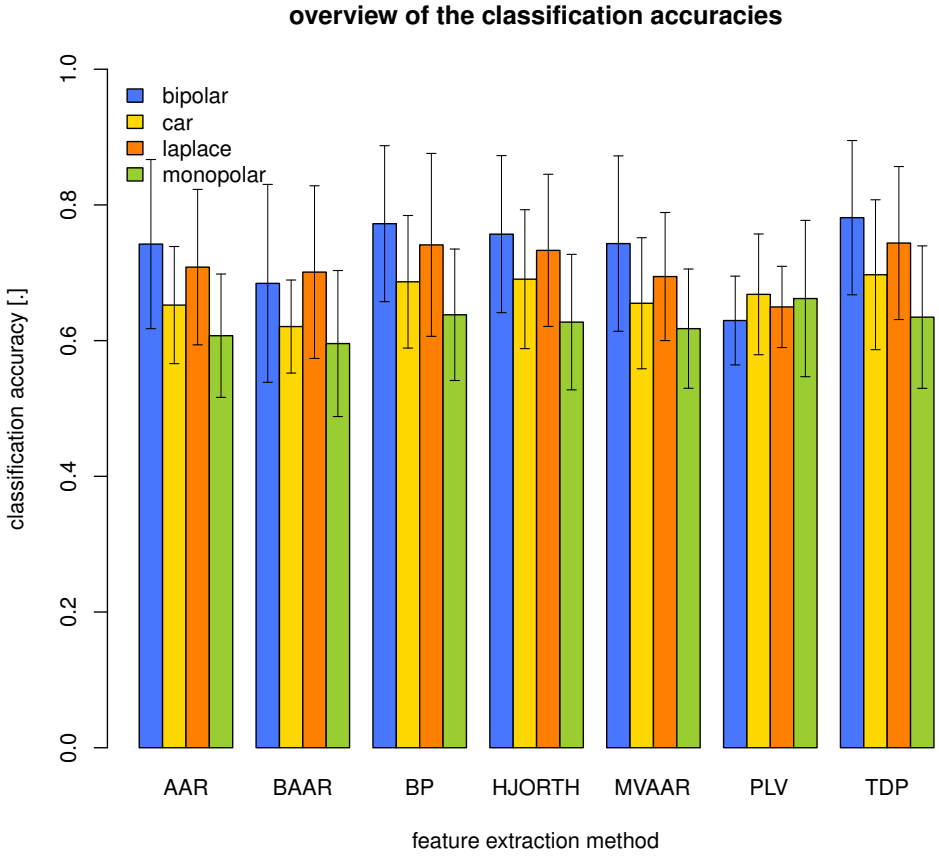


Figure 5.2: overview of mean values and standard deviations of the classification accuracies for all feature extraction methods and all spatial filters

ANOVA

A two-way repeated measures ANOVA was computed over the classification accuracies. The ANOVA tests if a certain choice of a feature extraction method and/or spatial filter has a significant effect on the classification accuracy. *Feature extraction method* (here often abbreviated with *method*) and *spatial filter* are called *factors*. Each factor has a certain number of *levels*. The factor method has e.g. 7 levels (AAR, BAAR, BP, etc.). A certain method/spatial filter combination is called *group*. Because there are 7 levels for the factor method and 4 levels for the factor spatial filter, there are $7 \cdot 4 = 28$ groups. A group contains the classification accuracies of the subjects which were evaluated with the method/spatial filter combination from that group. If a factor leads to significant differences in the mean values of the groups collapsed over all other factors, it is called *main effect*. If a factor influences another factor, it is called *interaction effect*. The ANOVA tests if there are any main effects or interactions; the null hypotheses are: there are no main effects/interactions.

A *repeated measures* ANOVA must be used, because each subject was tested with all combinations of the two (within-subject) factors: *feature extraction method* and *spatial filter*. In other words, a subject was repeatedly tested.

A so called *p-value* is important. This value specifies the probability that the null hypotheses is true. If p has a value smaller than 0.05, the null hypothesis can be rejected.

Normal distribution of the groups and sphericity are preconditions for the repeated measure ANOVA. Normal distribution was tested with the Shapiro-Wilk test. The resulting p-values are shown in Table 5.6 (null hypothesis: data is normally distributed). Sphericity was tested with Mauchly's sphericity test (null hypothesis: sphericity can be assumed), see Table 5.7 for the p-values. The sphericity assumption for the factor method is not valid, because the p-value is smaller than 0.05.

The ANOVA is shown in Table 5.8. Because the sphericity assumption is violated for the factor method, the Greenhouse & Geisser (G-G) or Huynh & Feldt (H-H) corrected p-values must be used for that factor. Both values are smaller than 0.05 and so it does not matter which correction is used. The null hypotheses of the ANOVA must be rejected, and there exist two main effects (method, spatial filter) and one interaction effect between the factors. The ANOVA does not show which group means differ. For that purpose, Tukey's post-hoc tests must be used (see next section).

	bipolar	**CAR**	**Laplace**	**monopolar**
AAR	0.23	0.19	0.52	0.05
BAAR	0.13	0.98	0.17	0.00
BP	0.29	0.35	0.35	0.00
HJORTH	0.54	0.97	0.52	0.01
MVAAR	0.02	0.09	0.11	0.04
PLV	0.73	0.08	0.09	0.12
TDP	0.30	0.93	0.64	0.01

Table 5.6: p-values of the Shapiro-Wilk test (p-values smaller than 0.05 are colored)

factor	p
method	0.02
spatial filter	0.20
method*spatial filter	1.00

Table 5.7: p-values of Mauchly's sphericity test

factor	DoF	F	p	G-G ε	G-G p	H-F ε	H-F p
method	6	8.20	0.00	0.45	0.00	0.70	0.00
error	48						
spatial filter	3	8.63	0.00	0.69	0.00	0.93	0.00
error	24						
method*spatial filter	18	2.58	0.00	0.20	0.06	0.39	0.02
error	144						

Table 5.8: results of ANOVA with sphericity corrected p-values
DoF ... degree of freedom
G-G ... Greenhouse & Geisser
H-H ... Huynh & Feldt

Tukey's Test

Tukey's test shows which groups have different mean values. The null hypothesis of Tukey's test is: the group mean values are equal. Tukey's test compares all groups against each other. Here, this would be $28 \cdot 28 = 784$ comparisons. It is not feasible to show all comparison results. The test results of the methods using their best spatial filter are shown in Table 5.9. These are also the methods which are depicted in Figure 5.1. Only the mean classification accuracy of the PLV (CAR) method differs significantly from the mean classification accuracies of the TDP (bipolar) and BP (bipolar) method. The test results of all 4 spatial filters in combination with the best method from Figure 5.1 (TDP) are shown in Table 5.10. The monopolar spatial filter differs significantly from the bipolar and Laplace spatial filters when the TDP feature extraction method is used.

An assumption of Tukey's test is the normal distribution of the groups. This was tested with the Shapiro-Wilk test, see Table 5.6. All groups in Tables 5.9 and 5.10 do not differ significantly from the normal distribution (except MVAAR (bipolar) and TDP (monopolar)).

5.1.2 Feature Extraction Methods Parameters

The genetic algorithm optimized the parameters of the feature extraction methods and these optimized parameters are analyzed in Tables 5.11–5.17. Mean values and variances are shown. For nominal scaled parameters, only the mode was calculated. The meaning of the parameters can be looked up in Chapter 2; the variable names are the same. "startFrq" and "stopFrq" belong to a band-pass filter applied to the EEG signal beforehand. Only

	AAR	BAAR	BP	HJORTH	MVAAR	PLV	TDP
AAR (bipolar)		1.00	1.00	1.00	1.00	0.42	1.00
BAAR (Laplace)	1.00		0.51	0.92	1.00	1.00	0.25
BP (bipolar)	1.00	0.51		1.00	1.00	0.01	1.00
HJORTH (bipolar)	1.00	0.92	1.00		1.00	0.10	1.00
MVAAR (bipolar)	1.00	1.00	1.00	1.00		0.40	1.00
PLV (CAR)	0.42	1.00	0.01	0.10	0.40		0.00
TDP (bipolar)	1.00	0.25	1.00	1.00	1.00	0.00	

Table 5.9: p-values from Tukey's test of the methods with their best spatial filters (as shown in Figure 5.1)

TDP	bipolar	CAR	Laplace	monopolar
bipolar		0.17	1.00	0.00
CAR	0.17		0.99	0.78
Laplace	1.00	0.99		0.01
monopolar	0.00	0.78	0.01	

Table 5.10: p-values of Tukey's test of all 4 spatial filters in combination with the TDP method

the parameters which belong to the best spatial filter for a feature extraction method are analyzed.

The BP feature extraction method used several frequency bands between 4 Hz and 39 Hz. It was the task of the genetic algorithm to find frequency bands which include information useful for classification. Frequency bands which include more discriminative information had a bigger chance to appear in the parameter set of the final optimized method than frequency bands with no useful classification information. All frequencies used by the optimized BP methods of all subjects were considered, and their probability density function (PDF) was estimated with a kernel density estimation method. The PDFs of the frequencies used by the optimized BP methods are shown in Figures 5.3 and 5.4. Remember, as stated in Section 4.2.1, BP and PLV used one method per hemisphere (with their own parameters). Therefore, there are two PDFs, and also the parameters in the corresponding tables are prefixed with the words left or right. The PDFs in Figures 5.3 and 5.4 show three ranges where discriminative information is probably found: mainly in the α- and γ-bands and to a certain extend in the β-band.

It is interesting that TDP and HJORTH have similar frequency ranges for the preceding band-pass filter (ca 10–20 Hz). Also, AAR, BAAR, MVAAR and PLV have similar frequency ranges (ca 10 – 30 Hz).

The optimal update coefficient UC used by AAR, BAAR and MVAAR depends on the choice of the update mode for the $\mathbf{Q}_k$ matrix (process noise covariance matrix). See Tables 5.14–5.16 for the ranges of UC.

	TDP (bipolar)			
	startFrq [Hz]	**stopFrq [Hz]**	**T [s]**	**p**
mean value	12.22	23.89	0.91	6.33
std deviation	6.80	7.69	0.15	2.92
	channel set			
mode	2 & 6			

Table 5.11: optimal parameters for TDP

	BP (bipolar)	
	left T [s]	**right T [s]**
mean value	0.69	0.79
std deviation	0.13	0.13
	channel set	
mode	2 & 6	

Table 5.12: optimal parameters for BP

	HJORTH (bipolar)		
	startFrq [Hz]	**stopFrq [Hz]**	**T [s]**
mean value	9.33	20.89	0.91
std deviation	7.68	7.74	0.12
	channel set		
mode	2 & 6		

Table 5.13: optimal parameters for HJORTH

	MVAAR (bipolar)				
	startFrq [Hz]	**stopFrq [Hz]**	**p**	**UC** (Q mode = 0)	**UC** (Q mode > 0)
mean	5.11	31.67	4.11	10^{-6} – 10^{-4}	10^{-3}
std dev	3.62	10.17	2.20		
	channel set	**Q mode**	**R m.**		
mode	2 & 6	0	0		

Table 5.14: optimal parameters for MVAAR

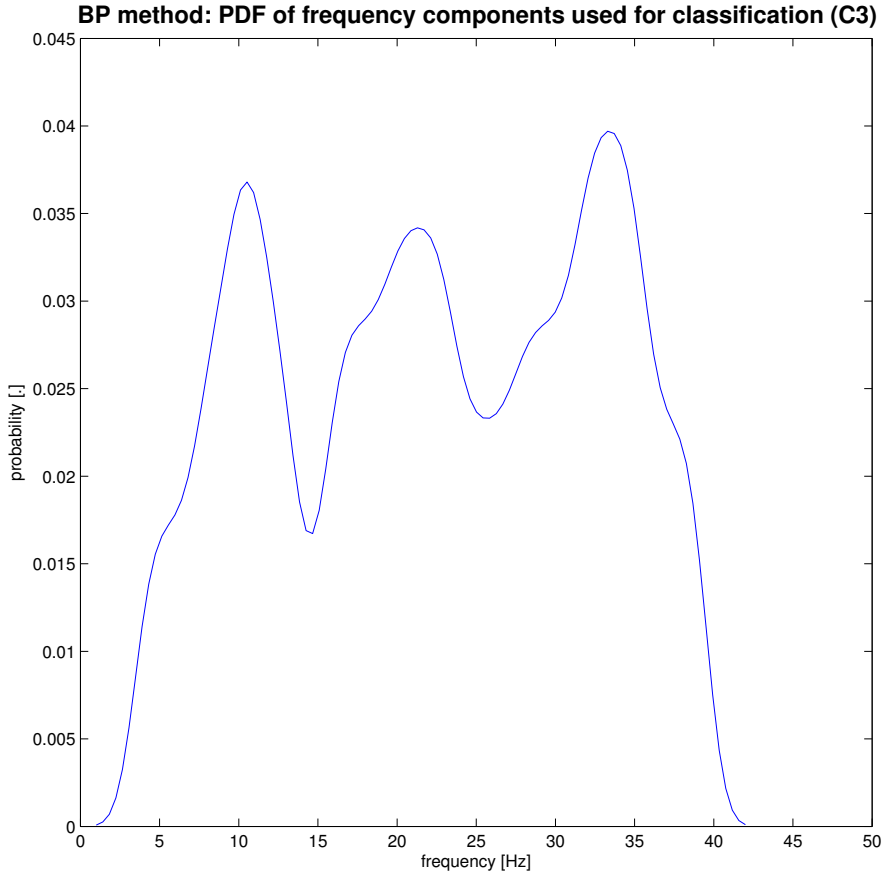


Figure 5.3: BP: estimated PDF of the frequency components used for classification on channel C3 (bipolar); this figure shows how likely it is that a frequency contains discriminative information

	AAR (bipolar)				
	startFrq [Hz]	**stopFrq [Hz]**	**p**	**UC** (Q mode = 0)	**UC** (Q mode > 0)
mean	7.56	33.11	4.22	10^{-6} – 10^{-5}	10^{-3}
std dev	9.59	7.90	1.86		
	channel set	**Q mode**	**R m.**		
mode	2 & 6	0	1		

Table 5.15: optimal parameters for AAR

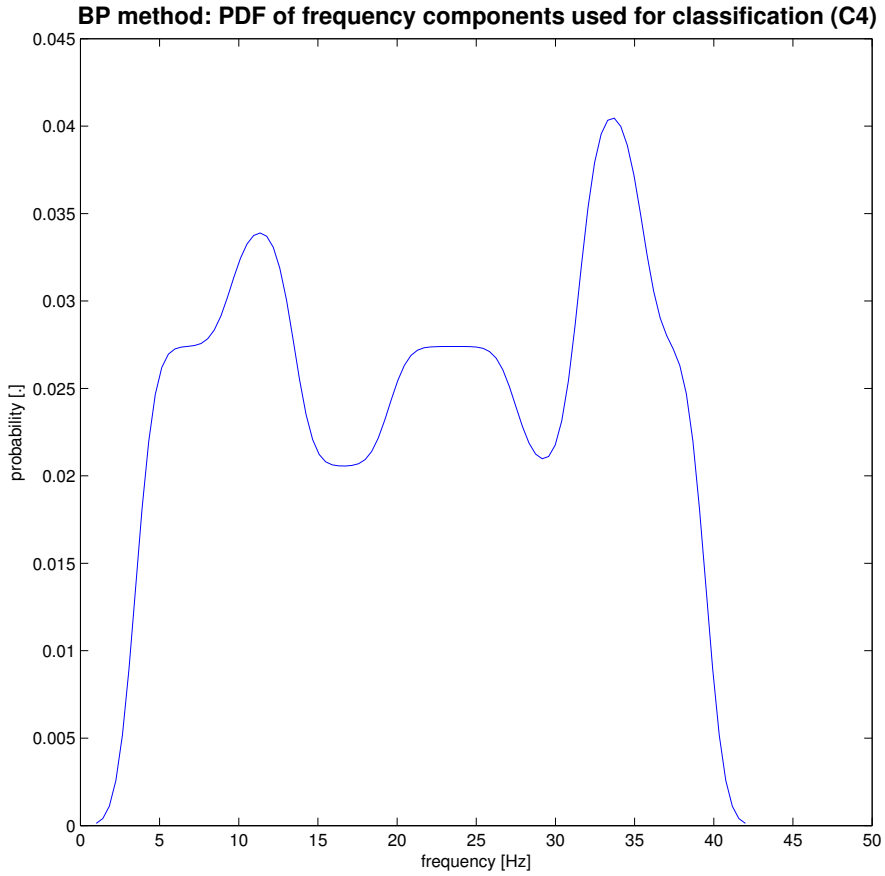


Figure 5.4: BP: estimated PDF of the frequency components used for classification on channel C4 (bipolar); this figure shows how likely it is that a frequency contains discriminative information

	BAAR (Laplace)						
	startFrq [Hz]	**stopFrq [Hz]**	**p**	**q1**	**q2**	**UC** (Q. = 0)	**UC** (Q. > 0)
mean	8.22	34.22	5.67	2.22	4.11	10^{-5} – 10^{-2}	10^{-2}
std dev	4.58	5.78	1.22	0.83	2.85		
	channel set	**Q mode**	**R m.**				
mode	2 & 6	0	0				

Table 5.16: optimal parameters for BAAR

	PLV (CAR)		
	left startFrq [Hz]	**left stopFrq [Hz]**	**left T [s]**
mean value	11.22	29.56	0.76
std deviation	8.06	10.11	0.16
	right startFrq [Hz]	**right stopFrq [Hz]**	**right T [s]**
mean value	11.44	27.56	0.62
std deviation	9.71	9.30	0.32
	channel set		
mode	8 & 2, 6 & 12		

Table 5.17: optimal parameters for PLV

5.2 General Comparison Results

The data from the training session was concatenated and a feature extraction method was optimized with this data, also the LDA classifier in the evaluation step was trained with this data. Because this data consisted of samples from all 9 subjects, the feature extraction method had no chance in the optimization step to adapt to one subject. The optimized feature extraction method was afterwards evaluated individually against the evaluation data from each subject. Here, each subject with the same feature extraction method, spatial filter and channel set was evaluated with the same parameters of the feature extraction method. To simplify things, only the channel set which leads to the best classification accuracy for a certain subject/method/spatial filter combination was used further. See Section 4.2.2 for a more detailed description. Classification accuracies are analyzed in Section 5.2.1, also a comparison is made between individual optimization/training results (Section 5.1) and general optimization/training results in this section. Parameters of the optimized methods are noted in Section 5.2.2.

5.2.1 Classification Accuracies

Descriptive Statistics

Mean values, standard deviations and medians of the classification accuracies of the 9 subjects are shown in Tables 5.18–5.20 (broken down by feature extraction method and spatial filter). Table 5.21 and 5.22 show mean values, standard deviations and medians of feature extraction methods and spatial filters, respectively. A box-and-whisker plot with the best spatial filter for each method is depicted in Figure 5.5. TDP (bipolar) works best. Principally, this figure shows the same situation as 5.1, except that the BAAR method works better with a bipolar spatial filter. Also, the ranking of the AAR, BAAR and MVAAR methods differ among each other. Figure 5.6 shows the mean values and standard deviations of the classification accuracies from all methods and all spatial filters. The effect of the spatial filter is the same as in the preceding section. For all methods – except PLV – bipolar and Laplace spatial filters yield better mean classification accuracies than CAR and monopolar filter.

	bipolar	**CAR**	**Laplace**	**monopolar**
AAR	0.71	0.63	0.67	0.57
BAAR	0.72	0.62	0.66	0.57
BP	0.73	0.65	0.70	0.60
HJORTH	0.73	0.65	0.68	0.60
MVAAR	0.71	0.64	0.67	0.57
PLV	0.61	0.65	0.63	0.65
TDP	0.74	0.67	0.69	0.61

Table 5.18: mean values of the classification accuracies

	bipolar	CAR	Laplace	monopolar
AAR	0.10	0.07	0.08	0.06
BAAR	0.08	0.04	0.09	0.02
BP	0.12	0.05	0.10	0.06
HJORTH	0.11	0.06	0.10	0.08
MVAAR	0.11	0.05	0.12	0.05
PLV	0.07	0.09	0.06	0.08
TDP	0.12	0.07	0.10	0.08

Table 5.19: standard deviations of the classification accuracies

	bipolar	CAR	Laplace	monopolar
AAR	0.68	0.64	0.64	0.57
BAAR	0.67	0.60	0.63	0.56
BP	0.70	0.65	0.67	0.58
HJORTH	0.69	0.64	0.69	0.56
MVAAR	0.69	0.65	0.63	0.56
PLV	0.58	0.61	0.62	0.63
TDP	0.70	0.67	0.65	0.58

Table 5.20: medians of the classification accuracies

	mean	sd	median
AAR	0.64	0.09	0.63
BAAR	0.64	0.08	0.62
BP	0.67	0.10	0.65
HJORTH	0.66	0.10	0.65
MVAAR	0.65	0.10	0.63
PLV	0.64	0.07	0.61
TDP	0.68	0.10	0.65

Table 5.21: mean values, standard deviations and medians of the classification accuracies of feature extraction methods

	mean	sd	median
bipolar	0.71	0.11	0.68
CAR	0.64	0.06	0.64
Laplace	0.67	0.09	0.64
monopolar	0.60	0.07	0.58

Table 5.22: mean values, standard deviations and medians of the classification accuracies of spatial filters

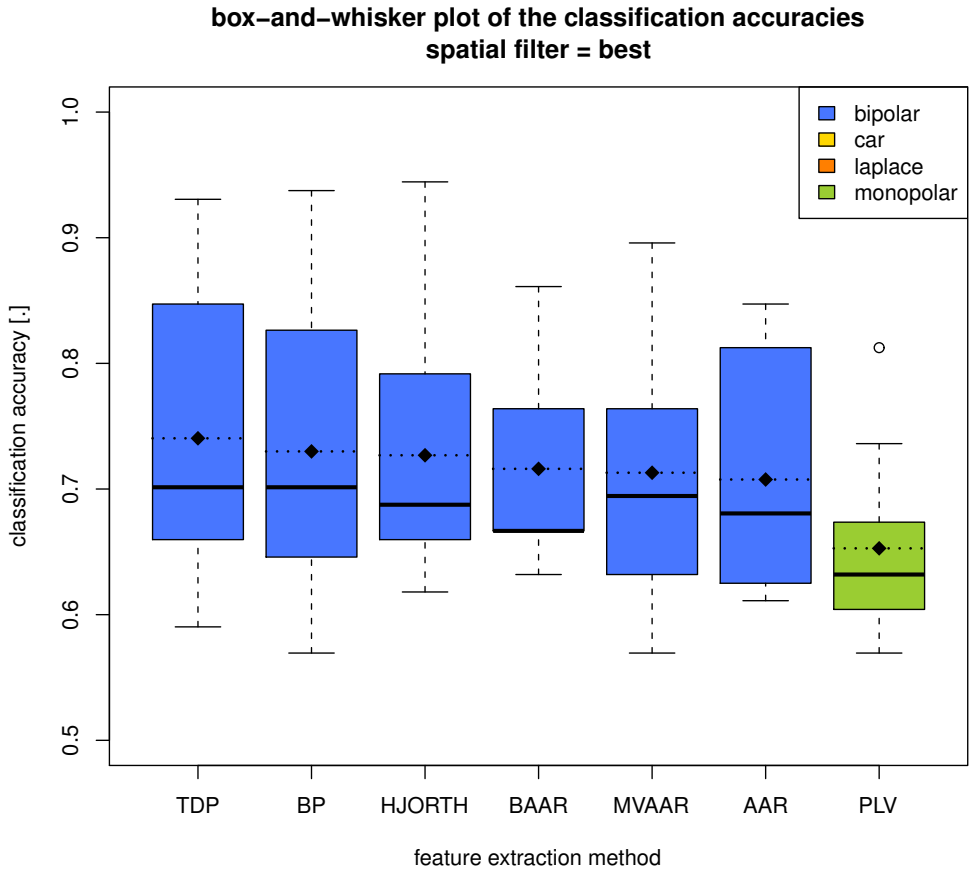


Figure 5.5: Box-and-whisker plot of the classification accuracies when the spatial filter with the highest mean classification accuracy for each feature extraction method was used. Feature extraction methods are sorted by the mean value of their classification accuracies. The thicker black solid line shows medians, the dotted line with the square marks mean classification accuracies of feature extraction methods.

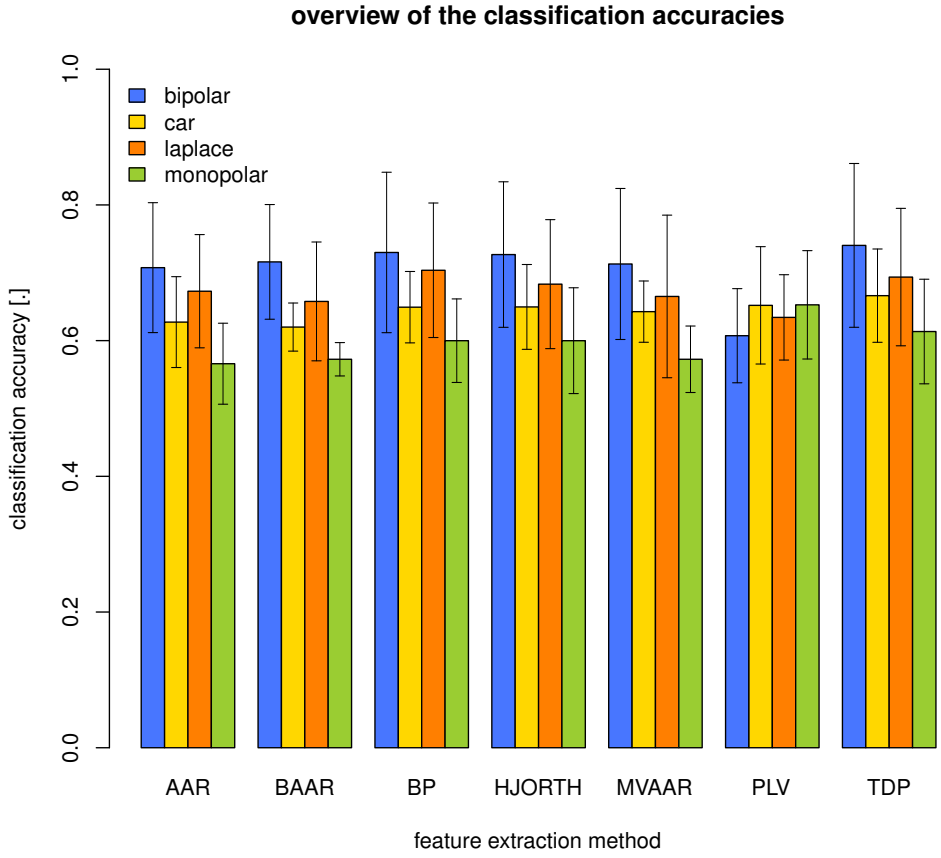


Figure 5.6: overview of mean values and standard deviations of the classification accuracies for all feature extraction methods and all spatial filters

ANOVA

An ANOVA was used to test for significant effects. The factors were: method and spatial filter. Some preconditions must be fulfilled (see Section 5.1.1). The test results for normal distribution are shown in Table 5.23, and the test results for sphericity are shown in Table 5.24. As in the previous chapter, the sphericity assumption for the factor method is not valid and so, the p-value of the factor method must be corrected when applying the ANOVA (with Greenhouse & Geisser or Huynh & Feldt). Table 5.25 shows the ANOVA results. It is not important which correction is used, because both p-values show an significant effect of the factor method. Spatial filter has an main effect and it exists an interaction effect between the factors.

	bipolar	**CAR**	**Laplace**	**monopolar**
AAR	0.07	0.44	0.03	0.27
BAAR	0.13	0.17	0.33	0.01
BP	0.63	0.75	0.42	0.17
HJORTH	0.23	0.56	0.51	0.00
MVAAR	0.73	0.38	0.12	0.00
PLV	0.08	0.07	0.47	0.27
TDP	0.17	0.82	0.15	0.00

Table 5.23: p-values of the Shapiro-Wilk test (p-values smaller than 0.05 are colored)

factor	**p**
method	0.01
spatial filter	0.56
method*spatial filter	1.00

Table 5.24: p-values of the Mauchly's sphericity test

factor	**DoF**	**F**	**p**	**G-G ε**	**G-G p**	**H-F ε**	**H-F p**
method	6	3.54	0.01	0.39	0.04	0.56	0.02
error	48						
spatial filter	3	9.46	0.00	0.73	0.00	1.00	0.00
error	24						
method*spatial filter	18	3.93	0.00	0.20	0.01	0.37	0.00
error	144						

Table 5.25: results of ANOVA inclusive sphericity corrected p-values
DoF . . . degree of freedom
G-G . . . Greenhouse & Geisser
H-H . . . Huynh & Feldt

Tukey's Test

Significant differences in the mean values of the methods shown in Figure 5.5 (the best spatial filter was used for each method) were identified with Tukey's test. Results are shown in Table 5.26. Only TDP (bipolar) differs significantly from PLV (monopolar). Additionally, the spatial filters of the method with the best mean value in Figure 5.5 (TDP) were tested for significant differences. Results are in Table 5.27: when using the TDP method, the monopolar spatial filter differs significantly from the bipolar and Laplace spatial filters.

	AAR	**BAAR**	**BP**	**HJORTH**	**MVAAR**	**PLV**	**TDP**
AAR (bipolar)		1.00	1.00	1.00	1.00	0.70	1.00
BAAR (bipolar)	1.00		1.00	1.00	1.00	0.38	1.00
BP (bipolar)	1.00	1.00		1.00	1.00	0.07	1.00
HJORTH (bipolar)	1.00	1.00	1.00		1.00	0.11	1.00
MVAAR (bipolar)	1.00	1.00	1.00	1.00		0.49	1.00
PLV (monopolar)	0.70	0.38	0.07	0.11	0.49		0.01
TDP (bipolar)	1.00	1.00	1.00	1.00	1.00	0.01	

Table 5.26: p-values from Tukey's test of the methods with their best spatial filters (as shown in Figure 5.1)

TDP	**bipolar**	**CAR**	**Laplace**	**monopolar**
bipolar		0.11	0.92	0.00
CAR	0.11		1.00	0.76
Laplace	0.92	1.00		0.04
monopolar	0.00	0.76	0.04	

Table 5.27: p-values of Tukey's test of all 4 spatial filters in combination with the TDP method

Individual vs. General Optimization/Training

All classification accuracies from Section 5.1 and from this section were compared with a repeated measures ANOVA. In addition to the already used factors "method" and "spatial filter", a factor was added: "comparison". This factor consisted of two levels: one level contains data from Section 5.1, the other level contains data from Section 5.2. The results of Mauchly's sphericity test can be looked up in Table 5.28. The degree of freedom for the factor comparison is 1, so there is no entry for this factor in the table. The main effect of the factor method and interactions with this factor are interesting. No significant effect regarding this factor can be concluded from the test results in Table 5.29.

factor	p
comparison	-
method	0.00
spatial filter	0.49
comparison*method	0.79
comparison*spatial filter	0.07
method*spatial filter	1.00
comparison*method*spatial filter	1.00

Table 5.28: p-values of Mauchly's sphericity test

factor	DoF	F	p	G-G ε	G-G p	H-F ε	H-F p
comparison	1	2.56	0.15	1.00	0.15	1.00	0.15
error	8						
method	6	6.90	0.00	0.39	0.00	0.56	0.00
error	48						
spatial filter	3	10.49	0.00	0.68	0.00	0.92	0.00
error	24						
comparison*method	6	2.20	0.06	0.60	0.10	1.00	0.06
error	48						
comparison*spatial filter	3	0.30	0.83	0.71	0.76	0.98	0.82
error	24						
method*spatial filter	18	3.92	0.00	0.18	0.02	0.31	0.00
error	144						
comp.*method*spatial f.	18	0.77	0.73	0.23	0.56	0.52	0.65
error	144						

Table 5.29: results of ANOVA inclusive sphericity corrected p-values
DoF . . . degree of freedom
G-G . . . Greenhouse & Geisser
H-H . . . Huynh & Feldt

5.2.2 Feature Extraction Methods Parameters

The feature extraction methods were optimized and the resulting parameters of each method are shown in Tables 5.30–5.33. In contrast with the individual comparison (Section 5.1) where a method was optimized separately for each subject, the methods were optimized with the concatenated data from all subjects. As a consequence, for each method/spatial filter/channel set combination exists only one optimized method (instead of 9). Therefore, no mean values etc. can be shown here, but the concrete parameters of the optimized methods. Only the parameters of a feature extraction method which belong to the spatial filter with the best mean classification accuracy are shown.

	channels	**startFrq [Hz]**	**stopFrq [Hz]**	**T [s]**	**p**
TDP	2 & 6	1.00	16.00	0.99	10
HJORTH	24 & 28	1.00	33.00	0.99	-

Table 5.30: optimal parameters for TDP and HJORTH (bipolar)

	channels	**startFrq [Hz]**	**stopFrq [Hz]**	**p**	**q1**	**q2**	**UC**	**Q m.**	**R m.**
AAR	24 & 28	3	13	2	-	-	10^{-6}	0	0
BAAR	24 & 28	1	38	9	1	5	10^{-6}	0	0
MVAAR	2 & 6	1	32	4	-	-	0.003	2	1

Table 5.31: optimal parameters for AAR, BAAR and MVAAR (bipolar)

	channel	**frequency bands [Hz]**	**T [s]**
BP left	2	10-13, 16-27, 32-39	0.71
BP right	6	10-23, 28-31	0.79

Table 5.32: optimal parameters for BP (bipolar)

	channels	**startFrq [Hz]**	**stopFrq [Hz]**	**T [s]**
PLV left	8 & 3	11	32	0.81
PLV right	5 & 12	6	40	0.72

Table 5.33: optimal parameters for PLV (monopolar)

5.3 Combination of Feature Extraction Methods

As described in Section 4.3 the genetic algorithm tried to find a combination of feature extraction methods which gives a ideally high classification accuracy. This optimization was done per subject. The mean value, standard deviation and median over the classification accuracies over all 9 subjects can be found in Table 5.34. Figure 5.7 shows a box-and-whisker plot of the classification accuracies, together with the TDP method already shown in Figure 5.1.

	combination of methods
mean value	0.78
standard deviation	0.13
median	0.81

Table 5.34: mean value, standard deviation and median over the classification accuracies of all 9 subjects

5.3.1 One Method vs. Combination of Methods

The method with the highest mean classification accuracy from Section 5.1 – TDP with a bipolar spatial filter – was compared against the classification accuracies from Section 5.3. For that purpose a paired t-test was applied. This test yielded a non-significant p-value of 0.77. Thus, method combinations do not result in better classification accuracies than the best method from Section 5.1 (TDP, bipolar spatially filtered). See Figure 5.7 for a graphic comparison.

Normal distribution is a precondition for a paired t-test. The classification accuracies resulting from the TDP (bipolar) method were already tested for normal distribution (they are normal distributed). The classification accuracies from Section 5.3 are also normal distributed, because the p-value from the Shapiro-Wilk test is 0.68. The result of the paired t-test should therefore be valid.

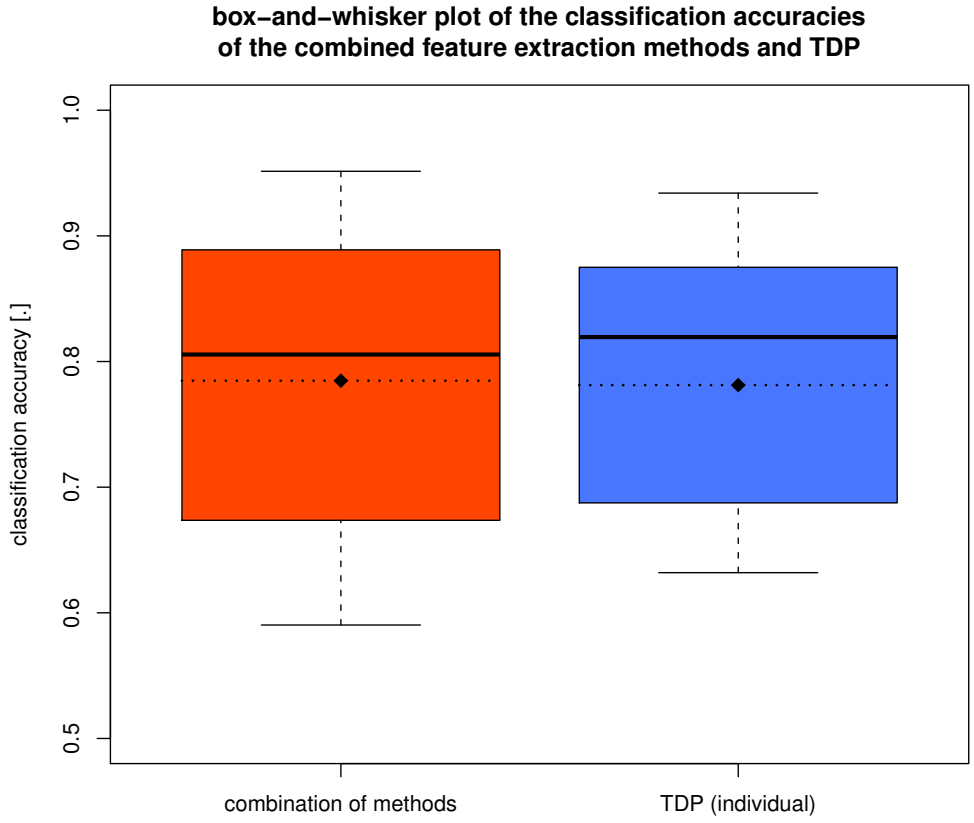


Figure 5.7: Box-and-whisker plot of the classification accuracies of the combined feature extraction methods and TDP (bipolar spatially filtered, from Section 5.1). The thicker black solid line shows medians, the dotted line with the square marks mean values of the classification accuracies.

Chapter 6

TestFEM Toolbox

A MATLAB toolbox called "TestFEM" was developed to facilitate the tests explained in Chapter 4. This toolbox is based on the "BioSig" project[1] which provides a comprehensive toolbox for processing bio-signals, and the "Global Optimization Toolbox" integrated in MATLAB. TestFEM

1. optimizes a feature extraction method or a combination of them,
2. trains a classifier using the optimized methods,
3. and tests a classifier using the optimized methods.

All steps operate on offline data, recorded with a cue-based paradigm. Step 1 is called "optimization step", steps 2 and 3 are called "evaluation step". The genetic algorithm as described in Section 3.2 is used in the optimization step. A certain classifier is trained in the evaluation step and tested afterwards. The test result is a vector of classification accuracies over a trial. The classification accuracies over a trial are determined as in the optimization step (see Figure 3.2, but no 90% quantile is taken).

The next sections show an overview of the data format, the in- and output parameters of a feature extraction method, and how to evaluate a method.

6.1 EEG Data

The EEG data must be stored in MAT-files, GDF files are not directly supported. Such a MAT-file must contains two variables: `signals` and `header`. `signals` is a cell array which contains spatially pre-filtered input signals. Table 6.1 shows which index of `signals` corresponds to which spatial filter. The signals itself are stored as double matrices. The columns correspond to EEG channels, the rows to sample points. `header` is a struct and constructed like a GDF header. `header` must have the following (GDF) entries: `Classlabel`, `TRIG`, `ArtifactSelection` and `SampleRate`. `ArtifactSelection` is a boolean value and indicates if a trial contains artifacts (TRUE: trial contains artifacts).

6.2 Feature Extraction Methods

Feature extraction methods perform the actual work, they process EEG signals and compute features. In the context of this chapter, they are functions which are called from

[1] `http://biosig.sourceforge.net`

index	spatial filter
1	monopolar
2	bipolar
3	Laplace
4	CAR

Table 6.1: indices and spatial filters of the cell array signals

MATLAB. To work together with TestFEM, these functions must be in the search path and have a specific method signature:

```
features = method(signals, fs, min_interval, parameters)
```

Table 6.2 explains the arguments. "compute & hold" means that it is basically not necessary to compute for every sample an appropriate feature vector, instead the last computed feature vector can be reused. Only every `min_interval` samples a new feature vector must be calculated. This should reduce the computing effort if supported by the feature extraction method. `parameters` is a structure and consists of parameters used by the feature extraction method. Only the feature extraction method is responsible for the *interpretation* of these parameters. The next section shows how these parameters are set (fixed parameters) and created/modified by the GA (unfixed parameters).

Each column of `signals` corresponds to an EEG channel. A function must compute features using *all channels* of the input signal. The order and the number of channels will change as the GA progresses. The dimension of the feature vectors is solely determined by this function.

argument	type	dimension	description
`signals`	double matrix	samples × channels	EEG signals
`fs`	integer	scalar	sampling frequency [Hz]
`min_interval`	integer	scalar	„compute & hold“
`parameters`	structure	scalar	parameters used by the method
`features`	double matrix	samples × features	computed features

Table 6.2: arguments of a feature extraction method

6.3 Testing a Method

Testing a feature extraction method means that the classification accuracy in combination with some classifier is determined. The feature extraction methods and the test procedure must be configured before.

Configuring the Feature Extraction Methods A feature extraction method is configured through a prototype. This prototype contains a name, the actual feature extraction

method, fixed parameters and parameters which should be determined by the genetic algorithm (unfixed parameters). A prototype is a MATLAB structure and contains the entries shown in Table 6.3. `methodHandle` is a MATLAB function handle and points to a feature extraction method. This feature extraction method must be created as described in Section 6.2. `parameters` is a structure and will be referred to a feature extraction method. Method parameters which should be fixed (not optimized) must be stored in this structure. `name` is a unique name and identifies a prototype. More prototypes with the same `methodHandle`, but e.g. a different `parameters` structure could be defined. However, their names must be different. `name` is also used for crossover (see Section 3.2): methods can only be combined if their `name` fields match. It must be noted that one entry is not shown in Table 6.3, because this entry is created at runtime by the GA: `channels`. This is a vector containing the used channel numbers.

At runtime in the optimization step, an individual consists of one or more instances of prototypes. More precisely, these instances (prototype structure + unfixed parameters + channels) are stored within a cell array of the individual. When the fitness function calculates the fitness score, for each instance the associated feature extraction method (`methodHandle`) is invoked with the parameters specified in `parameters`. In addition to the fixed parameters specified in the prototype, the GA also creates parameters in the structure `parameters` and modifies them at runtime. For a feature extraction method, there is really no principal difference between fixed and unfixed parameters, both are stored in the same structure at runtime.

How are parameters specified which should be optimized by the GA? This is the task of the `GAConfig` structure. The content of this structure is shown in Table 6.4. `fixedChannels` is a vector of channel numbers which all instances of a prototype must use. Normally, the GA alters the channels for each instance in crossover and mutation operations, but this is not the case when `fixedChannels` is used. `fixedNumberOfChannels` specifies the *exact* number of channels an instance must use, but the channels itself can change. This is useful for e.g. the PLV method, which can only be applied to two channels in a meaningful manner. `maxNumberOfChannels` specifies the *maximum* number of channels an instance can use. If `singleInstance` is set to true (default: false), an individual can not have more than one instance of the corresponding prototype. The latter two options are useful for computationally expensive methods like MVAAR.

`GAConfig.parameters` is a structure, and each parameter which shall be optimized is stored again as a structure in it (the parameter name is used as the field name), e.g. `GAConfig.parameters.exampleParameter`. `exampleParameter` is a structure and simultaneously the name of a parameter which should be optimized. This structure contains information how the GA should initiate, crossover and mutate this parameter; the fields are: `type`, `range` and optionally `loginit`. `type` is a string and one of: `int`, `double`, `selector` (= nominal number) and `bitstring`. The values of `type` correspond to the types shown in Table 3.2. The value range of a parameter is specified with `range`: a two element vector containing upper and lower bounds for "int", "double" and "selector"; an integer for "bitstring", setting the string length.

"int", "double" and "selector" parameter values are initialized with values from a uniform distribution over their range; each bit of a "bitstring" is switched on with a 50% probability. The range of a number parameter could extend over some magnitudes, e.g. UC for AAR was optimized within the range 10^{-6} to 1. The problem is that samples taken

from a uniform distribution over such a wide range are almost near the upper limit. Instead, the exponential of the initial value could be taken from a uniform distribution. `loginit = true` does exactly that (default: false), and mutations are also applied to the exponent of a value.

The parameters specified in the `GAConfig` structure are created by the toolbox in the `parameters` structure (in addition to the fixed parameters) and initialized accordingly. These parameters will be optimized by the GA.

entry	**type**	**description**
`name`	string	name of this method
`methodHandle`	function handle	method which is invoked
`parameters`	struct	parameters for the method
`GAConfig`	struct	configures how the GA processes parameters

Table 6.3: structure entries of a method prototype

entry	**type**	**dimension**
parameters	struct	scalar
fixedChannels	integer vector	channels
fixedNumberOfChannels	integer	scalar
maxNumberOfChannels	integer	scalar
singleInstance	bool	scalar

Table 6.4: structure entries of GAConfig

Configuring the Test Procedure The test procedure is set up with a structure, its entries are shown in Table 6.5. `gaData` sets the name of the data file containing the data used in the optimization step and for training the classifier in the evaluation step; `evaluationData` sets the name of the data file used for testing the classifier in the evaluation step. `evaluationData` could also be a cell array of strings. Each string indicates a data file against which the classifier is tested separately. These data files must be in the format specified in Section 6.1. All loaded EEG data is organized in a matrix with dimensions samples $\times$ channels.

`spatialFilters` and `loadChannels` specify which data should be loaded from the data files. Valid values for `spatialFilters` are: `monopolar`, `bipolar`, `laplace` and `car`. It is possible to load data from more than one spatial filter. In this case, `spatialFilters` must be a cell array of strings. The data from the different spatial filters is concatenated column wise with the same order as in `spatialFilters`. `loadChannels` specifies which channels should be loaded. If this entry is empty (`[]`) or not created, all channels from all given spatial filters are loaded. If only some specific channels should be loaded and `spatialFilters` is a string, `loadChannels` must be a vector containing the desired channel numbers for that spatial filter. If specific channels should be loaded and `spatialFilters` is a cell array

of strings, `loadChannels` must be a cell array of vectors. This cell array must have exactly as many entries as `spatialFilters`. Each vector specifies the channels of that spatial filter which should be loaded. For example, if channels 1 and 2 of the monopolar data and channels 10 and 20 of the bipolar data should be loaded, `spatialFilters` would contain `{'monopolar', 'bipolar'}` and `loadChannels` would be set to `{[1 2], [10 20]}`. As noted above, the loaded EEG data is stored in a matrix with dimension sample × channels; channels would be 4 in this case. It is important to understand that the code in the optimization and evaluation step only sees the loaded EEG data, which is usually a subset of the original data in the data file. Therefore the channel numbering changes. To stay with the example, if a method uses channels 3 and 4 (within the code in the toolbox), the channels really used would be 10 and 20 bipolar spatially filtered. Further note that the dimension variable channels mentioned before is not the same variable as in Table 6.2. The variable channels in Table 6.2 refers to the number of channels actually used by a method, whereas channels used here refers to the number of channels of the loaded EEG data. The structure field `channels` of a method instance selects which channels of the loaded EEG data a feature extraction method should use.

`methods` defines the feature extraction methods available in the optimization and evaluation step. `methods` could be set directly to a prototype (= structure) of a method, in this case all individuals use the same method and only the parameters of a method will be optimized. `methods` could also be set to a cell array consisting of prototypes. In this case, individuals use different combinations of methods and in addition to the parameters, also the combination will be optimized. Basically, an individual uses not all available methods; which methods an individual use is rather determined by the GA. If a prototype defines `GAConfig.fixedChannels`, an individual will always use an instance of this prototype. `maxNumberOfMethods` sets a limit how much instances an individual could use. Thus, it is possible that `methods` defines many methods, but only a certain amount of them can be used at the same time by an individual. This is useful for reducing the computation effort and simultaneously ensure that the best methods will be combined.

`gaTrainInterval` selects the samples within a trial used for training the classifier in the optimization step. `gaTrainInterval` is a 3 element vector: start time, interval, end time. E.g. if set to `[3 0.1 6]`, only feature vectors extracted 3 seconds after trial start until second 6 with a 0.1 second spacing between them will be used for training. `gaTestInterval`, `evaluationTrainInterval` and `evaluationTestInterval` are now self-explanatory. Start and end times depend on the actual paradigm used, and often it is not desired to use every extracted feature vector for training and testing a classifier.

`trimGaData` can be used to reduce the data used in the optimization step. If this parameter is specified, TestFEM cuts out a certain range of all trials and concatenates these fragments. At least all breaks between trials are removed. For example, `[0 6]` concatenates trials ranging from second 0 to second 6 of the original trial. This reduces the computation effort, because less feature vectors have to be computed. However, one must take care of the settling time of the used feature extraction methods. If `trimGaData` is specified, the optimization step uses the reduced data, but not the evaluation step.

`featureCombination` can be set to `'concat'` (default) or `'meta'`, see Section 2.6 for an explanation of these methods.

`classifier` specifies the used classifier. All classifiers supported by the BioSig toolbox

(see functions train_sc and test_sc) which do not require any additional parameters are supported, e.g. `'LDA'`.

`classes` is a two element array consisting of the class numbers used. TestFEM supports only two classes and only trials belonging to one of these classes are considered in the optimization and evaluation step. With some moderate code changes, TestFEM should be extensible to more classes.

`gaPopulationSize` and `gaGenerations` specify the size of the population and how much generations will be created.

entry	**type**	**description**
`name`	string	name of the test
`gaData`	string	GA and training data
`evaluationData`	string/cell array of strings	testing data
`spatialFilters`	string/cell array of strings	used spatial filters
`loadChannels`	vector/cell array of vectors	used channels
`methods`	struct or cell array of structs	method prototypes
`maxNumberOfMethods`	integer	maximum number of methods
`gaTrainInterval`	double array	trial interval used for training
`gaTestInterval`	double array	trial interval used for testing
`evaluationTrainInterval`	double array	trial interval used for training
`evaluationTestInterval`	double array	trial interval used for testing
`trimGaData`	integer array	trim GA data
`featureCombination`	string	feature combination method
`classifier`	string	used classifier
`classes`	array	used classes
`gaPopulationSize`	integer	population size
`gaGenerations`	integer	number of generations to create

Table 6.5: structure entries for setting up the test procedure

Start the Test After prototypes and the test procedure are set up, the actual test is started with:

```
test_result = doTest(test, data_dir, cache_dir, plot_progress)
```

`test` is the structure which sets up the test procedure; `data_dir` is the directory containing the files specified in `test.gaData` and `test.evaluationData`; `cache_dir` is the directory containing the cache files; `plot_progress` is a boolean value which configures if a plot containing the GA progress should be shown (a progress on the console is always shown).

A few notes about caching: TestFEM saves the current GA state into a cache file every tenth generation. If the optimization step is aborted and started again, it will resume from the last saved generation. The `test` procedure must not be altered in the meantime – with the exception of `test.gaGenerations`, because this way the GA can resume from an already finished optimization step if `test.gaGenerations` is increased afterwards.

`test_result` is a structure containing the original `test` structure and structure `result`. `test_result.result` contains the entries shown in Table 6.6.

`methods` is the individual with the lowest fitness score in the last generation of the GA and therefore represents the found optimum. Each cell entry of `methods` is an optimized instance of a method prototype. `finalScore` is the fitness score reached by this individual. `scores` contains the fitness scores of all gaPopulationSize individuals in all gaGenerations generations. The progress of the GA – e.g. mean and minimum over the scores of each generation – can be checked with these fitness scores.

The actual result is composed of the classification accuracies over a trial (like in Figure 3.2). `test.evaluationTestInterval` determines which classification accuracies are assessed. These are stored in a double vector. However, `data.evaluationData` could be a cell vector, specifying more than one evaluation file, and so for each evaluation file a separate double vector with the classification accuracies would be determined. Therefore, `accuracy` is always – even if `data.evaluationData` is a string – a cell vector containing the separate double vectors with the classification accuracies. If `data.evaluationData` is a string, then `accuracy` stores only one double vector; if `data.evaluationData` is a cell vector with evaluationFiles entries, then `accuracy` has also evaluationFiles entries (`accuracy{i}` contains the classification accuracies for `data.evaluationData{i}`).

Trials which contain artifacts (indicated by `ArtifactSelection`, see Section 6.1) are cut out in the optimization step, but not in the evaluation step. `accuracyWOA` contains the resulting classification accuracies when artifact contaminated trials are also cut out in the evaluation step.

entry	**type**	**dimension**	**description**
`methods`	cell array	instances	optimized methods
`finalScore`	double	scalar	fitness score of the best individual
`scores`	double matrix	gaGen. × gaPop.	fitness scores in all generations
`accuracy`	cell array	evaluationFiles	test results
`accuracyWOA`	cell array	evaluationFiles	test results (without artifacts)
`optTime`	double	scalar	duration of the optimization step [s]
`evalTime`	double	scalar	duration of the evaluation step [s]

Table 6.6: entries of the test_result.result structure

Chapter 7

Discussion

7.1 Individual Comparison

7.1.1 Spatial Filters

According to Figure 5.2, it can be said that for AAR, BP, HJORTH, MVAAR and TDP a bipolar spatial filter results in a better mean classification accuracy than other spatial filters. This is also true when comparing the medians in Table 5.3. For all named methods, a ranking of the spatial filters regarding the classification accuracy looks as follows: bipolar > Laplace > CAR > monopolar. It is nearly the same situation with BAAR, except that bipolar and Laplace are interchanged. Of course, the standard deviations are high (cf. Table 5.2). For PLV, CAR and monopolar spatial filters work better than bipolar and Laplace spatial filters, this is exactly the opposite in contrast with other methods. For all methods except PLV, it can be said that a bipolar spatial filter using C3/C4 and the corresponding *anterior* electrodes give better results than using the corresponding *posterior* electrodes (see mostly used channel sets in the tables in Section 5.1.2). One can conclude that the information sources (mainly ERD/ERS effects) are not located directly under C3 and C4 electrodes, because in that case it would not matter if an anterior or posterior bipolar electrode montage is used. They are rather located under the anterior electrodes (2 and 6 in Figure 4.2). This is consistent with the current opinion that the most discriminative information – regarding a BCI system as mentioned in the introduction – can be extracted from the primary motor cortex which is located below these electrodes. The Laplace spatial filter extracted information under C3 and C4 electrodes. Maybe this is a disadvantage and led to the lower classification accuracies of the Laplace spatial filter.

7.1.2 Best Feature Extraction Method

From Figure 5.1 it can be seen that the TDP method reaches the highest mean classification accuracy (0.78), and the MVAAR method reaches the highest median of the classification accuracies (0.83). The median of the TDP method is 0.82. Because TDP

- reaches the highest mean classification accuracy,
- nearly the same median value as MVAAR,
- has only few parameters,
- and is cheap from a computational point of view,

it must be concluded that TDP is the ideal method for extracting features. Of course, this is only valid for comparable paradigms and an LDA classifier.

7.1.3 Effects of Feature Extraction Methods and Spatial Filters

A repeated measures ANOVA was used to test for main and interaction effects of the factors "method" and "spatial filter" on the classification accuracy. Table 5.8 shows that each factor has a main effect and also an interaction effect exists (PLV works better with CAR and monopolar spatial filters, but not other methods). Therefore, it is important which feature extraction method and spatial filter are chosen. Tukey's test was used to find out which groups differ in their mean value, but it is not feasible to show all test results. When using the best spatial filter for each method (cf. Table 5.9), then there is only a significant difference between PLV and {BP, TDP}. Furthermore, the spatial filters in combination with the TDP method have been compared with Tukey's test (Table 5.10). Significant differences have been found between monopolar and {bipolar, Laplace} spatial filters. The spread of the classification accuracies in each group is high (e.g. see Figure 5.1). Tukey's test has probably not reached significance, because a set of 9 subjects is too small.

7.1.4 Parameters

Section 5.1.2 analyzes the parameters found in the optimization step. It is remarkable that HJORTH and TDP extracted information from similar frequency bands, as well as AAR, BAAR and MVAAR. This is due to similarities in their algorithms. Based on the found frequency limits of their preceding band-pass filter, there are 3 groups: {HJORTH, TDP}, {AAR, BAAR, MVAAR} and PLV. BP had no preceding band-pass filter, because band-pass filters are integrated in the method itself. According to Figures 5.3 and 5.4, BP uses the alpha band and frequencies close to 22 and 34 Hz. Further, the optimal update coefficient (UC) of the AAR, BAAR and MVAAR methods is dependent on the chosen update method of the process noise covariance matrix (Q mode).

7.2 General Comparison

7.2.1 Classification Accuracies

Figure 5.5 shows the classification accuracies when the feature extraction methods have been optimized and a classifier has been trained in a general manner. The ranking regarding the mean classification accuracy is similar to Figure 5.1, only the AAR-like methods are interchanged. The bipolar spatial filter shows again the best mean classification accuracy for all methods except PLV. This is also valid regarding the medians (see Table 5.20). Monopolar and CAR spatial filters yield better results for the PLV method (cf. Figure 5.6). The influence of the spatial filters is very similar as when optimizing and training individually (compare Figure 5.2 and 5.6). TDP with a bipolar spatial filter results in the highest mean and median of the classification accuracies.

7.2.2 Effects of Feature Extraction Methods and Spatial Filters

The repeated measures ANOVA indicates that there exist main and interaction effects of the factors "method" and "spatial filter" (see Table 5.25). When using the best spatial filter for each method, significant differences were only found between PLV and TDP

(see Table 5.26). When using the TDP method, significant differences were found between monopolar and {bipolar, Laplace} spatial filters. However, there were probably not enough subjects to make sound statements (finding more significant differences).

7.2.3 Individual vs. General

The individual and the general comparison yield qualitatively similar results (Section 7.2.1). It is now interesting if they also give the same results quantitatively. A repeated measures ANOVA with three factors "method", "spatial filter" and "comparison" was executed. "comparison" indicates if the data come from the individual comparison or the general comparison approach. "comparison" shows no significant effect. The interaction effect between "comparison" and "method" has a low p-value of 0.06. This is due to the fact that the mean classification accuracies of all methods got worse (but not significantly) except with BAAR and PLV where it stays nearly a the same level. This is not shown explicitly in a diagram or table. It could be concluded that it does not have any impact if a general optimization and training is used instead of an individually one. Nevertheless, the means in Table 5.18 are always slightly worse than in Table 5.1, except at BAAR/bipolar. This difference is even more apparent when considering the medians in Table 5.20 and 5.3 (e.g. TDP and BP bipolar). The medians in the general comparison are lower, except in 5 cases than in the individual comparison, especially with bipolar spatial filters. In summary, it must therefore be concluded that it is beneficial to optimize and train individually. Although it is unknown whether an individual optimization or an individual training or both yield higher classification accuracies.

7.2.4 Parameters

The found feature extraction parameters are shown in Section 5.2.2. AAR and TDP used comparatively small frequency bands (1–16 Hz and 3–13 Hz). TDP reached the highest mean and median of the classification accuracies, AAR results are similar (Figure 5.5). The discriminative information lies therefore mainly in lower frequency bands.

7.2.5 About the Test

In order to evaluate the generalizing properties of a feature extraction method, one should use different subjects for the optimization and evaluation step. Otherwise, the feature extraction method could adapt to subject specific properties, and perform better than usual in the evaluation step. However, this subject separation was not done. If 9 subjects were divided into an optimization and evaluation group, these groups would have been very small. Due to the high variances in the final classification accuracies (see Section 5.2.1), unreliable results would follow. Furthermore, the optimization data of the subject, against that the classifier was tested in the evaluation step, constituted only 1/9 of the entire optimization data. This limited the aforementioned adaption effect.

7.3 Combination of Methods

The mean and median values of the classification accuracies of the combined methods are 0.78 and 0.81, respectively. This is close to the mean and median values of the TDP method

with a bipolar spatial filter from the individual comparison: 0.78 and 0.82, respectivly. A paired t-test showed (see Section 5.3.1) that there is no significant difference in their mean values. The conclusion is that a method combination has no advantage, but it has high computational costs. One must keep in mind that some constraints regarding the optimization step were used, see Section 4.3. Therefore, the GA had not full freedom in finding an optimal solution. Furthermore, this conclusion is only true for the META combination method together with an LDA classifier. Maybe other combination methods and/or classifiers would lead to a significant advantage when combining methods.

7.4 About the Classifier

Linear discriminant analysis (LDA) is often used synonymous with Fisher's linear discriminant analysis (FLDA), so also in this work. Nevertheless, there is a difference between these methods: the FLDA does not assume normally distributed classes or equal class covariances. The classifier used in this work was an FLDA classifier [1]. Therefore, a bias in the classification accuracies, resulting from the distribution of the extracted features, does not exist. The TDP and BP methods used from the BioSig toolbox implemented a log-transformation to compute Gaussian distributed features. This was strictly speaking unnecessary. However, such a transformation is not harmful (log is a strictly increasing function).

One must keep in mind that all classification accuracy results are *only valid* with the used LDA classifier. The results could be different for e.g. support vector machines (SVMs). The LDA classifier was chosen because it is used widely in practice and it is computationally cheap.

To obtain sound classification accuracies, it is very important that the classifier is tested with *unseen* data. For that reason, the second session of the subjects was only used for testing the classifier and nothing else.

7.5 About the Genetic Algorithm

The optimization problem is quite complex: different types of parameters for a feature extraction method exist (integers, real numbers, bit strings, nominal numbers), and even different numbers of methods, each with a different channel/parameter set, could be combined. This search space could easily be too large for exhaustive search. It is further not possible to calculate the derivation of an error function and therefore gradient-based optimization methods cannot be used. Fortunately, metaheuristic optimization algorithms can be used. They try to iteratively improve a certain solution with regard to an objective function. The used genetic algorithm (GA) is such a metaheuristic optimization algorithm. A representation of an individual has been found which had the power to express the demanded complexity (see Section 3.2). This model arises in an intuitive manner when using a GA, but it is not so clear how this complexity can be handled with e.g. particle swarm optimization (PSO) [10]. Of course, crossover and mutation operations have to be adapted. Two points are important for the speed of the GA. First, the representation of the individual and the crossover operation must facilitate that two good parents could result in an even better child. The crossover operation must be in some sense meaningful.

Second, the representation of an individual and the mutation operation should be defined so that even single mutations lead to a better fitness score. If only a certain combination of mutations lead to a better fitness score, it lasts longer until the GA finds that combination. Therefore, an individual should not be more complex than necessary.

A disadvantage of metaheuristic algorithms is that they cannot guarantee that a global optimum has been found. The optimum found could also be a local optimum. Therefore, it is crucial when applying a GA to analyze the behavior on the current problem set. In practice, this will be a trial-and-error tuning of the meta parameters (e.g. mutation rate, selection function, crossover fraction, stopping criterion) with a smaller population set. When the correct values have been found, the GA is started with the full population set. The use of the tournament selection function also minimizes the risk of finding a local optimum (see Section 3.1).

Bibliography

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[2] C. Brunner, *Analysis of the electroencephalogram with an adaptive non-linear model*, Master's thesis, Graz University of Technology, 2003.

[3] C. Brunner, R. Leeb, G. Müller-Putz, A. Schlögl, and G. Pfurtscheller, *Bci competition 2008 - graz data set a.* `http://www.bbci.de/competition/iv/desc_2a.pdf`, 2008.

[4] C. Brunner, R. Scherer, B. Graimann, G. Supp, and G. Pfurtscheller, *Online control of a brain-computer interface using phase synchronization*, IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, 53 (2006), pp. 2501–2506.

[5] G. Dornhege, B. Blankertz, G. Curio, and K.-R. Müller, *Boosting bit rates in noninvasive eeg single-trial classifications by feature combination and multiclass paradigms*, IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, 51 (2004), pp. 993–1002.

[6] E. Gysels and P. Celka, *Phase synchronization for the recognition of mental tasks in a brain–computer interface*, IEEE TRANSACTIONS ON NEURAL SYSTEMS AND REHABILITATION ENGINEERING, 12 (2004), pp. 406–415.

[7] B. Hjorth, *Eeg analysis based on time domain properties*, Electroencephalography and Clinical Neurophysiology, 29 (1970), pp. 306–310.

[8] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press (reprinted in 1992 by The MIT Press), 1975.

[9] R. E. Kalman, *A new approach to linear filtering and prediction problems*, Transactions of the ASME-Journal of Basic Engineering, 82 (1960), pp. 35–45.

[10] J. Kennedy and R. Eberhart, *Particle swarm optimization*, Proceedings of IEEE International Conference on Neural Networks, 4 (1995), pp. 1942–1948.

[11] T. M. Mitchell, *Machine Learning*, McGraw-Hill Companies, Inc., 1997.

[12] E. Niedermeyer and F. L. D. Silva, *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*, Lippincott Raven, 5 ed., November 2004.

[13] G. Pfurtscheller, B. Z. Allison, C. Brunner, G. Bauernfeind, T. Solis-Escalante, R. Scherer, T. O. Zander, G. Mueller-Putz, C. Neuper, and N. Birbaumer, *The hybrid bci*, Frontiers in Neuroscience, 4 (2010).

[14] G. Pfurtscheller and F. H. L. D. Silva, *Event-related eeg/meg synchronization and desynchronization: basic principles*, Clinical Neurophysiology, 110 (1999), pp. 1842–1857.

[15] A. Schlögl, *The electroencephalogram and the adaptive autoregressive model: Theory and applications*, PhD thesis, Graz University of Technology, 2000.

[16] C. Vidaurre, N. Krämer, B. Blankertz, and A. Schlögl, *Time domain parameters as a feature for eeg-based brain–computer interfaces*, Neural Networks, 22 (2009), pp. 1313–1319.

[17] G. Welch and G. Bishop, *An introduction to the kalman filter*. `http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf`, July 2006.

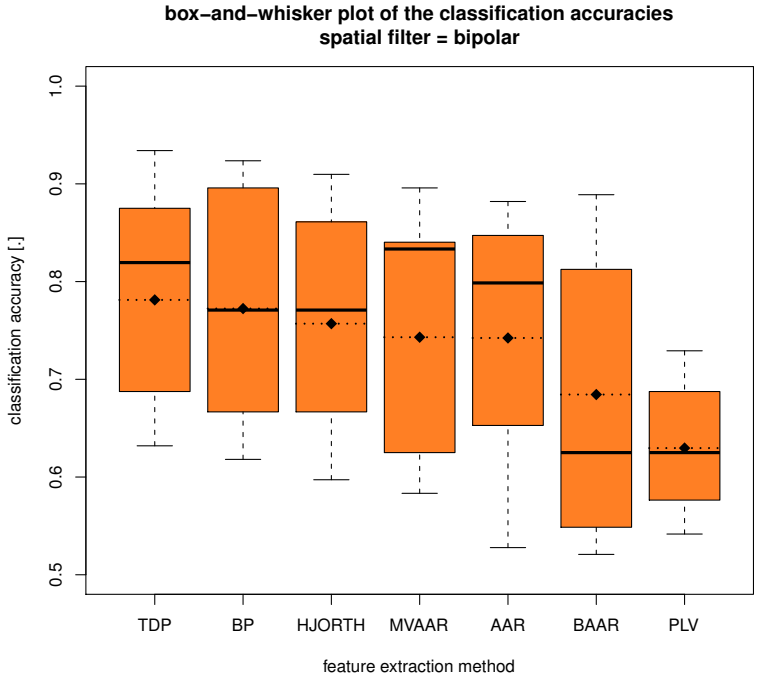
[18] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, *Brain-computer interfaces for communication and control*, Clinical Neurophysiology, 113 (2002), pp. 767–791.

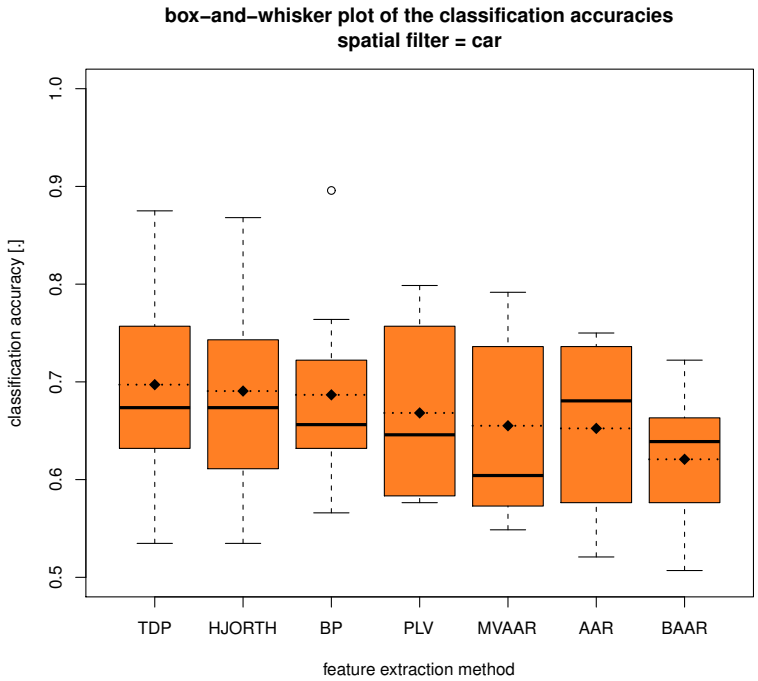
Appendix A

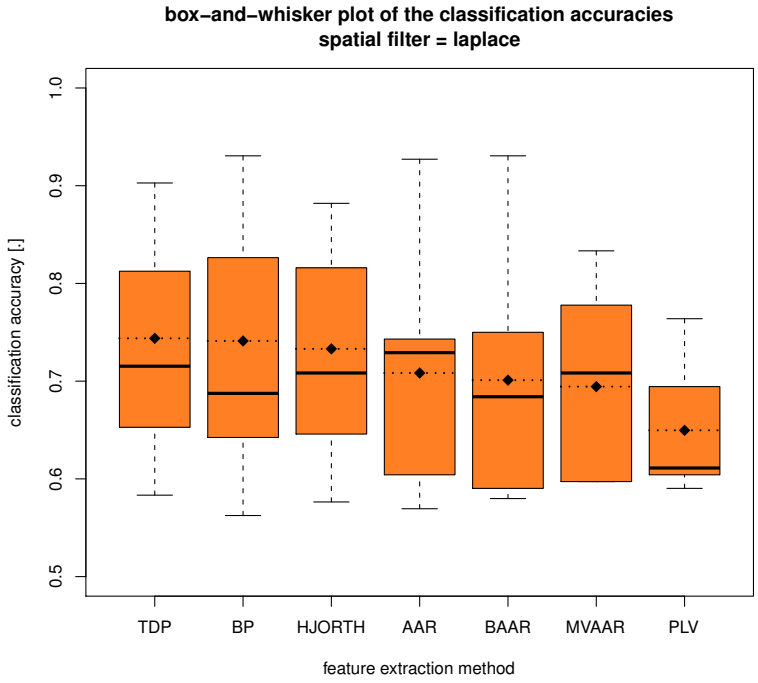
Additional Results

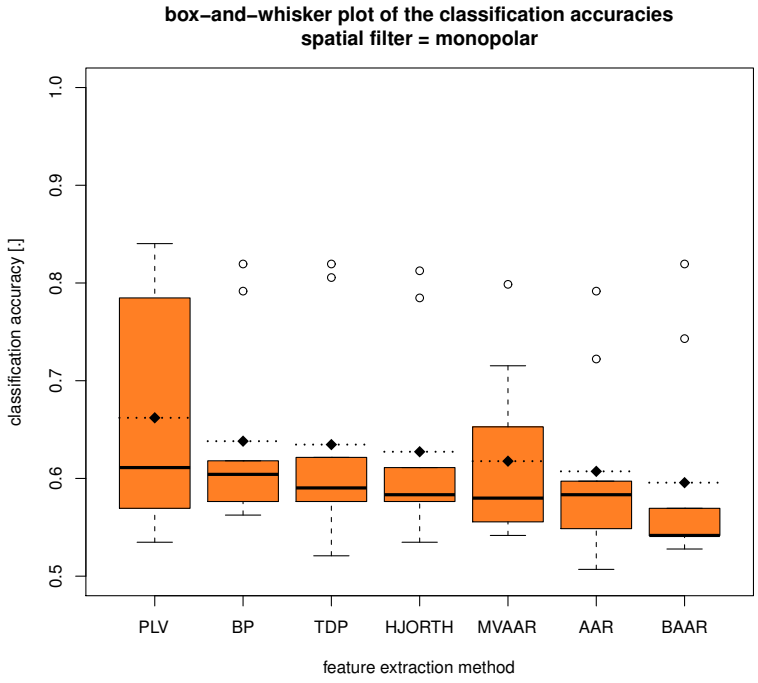
Not all results are shown in sections 5.1 and 5.2. Box-and-whisker plots of the classification accuracies from all methods and all spatial filters are shown below.

A.1 Individual Comparison

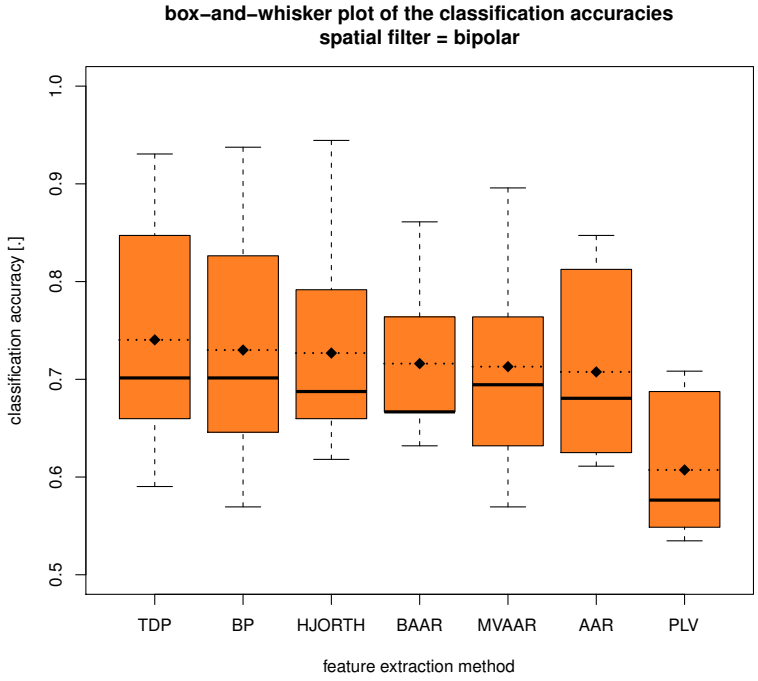


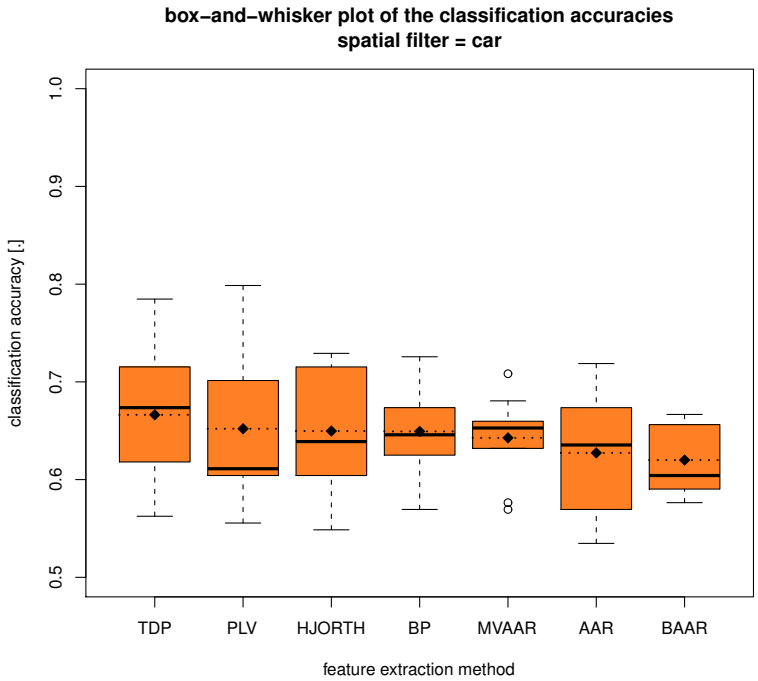


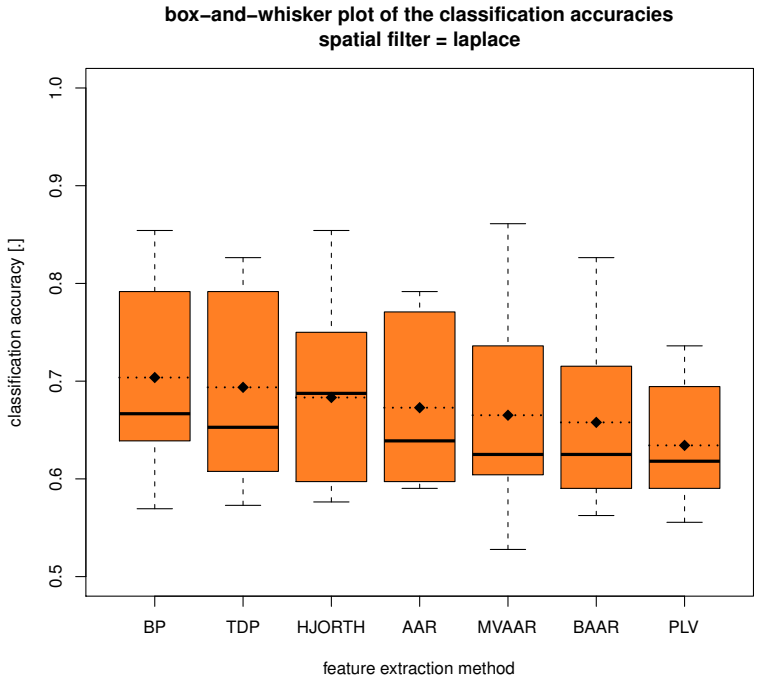

box−and−whisker plot of the classification accuracies
spatial filter = laplace
classification accuracy [.]
0.5
0.6
0.7
0.8
0.9
1.0
TDP
BP
HJORTH
AAR
BAAR
MVAAR
PLV
feature extraction method

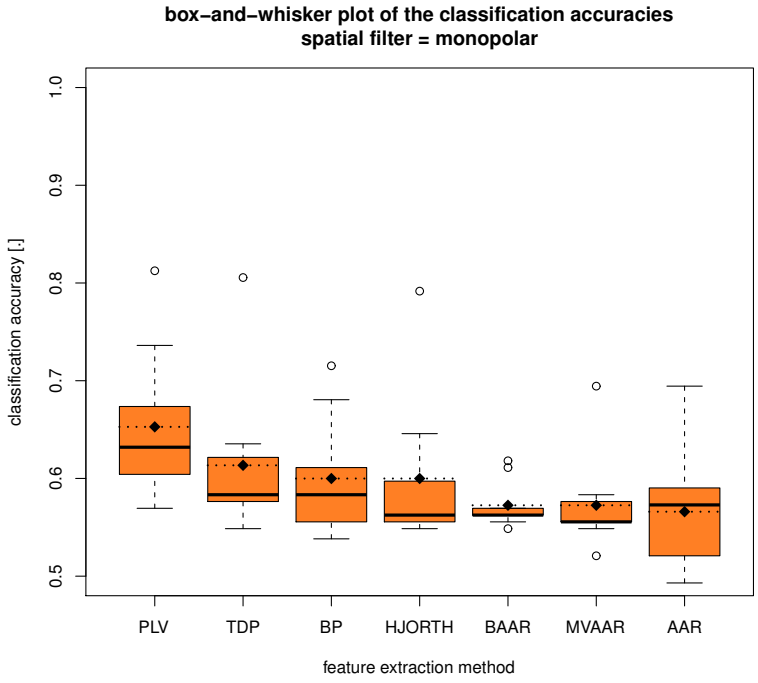

box−and−whisker plot of the classification accuracies
spatial filter = monopolar
classification accuracy [.]
0.5
0.6
0.7
0.8
0.9
1.0
PLV
BP
TDP
HJORTH
MVAAR
AAR
BAAR
feature extraction method

A.2 General Comparison






box−and−whisker plot of the classification accuracies
spatial filter = laplace
classification accuracy [.]
0.5
0.6
0.7
0.8
0.9
1.0
BP
TDP
HJORTH
AAR
MVAAR
BAAR
PLV
feature extraction method


box−and−whisker plot of the classification accuracies
spatial filter = monopolar
classification accuracy [.]
0.5
0.6
0.7
0.8
0.9
1.0
PLV
TDP
BP
HJORTH
BAAR
MVAAR
AAR
feature extraction method