David Steyrl

On the suitability of Random Forests for detecting mental imagery for non-invasive Brain-Computer Interfacing

Diploma thesis

Institute for Knowledge Discovery
Laboratory of Brain-Computer Interfaces
Graz University of Technology
Krenngasse 37, 8010 Graz, Austria
Head: Assoc. Prof. Dipl.-Ing. Dr.techn. Gernot Müller-Putz

Supervisor:
Ass. Prof. Dipl.-Ing. Dr.techn. Reinhold Scherer

Evaluator:
Assoc. Prof. Dipl.-Ing. Dr.techn. Gernot Müller-Putz

Graz, February 2012

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)

Acknowledgments

I would like to use this chance, to thank all the people who helped and/or supported me.

At first, Prof. Reinhold Scherer, Prof. Gernot Müller-Putz and the other staff from the "Institute of Knowledge Discovery" for their confidence in me and my ability to solve this challenge and for their patience to answer all my questions.

In equal measure, I want to express my thanks to my family, my mother, my father and my brother for making this work possible.

And finally, all my friends and my colleagues for the good times.

Thank you very much indeed!

Abstract

On the suitability of Random Forests for detecting mental imagery for non-invasive Brain-Computer Interfacing.

Brain-Computer Interfaces (BCIs) are devices that directly convert a user's brain activity into actions. One class of BCI is based on the detection of changes in oscillatory activity of non-invasive electroencephalographic (EEG) signals. Motor imagery (MI) is typically used to induce such changes, and machine learning and pattern recognition methods for translating corresponding EEG activity pattern into messages for devices.

The aim of this thesis is to explore the usefulness of the Random Forests classifier (RF) for the classification of MI tasks. The RF classifier is an ensemble classifier, which consists of many uncorrelated decision trees. The output of the RF classifier is chosen by a vote. To ensure more diverse votes, each decision tree is built up by randomized parameters.

The RF method was applied to EEG data recorded from ten able-bodied subjects while performing left hand (L), right hand (R) and feet (F) MI. The results of extensive offline cross-validation tests and offline BCI simulations suggest that RFs are suitable for the classification of oscillatory EEG activity patterns. Peak (mean ± std computed by averaging the peak accuracies for each subject) accuracies of 82% (59 ± 14%) for the 3-class problem, and 93% (67 ± 15%) for L vs R, 91% (77 ± 12%) for L vs F and 94% (77 ± 10%) for R vs F, respectively, were computed. The calculated results are comparable with state-of-the-art methods used in BCI research. Furthermore, online feedback experiments were performed with three able bodied subjects. Two were able to successfully operate the BCI. Subjects achieved peak accuracies of 92% and 88%, respectively.

Key Words: Brain-Computer Interface (BCI), Random Forests classifier, Machine Learning, Electroencephalogram (EEG), Event related desynchronization/synchronization (ERD/ERS).

Kurzfassung

Über die Anwendbarkeit von Random Forests Klassifikatoren zur Detektion von Bewegungsvorstellungen für nichtinvasive Gehirn-Computer-Kommunikation.

Gehirn-Computer Schnittstellen (BCI) sind Geräte die Gehirnaktivität direkt in Handlungen umsetzen. Eine Klasse von BCI basiert auf Änderungen in der oszillatorischen Aktivität von nichtinvasiven elektroenzephalographischen (EEG) Signalen. Bewegungsvorstellungen (MI) werden typischerweise verwendet um diese Änderungen hervorzurufen, welche durch maschinelle Lernalgorithmen in Befehle für Geräte übersetzt werden.

Das Ziel dieser Arbeit ist die Anwendbarkeit von Random Forests (RF) Klassifikatoren zur Klassifikation von MI zu untersuchen. Der RF Klassifikator ist ein Ensembleklassifikator der aus vielen unkorrelierten Entscheidungsbäumen besteht. Die Entscheidung eines RF Klassifikators wird mittels Abstimmung bestimmt. Um verschiedene Stimmen sicherzustellen, werden für die Erstellung der Entscheidungsbäume Zufallsparameter verwendet.

Die RF Methode wurde auf EEG Daten angewendet, welche während linker Hand (L), rechter Hand (R) und Füße (F) MI von zehn gesunden Probanden aufgezeichnet wurden. Die Resultate von Kreuzvalidierungstest und BCI Simulationen legen nahe, dass RF Klassifikatoren geeignet sind oszillatorische Muster in EEG Daten zu klassifizieren. Maximale Genauigkeiten (Mittelwert ± Standardabweichung, errechnet durch Mittelung der Maximalgenauigkeiten der Probanden) von 82% (59 ± 14%) im Fall von drei Klassen und 93% (67 ± 15%) für L gegen R, 91% (77 ± 12%) für L gegen F und 94% (77 ± 10%) für R gegen F wurden errechnet. Die berechneten Ergebnisse sind vergleichbar mit denen von aktuell im BCI Bereich eingesetzten Klassifikationsalgorithmen. Des weiteren wurden online Feedback Experimente mit drei gesunden Probanden durchgeführt. Zwei waren in der Lage mit dem BCI zu arbeiten. Sie erreichten Maximalgenauigkeiten von 92% bzw. 88%.

Schlüsselwörter: Gehirn-Computer-Schnittstelle (BCI), Random Forests Klassifikator, Maschinelles Lernen, Elektroenzephalogramm (EEG), Event related desynchronization/synchronization (ERD/ERS).

Table of Contents

1 Introduction

The topic of Brain-Computer Interfaces (BCI) is a relatively young, substantially growing field of research which attracts a lot of scientists from many disciplines [1]. The aim of a BCI is to provide instructions from the brain to a computer [2]. In some cases, for example for patients with locked-in-syndrome, this can be the only alternative left to communicate with the outside world. In other cases such a possibility for providing instructions can lead to easements in the everyday life as it may be used to control different, commonly computer based, applications for communication [3] [4] [5] or environmental control [6]. The general approach for a BCI is to measure the brain activity in an invasive or non-invasive way and to use these measurements to process control signals (see Figure 1.1) [1].

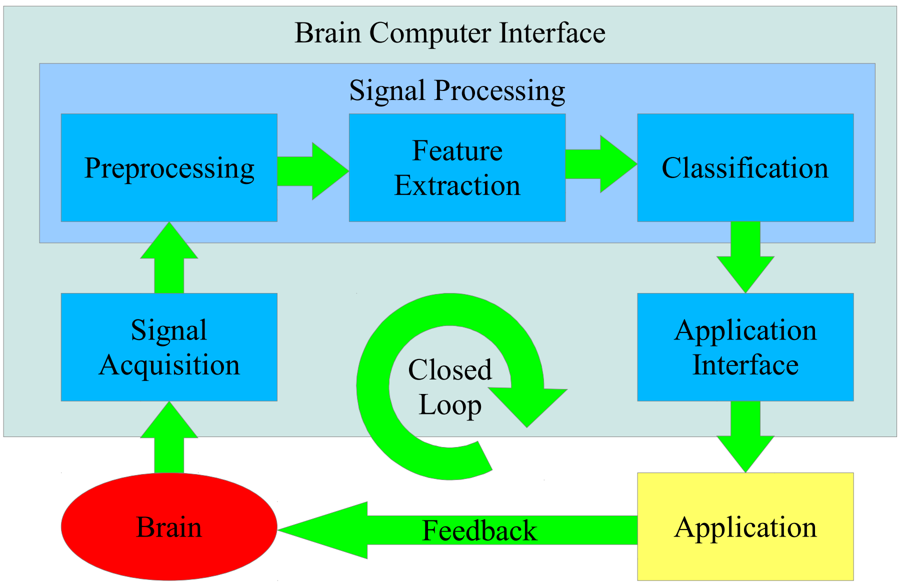

Figure 1.1: Schematic representation of the functional blocks of a BCI [7] [8].

There are many possible methods for measuring the brain activity [1]. A basic criteria for the measurement method of an online BCI is a high temporal resolution. Commonly, users do not want to wait for a long time to set commands, and in some applications a quick reaction is necessary for the task (e.g. control of machines). A second criteria is that the measurement method should be non-invasive. Non-invasive means that no break in the skin is created.

Invasive methods have to deal with some disadvantages, for example, the surgery itself or inflammation reactions of the body. Furthermore, the method should be reliable, fast to setup and easy to use. The electroencephalogram (EEG) [9] fulfills a lot of these requirements well and is therefore widely used in the field of BCI research [1], including in this work.

A precondition for a BCI is that the measurements of the brain activity contains components which can be deliberately modified by the user. In recent a multiplicity of different neurological phenomena have been investigated to address this precondition [1]. One class of BCIs uses changes in the power of specific frequencies of the EEG caused by imagery of movements as features [1]. These oscillatory changes, induced by different motor imagery (MI), varies the power of the Mu rhythm (the Mu rhythm indicates the resting state of motor neurons) in the somatotopic correlated areas of the motor cortex [10]. This process is called "event-related desynchronization and event-related synchronization (ERD/ERS)" and is described in [11]. Due to the different positions in the somatotopic correlated areas, MIs of different parts of the body produce distinguishable EEG measurements (see Figure 1.2) [10].

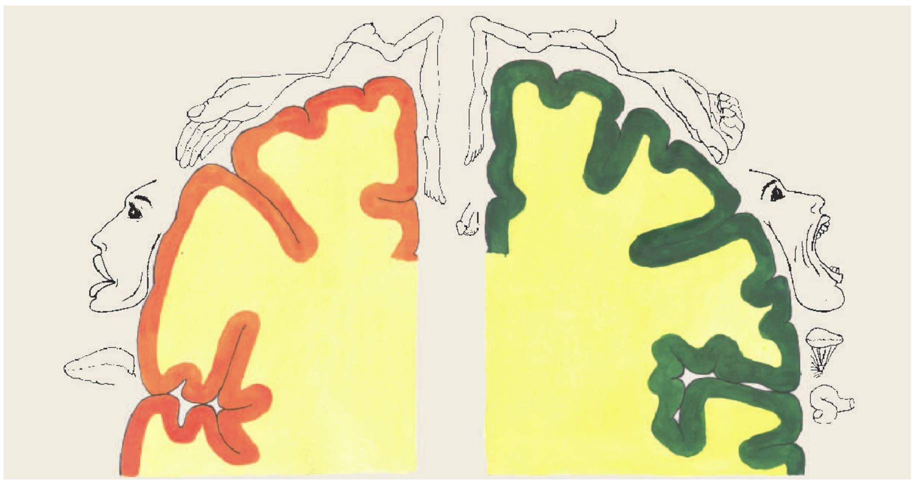
Figure 1.2: Geometric mapping between body parts and motor/somatosensory cortex [12].

In the past different machine learning algorithms were reviewed on their ability to classify oscillatory changes in EEG measurements caused by MIs [13]. (Classifiers use observations of EEG measurements during different MIs as examples for finding a differences scheme between them.) For this task, the algorithm needs training observations, which consist of two parts. On the one hand the features (the power of the frequencies which are modified by the MI) and on the other hand the class (the according MI). The algorithm will "compare" a new observation with the previous found scheme to find the corresponding MI. Nevertheless, the problem of classification of MI induced oscillations is not solved satisfactorily so far. The accuracies of the classifiers are high, but the results were obtained through a high effort of partly manually optimizations (e.g. screening for the best features of a subject) [14]. Furthermore due to the non-stationarity of the brain activity, the optimizations can become ineffective after some time and the optimizations may have to be repeated. Therefore, it makes sense to analyze how the characteristics of a classifier could help to overcome this problem.

The aim of this work is to investigate, whether a Random Forest (RF) classifier is able to detect the correct MI from the EEG data or not. This is especially important as literature research brought up that RF classifiers addresses some of the drawbacks of other classifiers [15].

Section 2 of this work, will cover the history and the theory of RF classifiers. Thereafter, section 3 will be about the methods used. Subsequently, section 4 will answer which parameters of the RF classifier will fit the MIs classification task best. Section 5 will cover offline tests and, finally, section 6 will demonstrate the online suitability of the classifier.

2 History and Theory of Random Forests

The Random Forests (RF) classifier was developed by Leo Breiman († 2005) and published in the paper “Random Forests” in 2001 [16]. The algorithm was based on many ideas and papers written by Breiman (e.g. “Bagging predictors” in 1996 [17]) and others, like Tin Kam Ho's papers on “Random decision forests” in 1995 [18] and on “The random subspace method for constructing decision forests” in 1998 [19]. Breiman's paper “Bagging predictors”, in which he described the method of “bootstrap aggregating”, also called “bagging”, is one of the most important fundamental papers. In this paper, randomly drawn observations (bootstrap) are used to build a lot of classifiers. A voting of all classifiers then chooses the class (aggregating). Using decision trees as classifier in “bagging” leads to RF [15].

A RF classifier is an ensemble classifier, which consists of many decision trees (see an example tree in Figure 2.1). In order to find a decision, each tree gets a vote. The decision of the RF classifier equals the majority decision of the decision trees. Or expressed mathematically:

$$\hat{C}_{rf}^{B}(x) = \textit{majority vote}\left\{\hat{C}_b(x)\right\}_1^B \quad . \quad (1)$$

The prediction $\hat{C}_{rf}$ of the RF (rf) classifier with B trees equals the majority decision of the predicted classes C_b of each of the B trees, where x is the observation vector. However the prediction of a RF classifier is only better than the prediction of a single tree, if the tree classifiers are built differently. If they are identical, the decision of the forest would be equal to the decision of a single tree and a forest would not be necessary. To understand how the creation of a tree works, it is necessary to explain first, how a decision tree work.

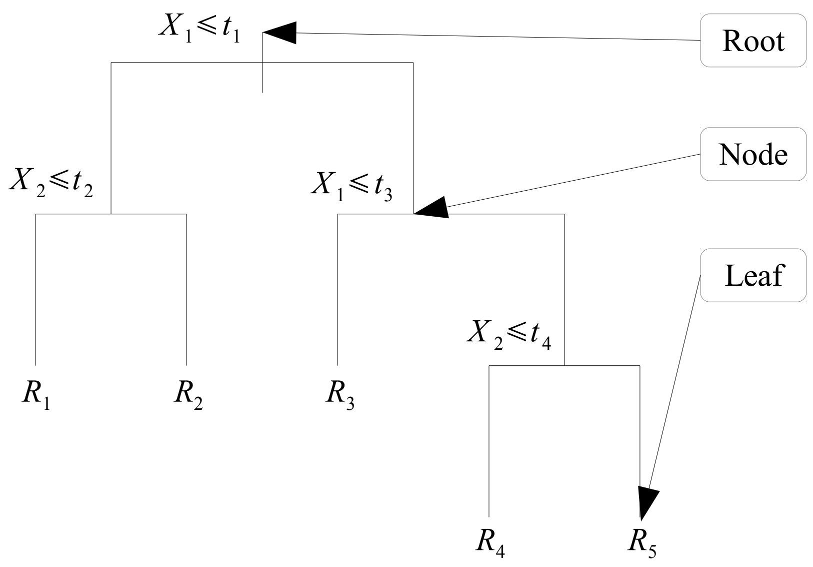

Figure 2.1: A possible decision tree (modified from [15]).

A decision tree partitions the feature space into a set of rectangles (see Figure 2.2). Then a simple model for each rectangle is fitted, normally using a constant value. This can be mathematically expressed as:

$$\hat{C}_t = \sum_{m=1}^{M} c_m I(x \in R_m) \quad . \qquad (2)$$

The prediction $\hat{C}_t$ of the decision tree classifier is the value of c of the region (R) m in which the observation vector x end up after its path through the decision tree (see Figure 2.1 and Figure 2.2).

This partitioning of the feature space and the fitting of the simple models is called "growing of the tree" and can be performed iteratively with the following algorithm according to [15]:

1. Start at the Root with all training observations.
2. Use the trainings observations to calculate the "best" split criterion at this node.
3. Split the node into two daughter nodes.
4. Allocate the trainings observations to the daughter nodes, according to the split criterion.
5. Continue at point 2 for each node.

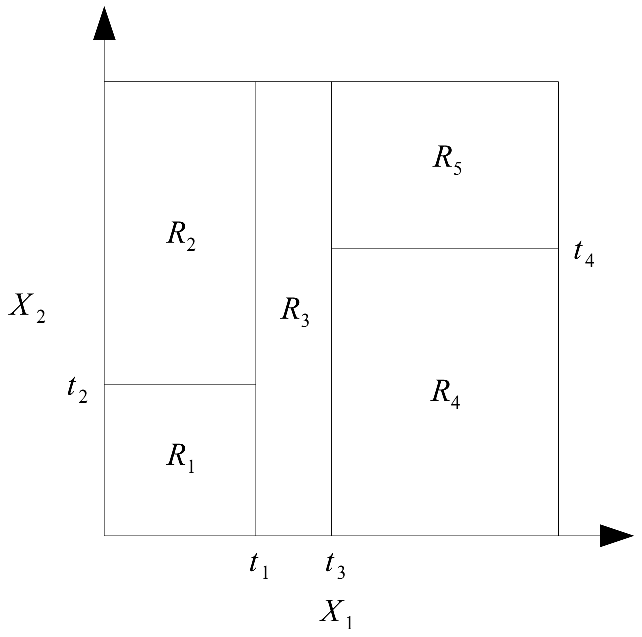

Figure 2.2: Possible partition of the feature space by a decision tree (modified from [15]).

The above described algorithm grows the tree iteratively and will stop if there is only one trainings observation left at each node. Such a node is then called a leaf and has the value of the class label of the training observation. A fully grown tree will highly over-fit the problem. For practical use such a tree has to be pruned back. During this procedure some of the leaves are cut off and the corresponding training observations of the pruned leaves are allocated to the new leaf, which now contains many training observations. The value of the new leaf is the mean of all values of the class labels of the training observations in this leaf. The search for the best pruned back tree is done by cross-validation, keeping in mind that this whole procedure works iteratively, global optimization is being done. The algorithm always uses the best criterion for a specific node, but there are many cases where a worse split criterion in one situation would lead to a better accuracy in the end. These two problems, the iterative procedure and the over-fitting, are the main reasons for the relatively bad accuracies of decision trees. Pruning is only a solution for the over-fitting problem, but not for the iterative procedure. In an iterative procedure, where a global view is missing, the "best" split criterion is:

> *"For each node, find splitting dimension and point to maximize proportion of class k observations in node m, when class k is the majority class in node m."* [15]

This means that it is the best to use the feature which contains the most information about the class membership of the training observations. There are different approaches for this task. Very often, an index is calculated and the minimization of the index is then the target. There are different indexes with different properties:

- Misclassification Error (MCE)
- Gini-Index
- Cross-entropie or diviance

The Gini-index is used in the very popular CART algorithm (Classification And Regression Trees, invented by Leo Breiman in 1983 [20]). The decision trees in a RF classifier are built up with the CART algorithm. Therefore only the CART algorithm and its Gini-index will be considered. The Gini-index is defined as follows:

$$Gini\,index = \sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk}) \quad , \qquad (3)$$

in which $\hat{p}_{mk}$ is the proportion of class k (k^{th} class of all K classes) observations in node m:

$$\hat{p}_{mk} = \frac{1}{N_m} \cdot \sum_{x_i \in R_m} I(y_i = k) \quad , \ (4)$$

with I as the identity function (is one if condition is true), N_m as the number of training observations in node m, x_i as feature vector of observation i, y_i as class of observation i and R_m as region m. For each node, calculate the Gini-index for the splitting points (dimension and point in dimension) and choose the minimum. This yields the best split point. Now the decision trees can be created.

As already mentioned, decorrelated trees are mandatory to improve the accuracy of the RF classifier compared to a single decision tree. To grow decorrelated trees some parameters need to be randomized. First of all, an individual set of training observations, which is generated by drawing with replacement (also called a bootstrap set) is assigned to each tree. The size of the bootstrap set is equivalent to the size of the trainings observations set. Secondly, at each node only a number of random features are used to calculate the split criterion. With this randomness it is possible to create an enormous amount of decision trees, which are all different from each other.

Besides classifying, the RF classifier offers a number of other useful options. After the training of the RF classifier, an approximation of the importance of the features and the expected error is calculated.

The approximation of the importance of the features is performed in the following way:

Separately sum up the improvements in the split criterion in each node of each tree for each feature. Divide each value by the number of trees. In other words this is the mean improvement in accuracy induced by each feature. Features with high mean improvements are important features.

The approximation of the expected error for regression is calculated this way:

> *"For each observation* $z_i=(x_i, y_i)$ *, construct its Random Forest predictor by averaging only those trees corresponding to bootstrap samples in which* z_i *does not appear."* [15]

In case of classification, no averaging is done, but the approach is the same. For each observation $z_i=(x_i, y_i)$, construct its own RF classifier using only those trees for voting, which correspond to bootstrap sets not containing z_i . The result is quite similar to cross-validation, if the number of trees is big enough. These expected errors are called Out-Of-the-Box errors (OOB errors).

3 Methods

3.1 Experimental paradigm

Due to the necessity of a controlled conduct of the experiments a paradigm was set up. One effect of the usage of a paradigm was that the corresponding MI (the class) of a feedback period was known. This was realized by an instructing cue. In the case of the paradigm for the EEG measurements used in Section 4 and 5, a brief explanation is presented below. These EEG measurements were originally recorded for [14], therefore a detailed description can be found there.

Ten able bodied subjects (6 males and 4 females) were sitting in a chair in front of an LCD display, which provided randomized instructions for different MIs. A single trial consisted of the following steps (see also Figure 3.1):

- Second 0: A cross showed up on the screen.
- Second 2: A beep sounded to focus the subject's attention.
- Second 3: One of three cues was displayed for 1.25 seconds. The three cues were: Right pointing arrow for right hand MI, left pointing arrow for left hand MI and down pointing arrow for feet MI.
- Second 4: The subject was instructed to keep the imagery until the cross disappeared.
- Second 8: End of the MI phase. A break with a randomized length (0.5 to 2.5 s) was the last part of the trial.

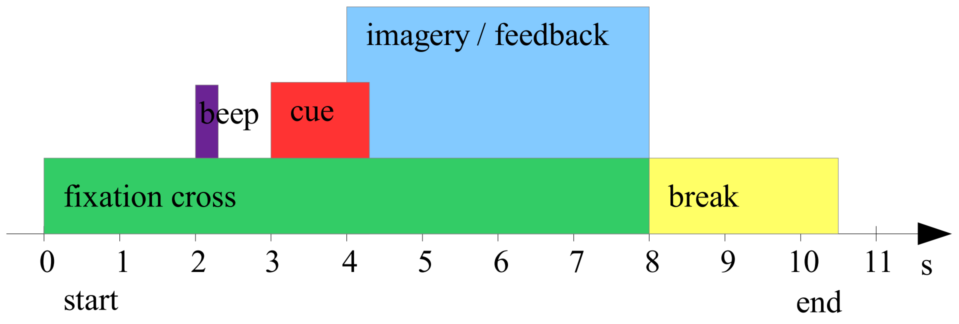

Figure 3.1: Recording paradigm for the EEG measurements used in Section 4 and 5.

One run consisted of 30 trials, ten for each class. A session consisted of 8 runs, resulting in 80 trials for each class.

3.2 Signal acquisition

The corresponding areas of the hands and the feet on the motor cortex in the international 10-20-system are the positions C3, Cz and C4 (Figure 1.2) [21]. Hence signals acquired from these positions were used.

In case of the EEG measurements used in Section 4 and 5, the EEG was recorded with 32 Ag/AgCl electrodes, which covered the sensorimotor area over C3, Cz and C4. The reference electrode was attached at the left and the ground electrode at the right mastoid (see Figure 3.2). The recording was performed with a monopolar amplifier (Synamps, Compumedics Germany GmbH, Singen, Germany) at a sampling rate of 1 kHz, including a bandpass with 0.05-200 Hz and a notch filter at 50 Hz. The full description of the signal acquisition can be found in [14].

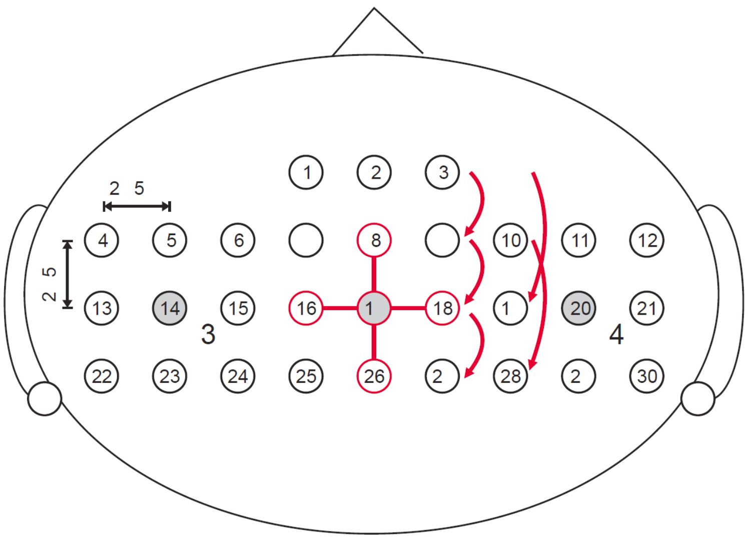

Figure 3.2: Mounting of the electrodes [14].

3.3 Signal preprocessing

Due to other bio-signals (e.g. electromyography signals), the EEG is afflicted with artifacts which cover the brain signals. In offline data a visual screening was performed to exclude contaminated parts of the EEG signal. This rejection was executed for the data used in Section 4 and 5.

EEG signals are often very noisy, hence the signal to noise ratio (SNR) is very low [12]. To improve the SNR of a specific electrode position, the Laplace derivation of this position was used instead of the raw signal [12]. The derivation was calculated sample-by-sample with:

$$C_{LAP} = C_{Center} - \frac{1}{4} \cdot \sum_{i=1}^{4} C_{surr,i} \qquad (5)$$

Where C_{Center} was C3, Cz and C4, respectively, and C_{surr} was one of the four surrounding electrode positions.

3.4 Feature extraction

The frequency of the Mu band can vary from one subject to another [11]. Hence a search for the specific frequency of the Mu band for each subject would be necessary. But, according to literature [15], RF classifiers can deal with features carrying no information about the class membership of an observation. Therefore a simple approach was chosen. The powers of each frequency between 1 and 40 Hz in steps of 1 Hz of each of the three electrode positions (C3, Cz and C4) were used as features for detection of MIs. In total, 120 features for an observation. The calculations of the powers of the frequencies were done as follows:

1. A period of one second was picked out of the signal.
2. The Fourier transformation (FT) was applied on the picked signal period.
3. The powers of the frequencies were obtained through squaring of the absolute values of the results of the FT.
4. For comparison with other classifiers the powers of the frequencies were used logarithmically.

A trainings observation z_i consists of 120 features x_i (the powers of the frequencies) and in addition a class label y_i .

3.5 Software & tools

For computations in Section 4 and 5 MATLAB 2010a (Mathworks Inc., Natick, USA) and in Section 6 MATLAB / Simulink 2011b (Mathworks Inc., Natick, USA) was used. The RF classifier was implemented by using the MATLAB RF package, downloaded from http://code.google.com/p/randomforest-matlab/ on Nov 26th 2010 (published under the GNU GPL v2 license). This package is using precompiled mex files for importing code from other languages to MATLAB. Here, the R implementation of the Random Forests classifier of Andy Liaw is used. This again, is a port from the original Fortran code by Leo Breiman and Adele Cutler.

4 Influence of parameters on the Random Forests classifier

Like any algorithm, there are many "adjusting screws" for the RF classifier which have an influence on the classification accuracy [15]. Several parameters are resulting out of the trainings procedure of the RF classifier (Number of trees, Tree correlation factor, Tree leaf size). Others are modifications on the trainings observations (Logarithmic power of frequencies vs power of frequencies, noisy observations). The finding of the best values for the parameters is a necessity for obtaining the highest classification accuracies.

4.1 Number of trees

4.1.1 Scope

It is not clear in advance how many trees are to be used, during training of an RF classifier. On the one hand, it is necessary to reach an amount that is high enough for stable voting results, but on the other hand a lot of trees are more complex with respect to computational effort. The OOB error could be a feature for a stop criterion. If the OOB error is not changing with the number of trees anymore, the training could be stopped. However, the OOB error is not obtained until the training is done. In practice a standard value is used for the sake of simplicity. Usually it is based on values of 500 to 1000 trees. 500 trees are used as the standard value in the used program library (see Section 3.5). 1000 is the amount of trees recommended by Leo Breiman [12]. Due to the fact that the training of a RF classifier using 500 trees is very fast (about 1 s on a 3 GHz CPU) lower amounts of trees were not investigated. Unfortunately there are no references whether this values fits for EEG measurements or not. [15] [16]

4.1.2 Methods

The 10×10-fold cross-validation computations were performed for a specific observation time (4.5 to 5.5 seconds after starting the trial), but for different amounts of trees. For the visualization of the achieved misclassification error (MCE) improvements through more than 500 trees, the MCE results of one subject were related to the MCE results of 500 trees of this subject with:

$$MCE_{Diff} = MCE_{act} - MCE_{500} \quad . \qquad (6)$$

Where "act" means the actual amount of trees. The results of the ten subject were averaged for each amount of trees. See section 9.1 in the appendix for coding details.

4.1.3 Results

Figure 4.1 shows the mean differences in MCE over all ten subjects between using 500 trees and different amounts of trees. Negative values indicate that 500 trees showed a worse performance.

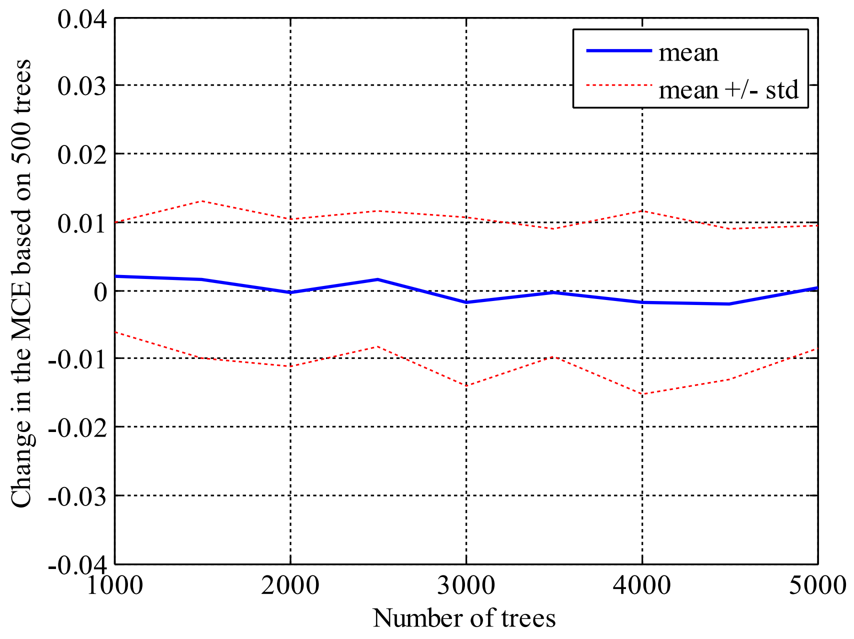

Figure 4.1: Differences between MCEs using 500 trees and different amounts of trees, respectively.

4.1.4 Discussion

As shown in Figure 4.1, using more than 500 trees has on average no influence on the MCE. The MCE is not decreasing with more trees. A statistically relevant amount of trees is already reached with 500 trees. A bigger quantity is not necessary, which is why 500 trees are used for all further calculations.

4.2 Tree correlation factor

4.2.1 Scope

A RF classifier consists of decorrelated decision trees. Two mechanisms ensure that the trees are not correlated, bootstrap sets and random features at each node of each tree for the splitting, respectively. However, it is not the best idea to create completely uncorrelated trees, as the feature at each splitting point of each tree must be chosen randomly. If the trainings observations contains many features, the probability to randomly pick a relevant feature, is very small. This leads to many trees with a low accuracy. On the other hand, if more features are chosen at random and the best one is used, the trees are not uncorrelated (parts of the trees are the same). This results in lower accuracy again, because the finding of the results is performed in the same way. The quantity of randomly chosen features is called the correlation factor. Finding the best correlation factor is not easy. The only known way is to try all different correlation factors. This is computationally very time consuming. To overcome this effort, a standard value is used. The standard value of the correlation factor for classification is $\sqrt{Number\ of\ features}$. Again there are no references if this value is suitable for EEG measurements. [15] [16]

4.2.2 Methods

For all ten subjects 10×10-fold cross-validations were calculated for each possible correlation factor (1 to 120, because of 120 features). The observation time was the same as in Section 4.1.2. Out of averaging reasons, the MCE of a subject were based on the minimum MCE of this subject with:

$$MCE_{Diff} = MCE_{act} - MCE_{min} \quad . \qquad (7)$$

MCE_{min} denotes the minimum value in this specific data set, MCE_{act} describes the current MCE for the different correlation factors. The outcomes were averaged over the subjects. See Section 9.1 in the appendix for coding details.

4.2.3 Results

Figure 4.2 shows the mean difference over all ten subjects between MCEs using different tree correlation factors. The global minimum of this curve is, on average, the most accurate tree correlation factor.

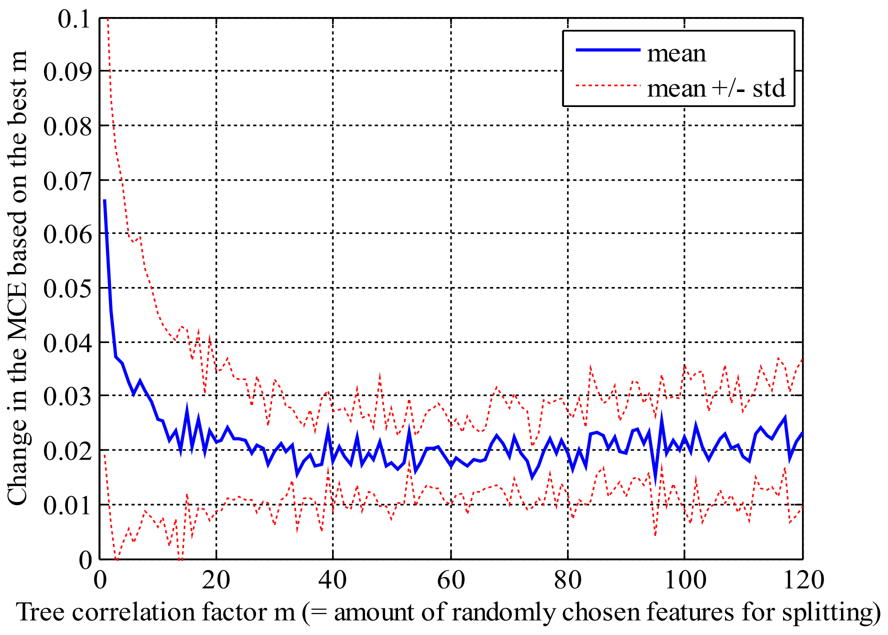

Figure 4.2: Differences between MCEs using different tree correlation factors.

4.2.4 Discussion

The fact that small values of the correlation factor lead to worse results is shown in Figure 4.2. The best results are achieved for values of the correlation factor of around 60. The standard value would be $\sqrt{120} = 10{,}9544\ldots \approx 11$, however this is not in the field of the best values. As the difference between the best values and the standard value is very small (on average smaller than 1%), it is therefore possible to use the standard value. Hence the standard value, $\sqrt{Number\, of\ \ features}$ for the correlation factor, is used for further calculations.

4.3 Tree leaf size

4.3.1 Scope

Decision trees are the basis of RF classifiers. In the normal proceeding for classification, each decision tree is fully grown. This means that the trees are grown until there is only one training observation in every leaf. For regression this standard proceeding is different. There, each tree is grown until every leaf contains 3 trainings observations. In case of using a single tree for classification, it is not a good idea to grow the tree fully. This leads to overfitting. For better results of a single tree, the tree is pruned back. In this case, each leaf contains more than one trainings observation. Thus, a better generalization can be achieved. It could be a good idea to check whether RF classifiers generalize better, if there is more than one training observation in each leaf of each tree. [15] [16]

4.3.2 Methods

Again, a specific observation time was used for the 10×10-fold cross-validation to obtain the MCE (same as in Section 4.1.2). The difference was the amount of samples in each leaf (1 to 50). This amount is called the tree leaf size. For each subject the lowest MCE was subtracted from all other MCE of the subject by using (7). Averaging was done over the results of the ten subjects. See Section 9.1 in the appendix for coding details.

4.3.3 Results

Figure 4.3 shows the mean difference in MCEs over all ten participants between using different tree leaf sizes. The global minimum of this curve represents, on average, the most accurate tree leaf size.

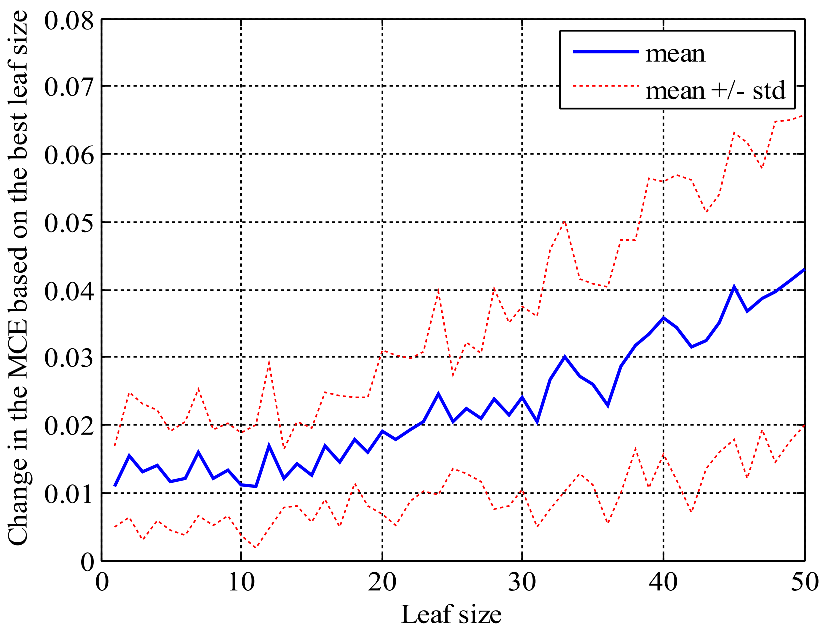

Figure 4.3: Differences between MCEs using different tree leaf sizes.

4.3.4 Discussion

Figure 4.3 shows that a tree leaf size above ten is having an influence on the classification accuracy, the MCE is increasing. Smaller leaf sizes below twenty, lead to better results, but there is little change between one and twenty (the change is smaller than 1%). Because of that, the standard value, one, is used further on. This result was expected, because RFs are using the method of "bagging". This means, many simple classifiers are voting for a class and the result of the voting is much more accurate than the result of a single simple classifier. A stand-alone classification accuracy of slightly higher than 50% is sufficient to get accurate results, if the number of simple classifiers is big enough. 500 trees are appropriate for a tree leaf size between one and about twenty. If the accuracies of the trees are getting worse, a higher amount of trees would be necessary for compensation. [15]

4.4 Logarithmic power of frequencies vs power of frequencies

4.4.1 Scope

Many classifiers require a special distribution of the observations. Linear Discriminant Analysis (LDA), for example calls for a normal distribution [15]. In case of using LDA for the classification of EEG measurements of MIs, the observations are not normally distributed. To obtain a normal like distribution, the observations are normally used logarithmically (Chi-squared distribution with high degrees of freedoms). According to the literature, no such special distribution is needed for RF classifiers, but for drawing comparisons it is necessary to know if the identical observations basis can be used. [15] [16]

4.4.2 Methods

10×10-fold cross-validation was used for calculating the MCE for every half of a second over the lapse of the trials of all subjects. The observation time was overlapping in time from one observation to the next. The computation was carried out twice, first on the power of frequencies observations, then on the logarithmic power of frequencies observations. The differences in the MCE between them were calculated for each subject using:

$$MCE_{Diff} = MCE_{raw} - MCE_{\log bp} \qquad (8)$$

The results were averaged over the subjects. See Section 9.1 in the appendix for coding details.

4.4.3 Results

Figure 4.4 shows the mean difference over all ten participants between MCEs using power of frequencies or logarithmic power of frequencies as features for classification. If values are negative, the power of frequencies produces lower MCEs than the logarithmic power of frequencies.

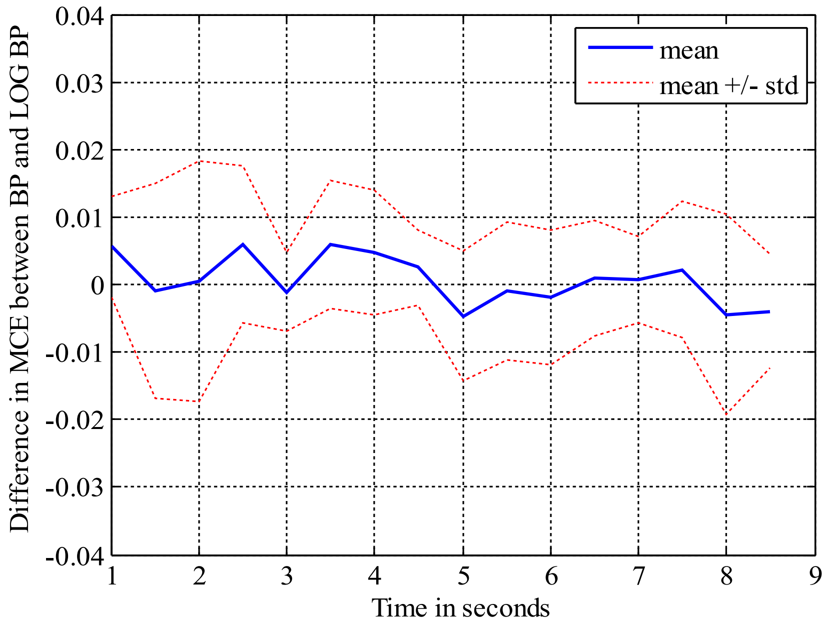

Figure 4.4: Difference between MCEs using power of frequencies or logarithmic power of frequencies.

4.4.4 Discussion

Figure 4.4 shows that there is no difference whether power of frequencies or logarithmic power of frequencies are used as features for classification. The differences are below 1% on average (smaller than 0.01). This result was expected as, according to literature, a RF classifier does not depend on a special distribution of the observations. There is no need for a Gaussian like distribution of the observations, but to have an equal basis for comparison, the logarithmic power of frequencies was used for all calculations. [16] [15]

4.5 Noisy observations

4.5.1 Scope

Another means of generating higher accuracies is the use of noisy observations. The method uses an observation more than one time, once in a raw, the other times in a noisy form. Thus it becomes very easy to generate a huge amount of “new” observations. The assumption is, that more observations lead to a higher accuracy.

4.5.2 Methods

Like before, a specific observation period was used for the 10×10-fold cross-validation to calculate the MCE (same period as in Section 4.1.2). Copies were made of each observation. One copy of each observation was not changed, to all others white Gaussian noise was added. Hence noise with a Gaussian distribution was used, as this distribution is the most common. Two parameters were varied: The number of noisy observations and the amplitude of the noise. To each observation an individual noise with the same statistical properties was added. The distribution was Gaussian, the mean value was zero and the standard deviation was the maximum value in this observation, multiplied with the percent of noise, divided by 100. Or expressed mathematically:

$$added\ noise = observation_{act} + \frac{observation_{max} \cdot percent\ of\ noise}{100} \cdot N(0,1) \qquad (9)$$

Where $N(0,1)$ is the normal distribution. The goal was to check, whether the average MCE is becoming lower with more observations or not. See Section 9.1 in the appendix for coding details.

4.5.3 Results

Figure 4.5 shows the mean differences over all ten subjects between using no noisy observations and different amounts of noisy observations and different amplitudes of noise. Positive values mean that the unmodified observations were performing better than noisy observations.

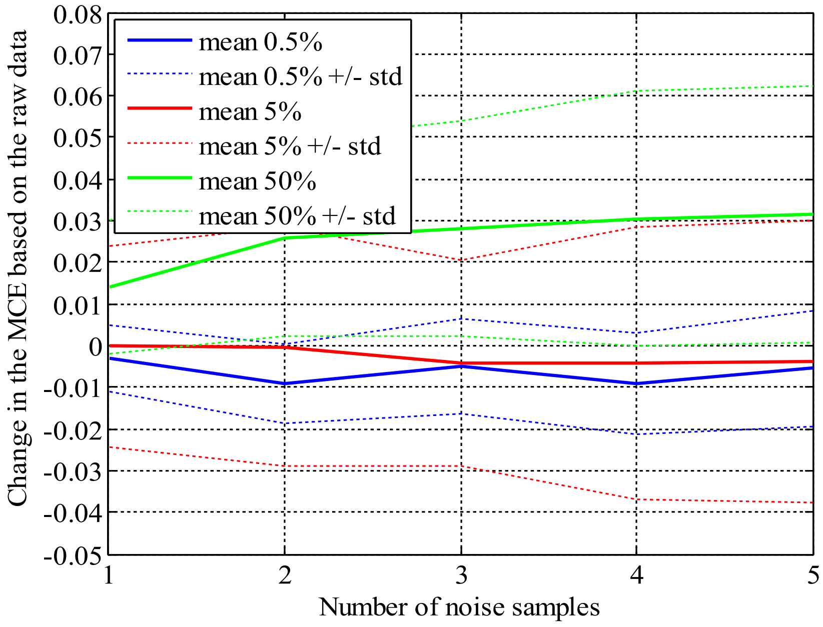

Figure 4.5: Differences between MCEs using unmodified observations and different amounts of noisy observations, respectively.

4.5.4 Discussion

Neither changing the amplitude of noise nor adding more noisy observations lead to better classification results (see Figure 4.5). The RF classifier recognizes the important features without using noisy observations. Because of that, noisy observations are not used for further calculations.

4.6 Summary

The results suggest that the standard values of the parameters of a RF classifier fit the problem of classifying EEG measurements best. Due to the difficulty of multidimensional parameter optimizations, in this work only individual parameters were tuned. Therefore, there is the possibility that combinations of parameters can result in higher accuracies. This could be a topic for further research.

5 Cross-validation tests and offline BCI operation simulations

After finding proper parameters in Section 4 of the thesis, extensive tests on the distinguishability between different classes, the offline BCI operation simulations and the behavior of the RF classifier on different amounts of trainings observations were performed on offline data.

5.1 Cross-validation tests

5.1.1 Scope

This Section will answer the questions about the distinguishability and about the optimal observation time as well as the questions about the OOB error and the importance of the features.

5.1.2 Methods

10×10-fold cross-validation tests were computed for different class combinations and different observation times. Due to the time difference of half a second between two observation times, the observations times are overlapping (segments [t-1 t] with t = 1:0.5:9). For the observation time between second 4.5 and 5.5 the OOB errors were calculated as well as the importance of the features. See Section 9.1 in the appendix for coding details.

5.1.3 Results

5.1.3.1 3 Classes cross-validations

Figure 5.1 shows the 10×10-fold cross-validation MCEs of all ten subjects for the three classes case.

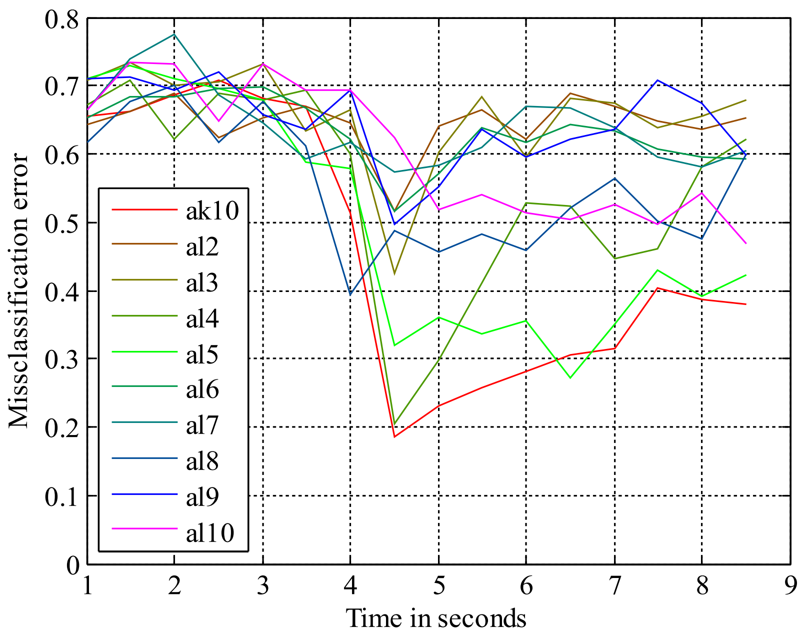

Figure 5.1: MCE over time for three classes (left hand vs right hand vs feet).

Table 5.1 shows the peak, mean and standard deviation values of the accuracies, corresponding to the cross-validation results shown in Figure 5.1.

3 classes	Subjects											
L vs R vs F	ak10	al2	al3	al4	al5	al6	al7	al8	al9	al10	mean	std
peak during imagery period	0,82	0,48	0,57	0,80	0,73	0,48	0,43	0,54	0,50	0,53	0,59	0,14
mean over imagery period	0,70	0,36	0,37	0,55	0,64	0,40	0,39	0,49	0,39	0,47	0,48	0,12
standard deviation over imagery period	0,07	0,05	0,08	0,13	0,05	0,04	0,04	0,05	0,06	0,04	0,06	0,03

Table 5.1: Peak, mean and standard deviation of the accuracies of three classes cross-validation tests.

5.1.3.2 2 Classes cross-validations

Figure 5.2 shows the 10×10-fold cross-validation MCEs of all ten subjects for the two classes case of left hand vs right hand.

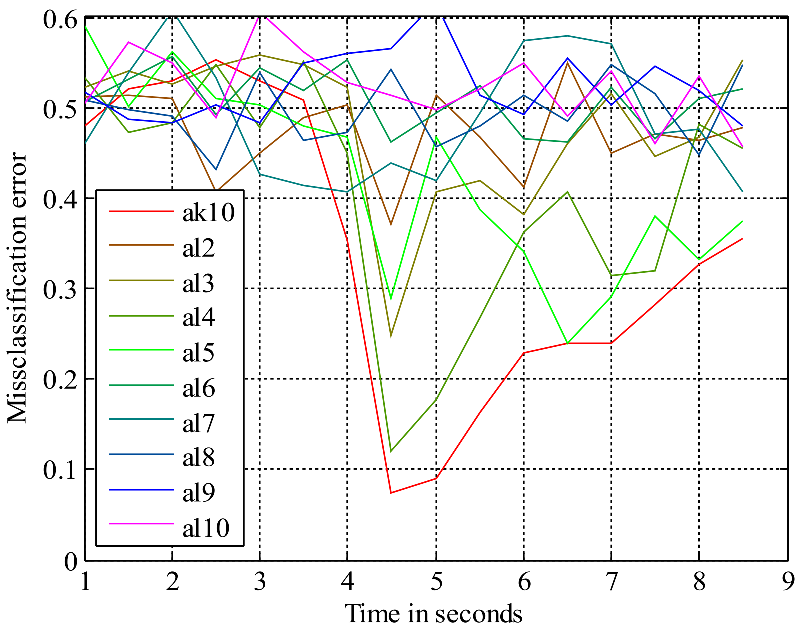

Figure 5.2: MCE over time for two classes (left hand vs right hand).

Table 5.2 shows the peak, mean and standard deviation values of the accuracies, corresponding to the cross-validation results shown in Figure 5.2.

2 classes L vs R	Subjects											
	ak10	al2	al3	al4	al5	al6	al7	al8	al9	al10	mean	std
peak during imagery period	0,93	0,63	0,75	0,88	0,76	0,54	0,59	0,55	0,52	0,54	0,67	0,15
mean over imagery period	0,78	0,54	0,57	0,68	0,66	0,51	0,51	0,50	0,47	0,49	0,57	0,10
standard deviation over imagery period	0,10	0,05	0,09	0,12	0,07	0,03	0,07	0,04	0,04	0,03	0,06	0,03

Table 5.2: Peak, mean and standard deviation of the accuracies of two classes cross-validation tests.

Figure 5.3 shows the 10×10-fold cross-validation MCEs of all ten subjects for the two classes case of left hand vs feet.

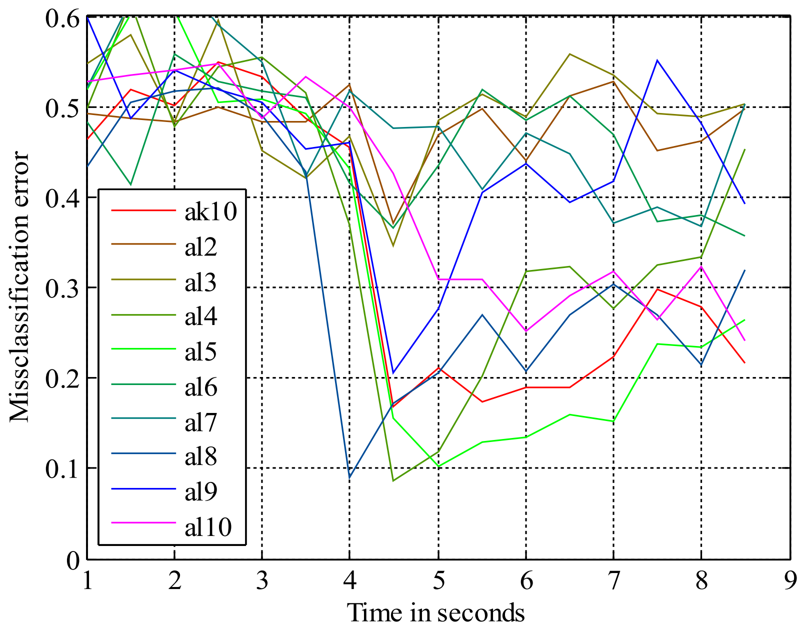

Figure 5.3: MCE over time for two classes (left hand vs feet).

Table 5.3 shows the peak, mean and standard deviation values of the accuracies, corresponding to the cross-validation results shown in Figure 5.3.

2 classes	Subjects											
L vs F	ak10	al2	al3	al4	al5	al6	al7	al8	al9	al10	mean	std
peak during imagery period	0,83	0,63	0,65	0,91	0,90	0,64	0,63	0,83	0,79	0,76	0,76	0,11
mean over imagery period	0,78	0,53	0,51	0,73	0,83	0,57	0,57	0,75	0,60	0,70	0,66	0,11
standard deviation over imagery period	0,05	0,05	0,06	0,12	0,06	0,07	0,05	0,05	0,10	0,05	0,06	0,02

Table 5.3: Peak, mean and standard deviation of the accuracies of two classes cross-validation tests.

Figure 5.4 shows the 10×10-fold cross-validation MCEs of all ten subjects for the two classes case of right hand vs feet.

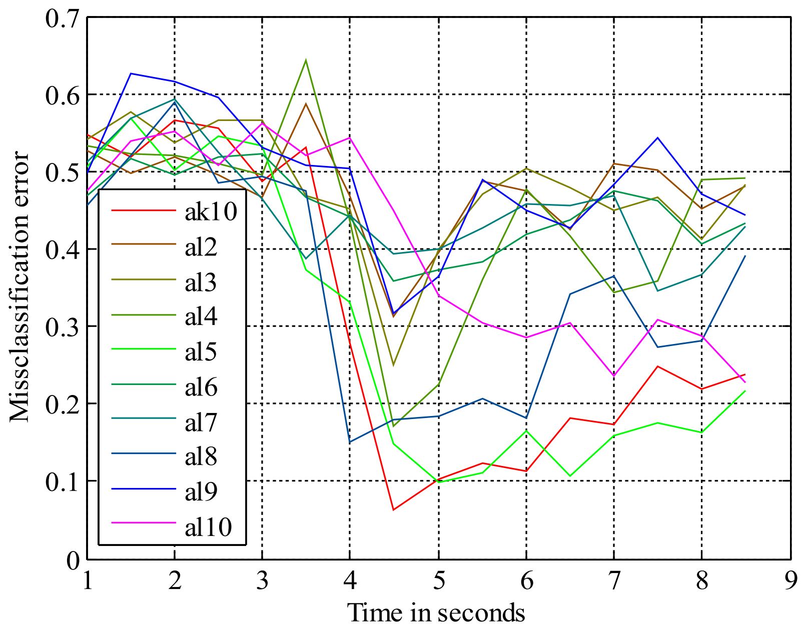

Figure 5.4: MCE over time for two classes (right hand vs feet).

Table 5.4 shows the peak, mean and standard deviation values of the accuracies, corresponding to the cross-validation results shown in Figure 5.4.

2 classes	Subjects											
R vs F	ak10	al2	al3	al4	al5	al6	al7	al8	al9	al10	mean	std
peak during imagery period	0,94	0,69	0,75	0,83	0,90	0,64	0,66	0,82	0,68	0,77	0,77	0,10
mean over imagery period	0,84	0,55	0,57	0,63	0,85	0,58	0,58	0,73	0,56	0,70	0,66	0,11
standard deviation over imagery period	0,07	0,06	0,08	0,11	0,04	0,04	0,04	0,08	0,07	0,07	0,07	0,02

Table 5.4: Peak, mean and standard deviation of the accuracies of two classes cross-validation tests.

5.1.3.3 3 Classes OOB errors

Figure 5.5 shows the OOB error of all ten subjects over the number of trees for the three classes problem of left hand vs right hand vs feet.

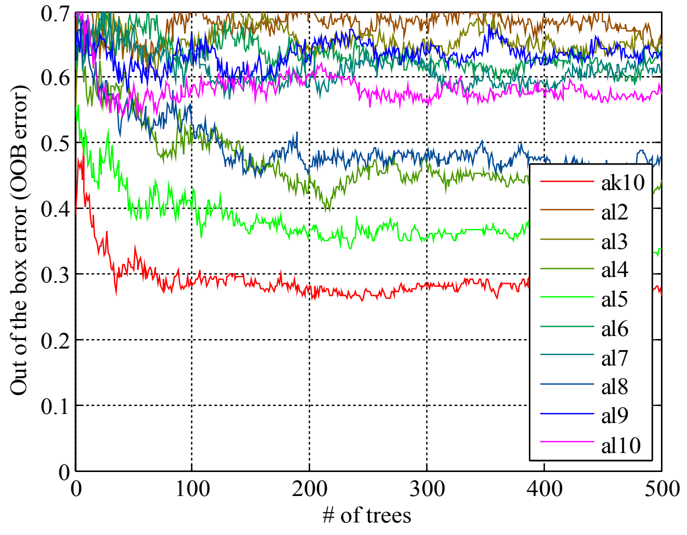

Figure 5.5: OOB error over number of trees, for three classes (left hand vs right hand vs feet).

5.1.3.4 2 Classes OOB errors

Figure 5.6 shows the OOB error of all ten subjects over the number of trees for the two classes problem of left hand vs right hand.

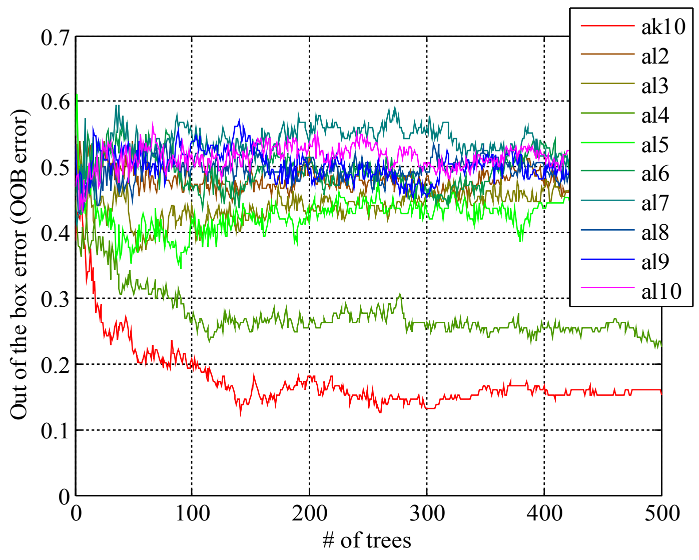

Figure 5.6: OOB error over number of trees, for two classes (left hand vs right hand).

Figure 5.7 shows the OOB error of all ten subjects over the number of trees for the two classes problem of left hand vs feet.

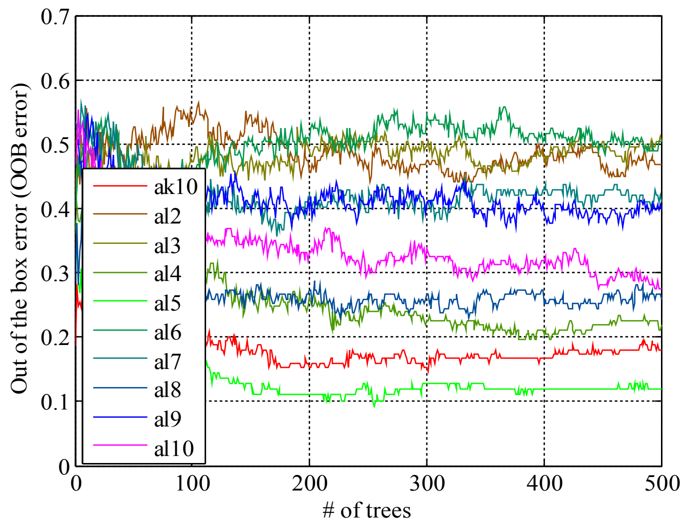

Figure 5.7: OOB error over number of trees, for two classes (left hand vs feet).

Figure 5.8 shows the OOB error of all ten subjects over the number of trees for the two classes problem of right hand vs feet.

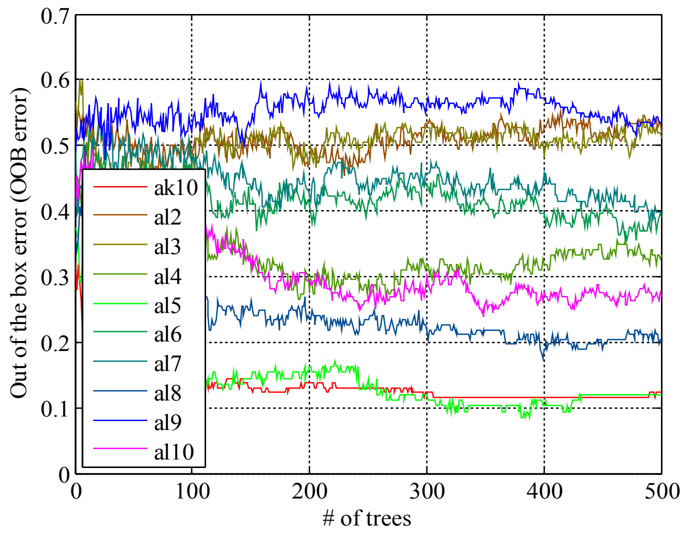

Figure 5.8: OOB error over number of trees, for two classes (right hand vs feet).

5.1.3.5 Importance of the features

Figure 5.9 shows the mean importance of the features over all ten subjects in the three classes problem of left hand vs right hand vs feet. The magnitude is an indicator of the importance of the features.

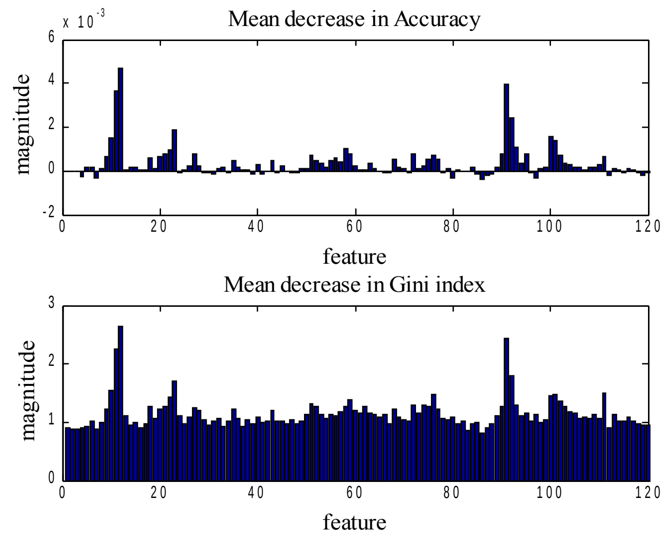

Figure 5.9: Importance of the features. Feature sectioning: 1 to 40 log power of frequencies of C3, 41 to 80 log power of frequencies of Cz, 81 to 120 log power of frequencies of C4. Three classes (left hand vs right hand vs feet).

Figure 5.10 shows the mean importance of the features over all ten subjects in the two classes problem of left hand vs right hand.

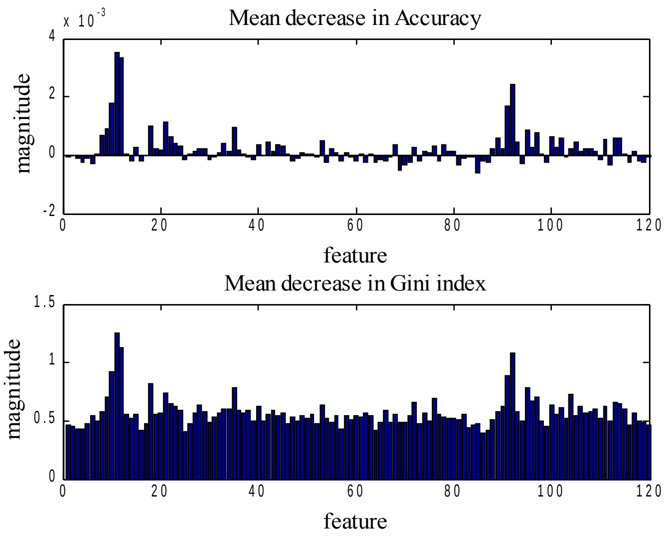

Figure 5.10: Importance of the features. Feature sectioning: 1 to 40 log power of frequencies of C3, 41 to 80 log power of frequencies of Cz, 81 to 120 log power of frequencies of C4. Two classes (left hand vs right hand).

Figure 5.11 shows the mean importance of the features over all ten subjects in the two classes problem of left hand vs feet.

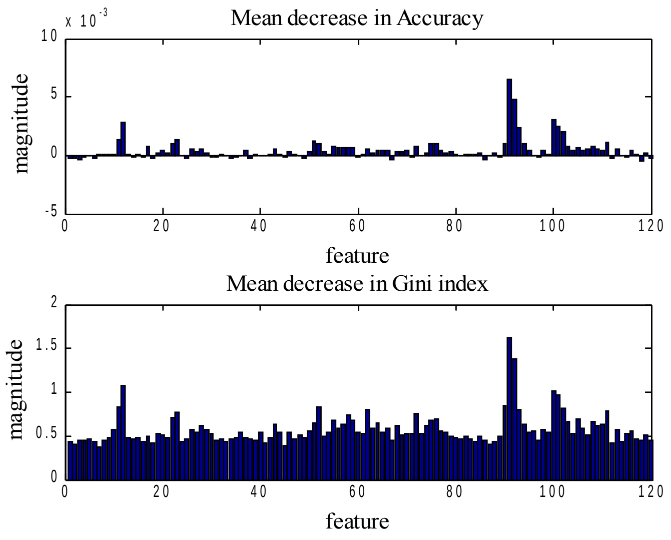

Figure 5.11: Importance of the features. Feature sectioning: 1 to 40 log power of frequencies of C3, 41 to 80 log power of frequencies of Cz, 81 to 120 log power of frequencies of C4. Two classes (left hand vs feet).

Figure 5.12 shows the mean importance of the features over all ten subjects in the two classes problem of right hand vs feet.

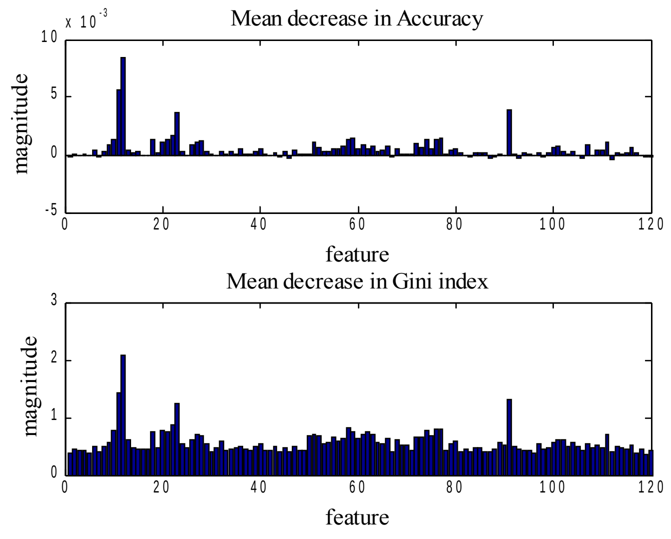

Figure 5.12: Importance of the features. Feature sectioning: 1 to 40 log power of frequencies of C3, 41 to 80 log power of frequencies of Cz, 81 to 120 log power of frequencies of C4. Two classes (right hand vs feet).

Figure 5.13 shows the importance of the features of subject ak10 for demonstrating the importance of Cz for some of the subjects in the two class problem of right hand vs feet.

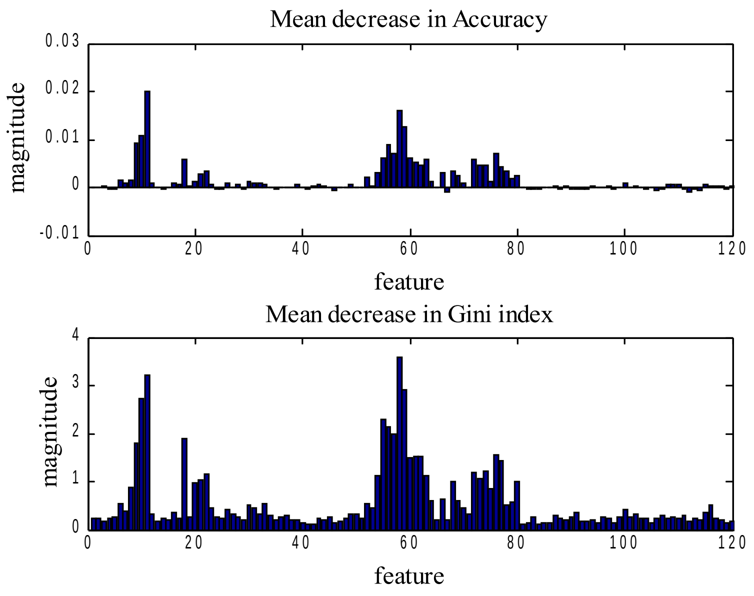

Figure 5.13: Importance of the features. Feature sectioning: 1 to 40 log power of frequencies of C3, 41 to 80 log power of frequencies of Cz, 81 to 120 log power of frequencies of C4. Two classes (right hand vs feet).

5.1.4 Discussion

In the three classes problem (see Figure 5.1 and Table 5.1), only three subjects were able to reach high accuracies. The mean accuracies during the feedback period of these three subjects (ak10, al4 and al5) were higher than 55%, this was, according to [22], better than chance ($\alpha = 1\%$). The subject with the highest accuracy (ak10) achieved about 82% in peak. There are two more subjects who were potentially better than chance (al8 and al10), but due to the limited amount of samples, the mean accuracies of these subjects, with about 45%, were lower than the chance level of 50% ($\alpha = 1\%$) [22].

Comparing Figures 5.2, 5.3, 5.4 and Tables 5.2, 5.3, 5.4 shows that the accuracies of two class problems are much higher. Peak accuracies over 80% are not a rarity, on the contrary, accuracies up to 94% are reachable. Another conclusion is that more subjects are able to distinguish two classes than three classes (up to five, with 70% mean accuracy as boundary for distinction, according to [22]). Summing up, distinguishability is better in two class problems and with the data at hand, the highest accuracies are achieved between right hand and feet motor imagery.

best	Subjects											
best	ak10	al2	al3	al4	al5	al6	al7	al8	al9	al10	mean	std
peak during imagery period	0,94	0,69	0,75	0,91	0,90	0,64	0,66	0,83	0,79	0,77	0,79	0,11
mean over imagery period	0,84	0,55	0,57	0,73	0,85	0,58	0,58	0,75	0,60	0,70	0,68	0,11
standard deviation over imagery period	0,07	0,06	0,08	0,12	0,04	0,04	0,04	0,05	0,10	0,07	0,07	0,03
classes	RvsF	RvsF	RvsF	LvsF	RvsF	RvsF	RvsF	LvsF	LvsF	RvsF		

Table 5.5: Best RF cross-validation results.

Table 5.5 shows the best cross-validation results achieved through the subjects. This results in a mean peak accuracy of 79%.

Accuracies	Subjects											
Method	Ak10 R vs F	Al2 R vs F	Al3 R vs F	Al4 L vs F	Al5 L vs F	Al6 R vs F	Al7 R vs F	Al8 R vs F	Al9 R vs F	Al10 L vs F	mean	std
Peak DSLVQ	94,00	72,00	70,00	94,00	92,00	78,00	84,00	88,00	82,00	84,00	83,80	8,56
Peak RF	94,00	69,00	75,00	91,00	90,00	64,00	66,00	82,00	68,00	76,00	77,50	11,14
Peak LDA	72,80	51,88	76,88	90,00	X	70,36	80,63	X	91,56	88,90	77,88	13,21

Table 5.6: Cross-validation accuracies of Distinction Sensitive Learning Vector Quantization (DSLVQ) vs RF (vs LDA online feedback accuracies with DSLVQ found setup) [14].

Table 5.6 shows the comparison of Distinction Sensitive Learning Vector Quantization (DSLVQ) with RF. The results of DSLVQ are from [14]. The resulting mean of peak accuracy of RF is about 5% less than the accuracy of DSLVQ. This is mainly caused by three subjects (al6, al7 and al9). It is not clear why these accuracies reached with RF were as low, compared with the accuracies reached with DSLVQ.

Comparing Figures 5.1, 5.2, 5.3 and 5.4 shows that the observation period from second 3.5 to 4.5 holds the promise for the highest classification accuracies. However, this observation period cannot be used, because it is not clear, if the classification would be done on some brain processes triggered by the cue, or really triggered by imagery itself [23]. A closer look on the cross-validation results shows that the classification is getting worse the more time passes after the cue has disappeared. The first data without influence of the cue should be the best choice. As the cue ends 4.25 seconds after starting the trial, the EEG measurements recorded later on should be free of influence of the cue. Hence, the focus is on EEG measurements recorded from second 4.5 to 5.5.

The corresponding OOB error plots (Figures 5.5, 5.6, 5.7 and 5.8) reflect the results of the cross-validation tests. Good results in the cross-validation tests are having equally good OOB error results. The ranking of accuracies provided by the cross-validation is the same as shown in the corresponding OOB error plots. The absolute values are a relatively good estimation with a difference of about 5% to the results from the cross-validation. Therefore, the results suggests that the OOB error is a reliable estimation of the expected error.

Figures 5.9, 5.10, 5.11 and 5.12 presents the affiliated importance of the feature plots. These figures show that the most relevant features are the powers of frequencies of C3 and C4 at about 10 Hz (alpha rhythm) and slightly over 20 Hz (beta rhythm). Such an outcome was expected [11]. As Figures 5.9, 5.10, 5.11 and 5.12 suggest, the powers of frequencies of Cz seem not to be that important. But the features importance are an average of ten subjects. Thus, there are subjects, who are benefiting from features of Cz (see Figure 5.13).

5.2 Offline BCI simulations

The results in Section 5.1 suggests that RFs are suitable for the classification of MI in EEG measurements. However the suitability for a real setting is not yet clear. Therefore, in a first step, an offline simulation of a BCI was performed.

5.2.1 Scope

The question is, if the RF classifier is able to classify MIs out of completely unseen runs of EEG measurements. In case of cross-validation, the observations used for the training of the classifiers were picked from all runs. In this Section coherent runs were used for the training and following runs for the testing. This division is closer to real BCI behavior than cross-validations.

5.2.2 Methods

The trainings of the RF classifiers were based on the observation periods from second 4.5 to 5.5 of the trials of the first five runs of a subject. The remaining 3 runs of the subject were used to perform a BCI simulation. In the simulation process the obtained RF classifier was used to execute a classification every tenth of a second during each trial. Due to the length of an observation (one second) and the repetition every tenth of a second, the observations used overlapping measurements. The strict partitioning of training and testing is like in a real BCI. This simulation was performed for different combinations of classes and in each combination for all ten subjects. The corresponding OOB errors and importance of the features were calculated too. Coding details can be found in the appendix, Section 9.1.

5.2.3 Results

5.2.3.1 3 Classes BCI simulations

Figure 5.14 shows the MCEs of all ten subjects in the three classes problem of left hand vs right hand vs feet.

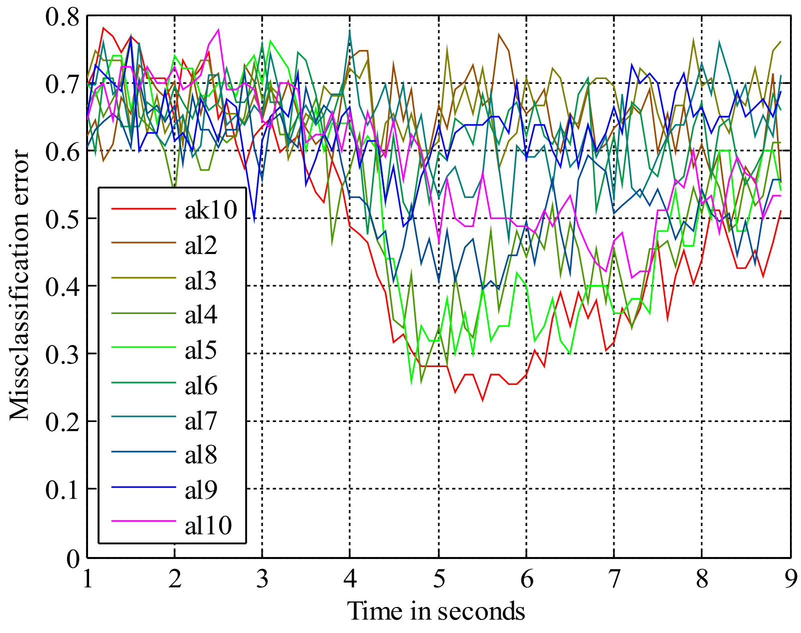

Figure 5.14: MCEs over time. Three classes (left hand vs. right hand vs. feet).

Table 5.7 shows the peak, mean and standard deviation values of the accuracies, corresponding to the BCI simulation results shown in Figure 5.14.

3 classes	Subjects											
L vs R vs F	ak10	al2	al3	al4	al5	al6	al7	al8	al9	al10	mean	std
peak during feedback period	0,77	0,52	0,43	0,71	0,70	0,49	0,53	0,60	0,43	0,59	0,58	0,12
mean during feedback period	0,64	0,36	0,32	0,55	0,58	0,38	0,39	0,51	0,35	0,50	0,46	0,11
std during feedback period	0,08	0,06	0,04	0,08	0,09	0,05	0,06	0,05	0,04	0,05	0,06	0,02

Table 5.7: Peak, mean and standard deviation of the accuracies of three classes BCI simulation.

5.2.3.2 2 Class BCI simulations

Figure 5.15 shows the MCEs of all ten subjects in the two class problem of left hand vs right hand.

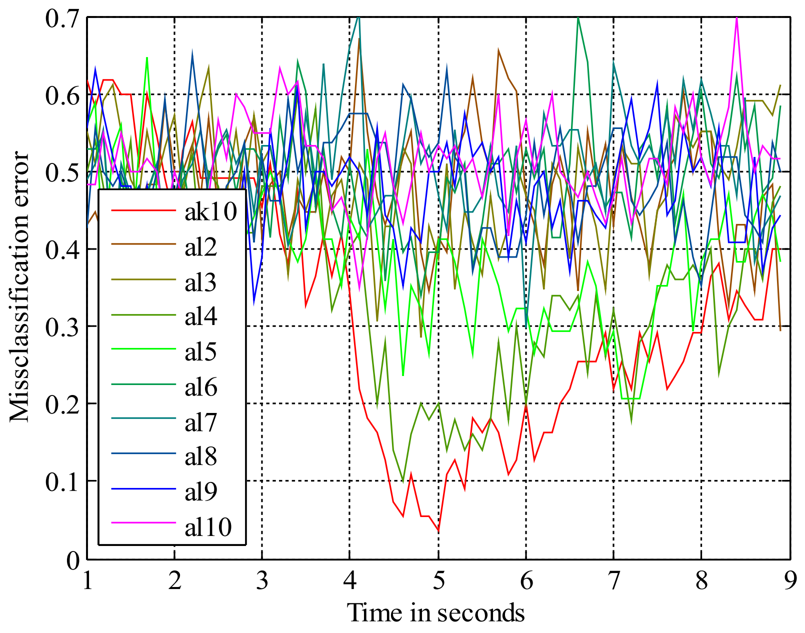

Figure 5.15: MCEs over time. Two classes (left hand vs. right hand).

Table 5.8 shows the peak, mean and standard deviation values of the accuracies, corresponding to the BCI simulation results shown in Figure 5.15.

2 classes	Subjects											
L vs R	ak10	al2	al3	al4	al5	al6	al7	al8	al9	al10	mean	std
peak during feedback period	0,96	0,71	0,65	0,86	0,79	0,62	0,70	0,65	0,63	0,58	0,72	0,12
mean during feedback period	0,77	0,52	0,52	0,71	0,65	0,50	0,49	0,53	0,52	0,48	0,57	0,10
std during feedback period	0,09	0,09	0,07	0,09	0,07	0,07	0,07	0,06	0,06	0,05	0,07	0,01

Table 5.8: Peak, mean and standard deviation of the accuracies of two classes BCI simulation.

Figure 5.16 shows the MCEs of all ten subjects in the two class problem of left hand vs feet.

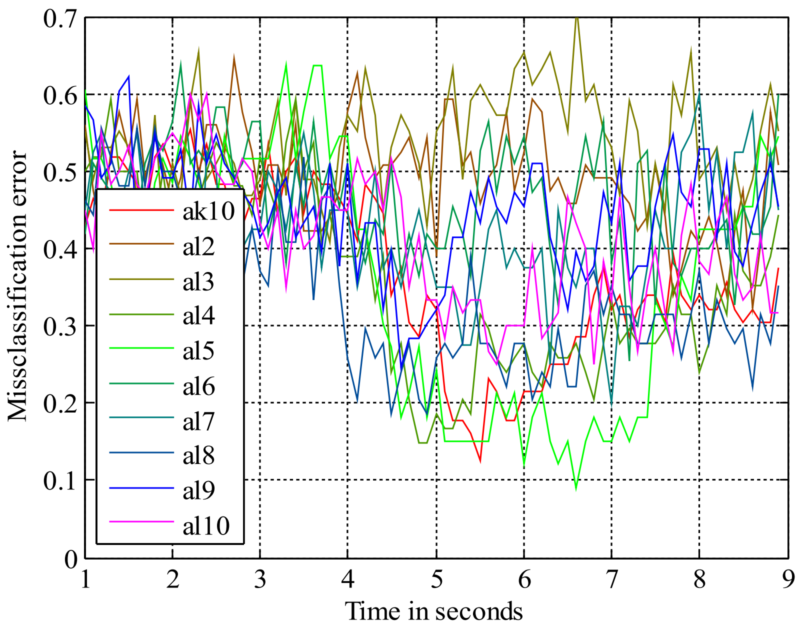

Figure 5.16: MCEs over time. Two classes (left hand vs. feet).

Table 5.9 shows the peak, mean and standard deviation values of the accuracies, corresponding to the BCI simulation results shown in Figure 5.16.

2 classes	Subjects											
L vs F	ak10	al2	al3	al4	al5	al6	al7	al8	al9	al10	mean	std
peak during feedback period	0,88	0,68	0,57	0,85	0,91	0,75	0,80	0,81	0,70	0,75	0,77	0,10
mean during feedback period	0,72	0,53	0,44	0,72	0,74	0,56	0,60	0,71	0,56	0,65	0,62	0,10
std during feedback period	0,07	0,07	0,07	0,07	0,13	0,08	0,09	0,05	0,06	0,06	0,07	0,02

Table 5.9: Peak, mean and standard deviation of the accuracies of two class BCI simulation.

Figure 5.17 shows the MCEs of all ten subjects in the two class problem of right hand vs feet.

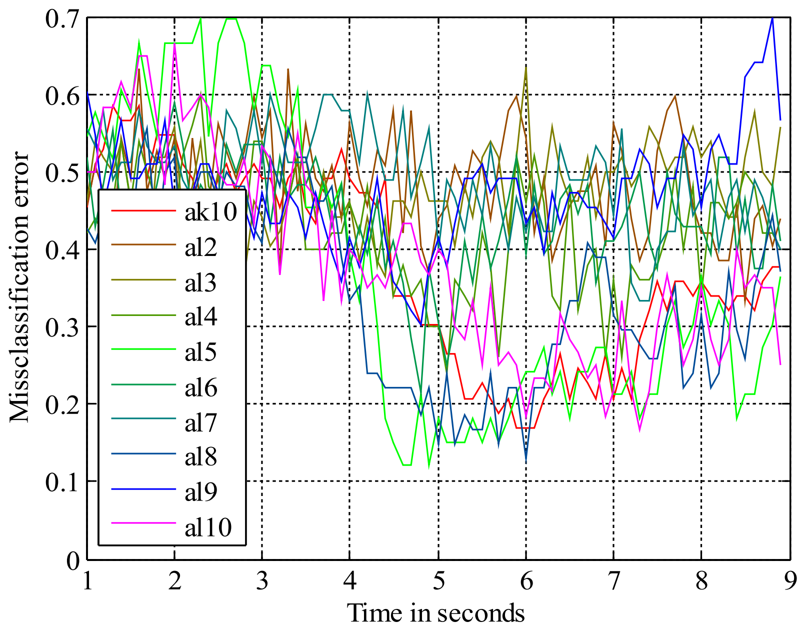

Figure 5.17: MCEs over time. Two classes (right hand vs. feet).

Table 5.10 shows the peak, mean and standard deviation values of the accuracies, corresponding to the BCI simulation results shown in Figure 5.17.

2 classes	Subjects											
R vs F	ak10	al2	al3	al4	al5	al6	al7	al8	al9	al10	mean	std
peak during feedback period	0,83	0,67	0,60	0,76	0,88	0,75	0,67	0,87	0,62	0,83	0,75	0,10
mean during feedback period	0,73	0,53	0,50	0,61	0,77	0,58	0,54	0,73	0,51	0,71	0,62	0,10
std during feedback period	0,07	0,07	0,05	0,08	0,06	0,06	0,06	0,08	0,07	0,06	0,06	0,01

Table 5.10: Peak, mean and standard deviation of the accuracies of two class BCI simulation.

5.2.3.3 3 Classes OOB errors

Figure 5.18 shows the OOB errors of all ten subjects over the number of trees for the three class problem of left hand vs right hand vs feet.

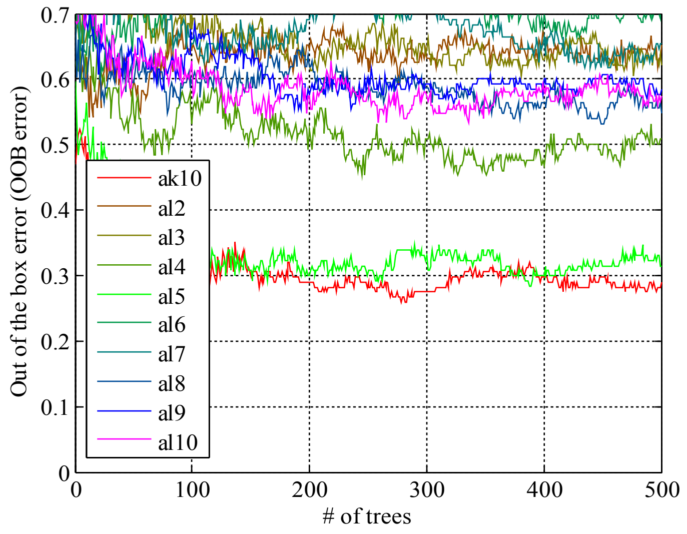

Figure 5.18: OOB error over number of trees, for three classes (left hand vs. right hand vs. feet).

5.2.3.4 2 Classes OOB errors

Figure 5.19 shows the OOB errors of all ten subjects over the number of trees for the two class problem of left hand vs right hand.

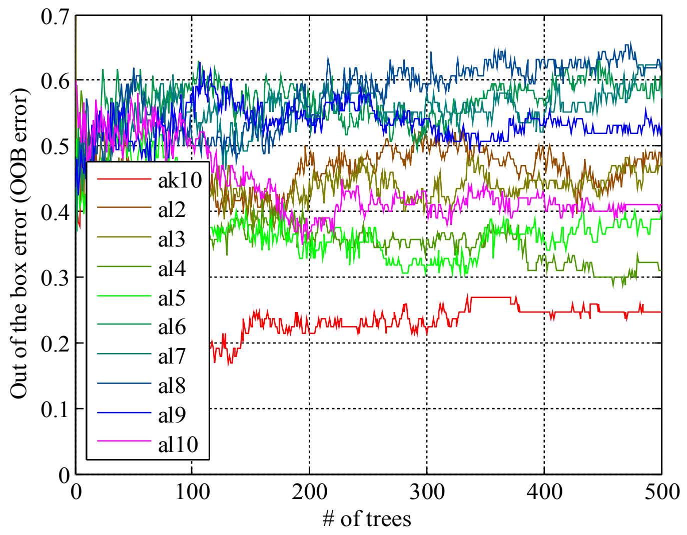

Figure 5.19: OOB error over number of trees, for two classes (left hand vs. right hand).

Figure 5.20 shows the OOB errors of all ten subjects over the number of trees for the two class problem of left hand vs feet.

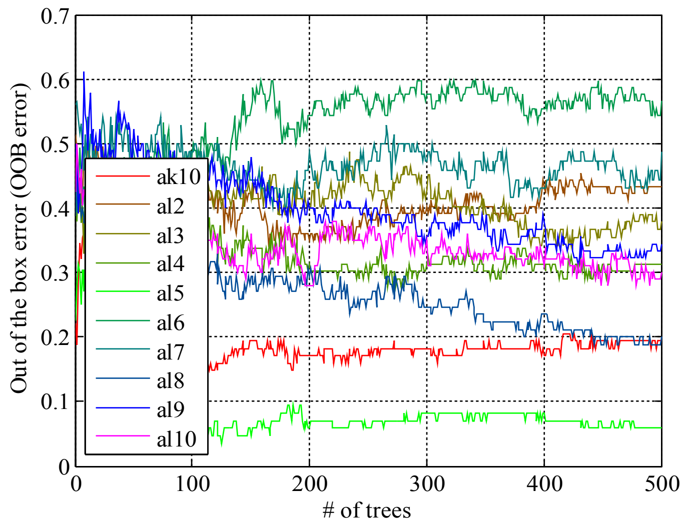

Figure 5.20: OOB error over number of trees, for two classes (left hand vs. feet).

Figure 5.21 shows the OOB errors of all ten subjects over the number of trees for the two class problem of right hand vs feet.

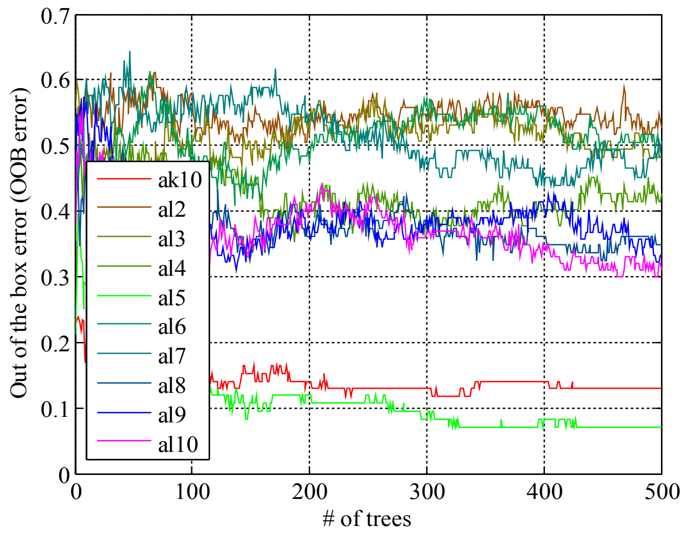

Figure 5.21: OOB error over number of trees, for two classes (right hand vs. feet).

5.2.3.5 Importance of the features

Figure 5.22 shows the mean importance of the features of all ten subjects in the three class problem of left hand vs right hand vs feet.

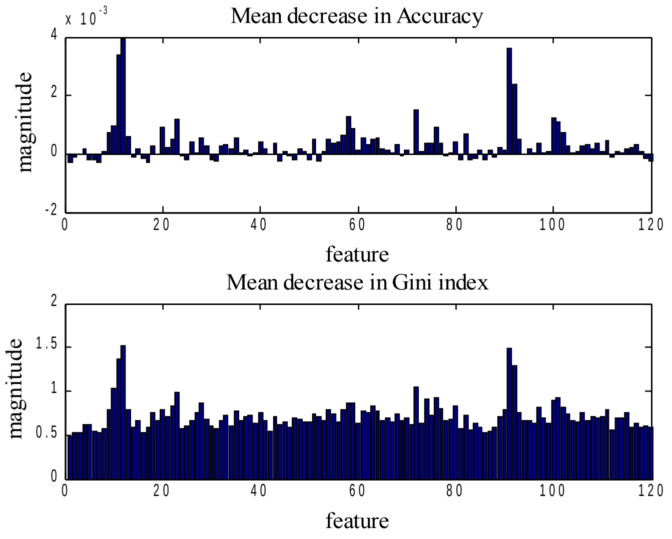

Figure 5.22: Importance of the features. Feature sectioning: 1 to 40 log power of frequencies of C3, 41 to 80 log power of frequencies of Cz, 81 to 120 log power of frequencies of C4. Three classes (left hand vs right hand vs feet).

Figure 5.23 shows the mean importance of the features of all ten subjects in the two class problem of left hand vs right hand.

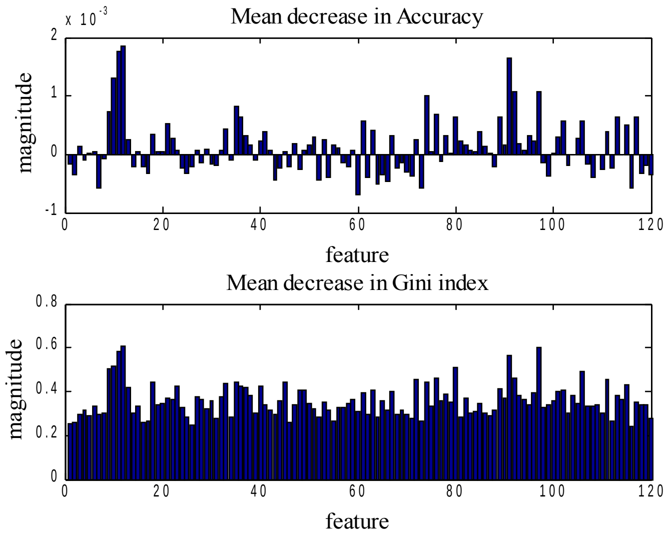

Figure 5.23: Importance of the features. Feature sectioning: 1 to 40 log power of frequencies of C3, 41 to 80 log power of frequencies of Cz, 81 to 120 log power of frequencies of C4. Two classes (left hand vs right hand).

Figure 5.24 shows the mean importance of the features of all ten subjects in the two class problem of left hand vs feet.

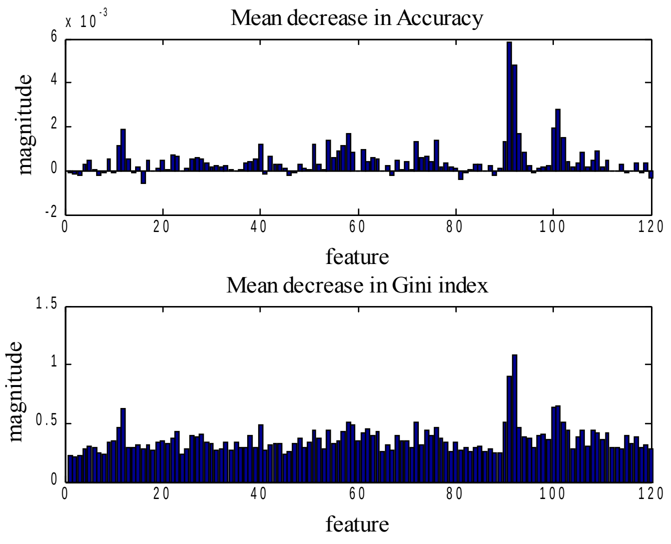

Figure 5.24: Importance of the features. Feature sectioning: 1 to 40 log power of frequencies of C3, 41 to 80 log power of frequencies of Cz, 81 to 120 log power of frequencies of C4. Two classes (left hand vs feet).

Figure 5.25 shows the mean importance of the features of all ten subjects in the two class problem of right hand vs feet.

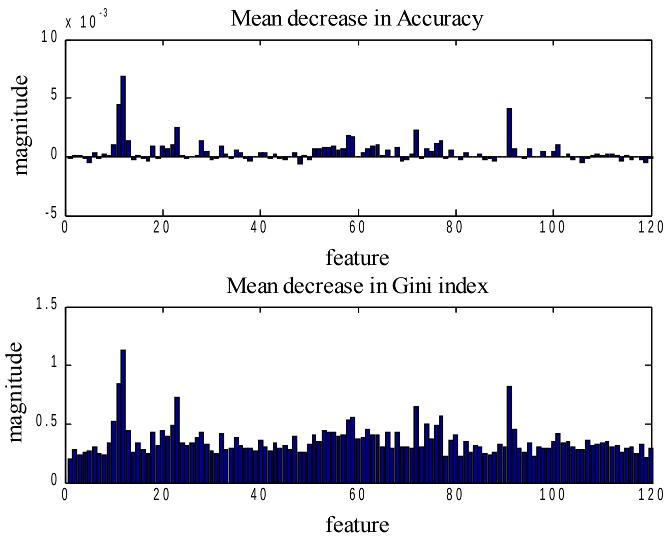

Figure 5.25: Importance of the features. Feature sectioning: 1 to 40 log power of frequencies of C3, 41 to 80 log power of frequencies of Cz, 81 to 120 log power of frequencies of C4. Two classes (right hand vs feet).

5.2.4 Discussion

Figures 5.14, 5.15, 5.16 and 5.17 show that the classifier is able to detect the correct classes on unseen EEG measurements, but not for all subjects. Comparing the results with those from section

5.1.3 shows that the same subjects are good performers. The accuracies are a little worse. That is not surprising, because in the case of cross-validation, the training was done with 144 observations (2 class case) and in the case of BCI simulation with 100 observations only. Less observations lead to less accuracy [15]. This behavior will be investigated more detail in the next Section.

Once again, like in the cross-validation Section, the OOB error gives a good estimation of the prospective accuracies (see Figures 5.18, 5.19, 5.20 and 5.21).

The importance plots (see Figures 5.22, 5.23, 5.24 and 5.25) show, again, that the important frequencies were in the alpha and beta band.

5.3 Behavior on different amounts of trainings observations

After successful results in classification of unseen EEG measurements in Section 5.2, the unanswered question about the amount of trainings observations is addressed on this Section.

5.3.1 Scope

In the context of machine learning it is clear that more trainings observations lead to higher accuracies, assuming separable classes [15]. But the consistency of the trainings observations, a precondition for the previous statement, cannot be guaranteed when using bio-signals, because of their non-stationarity. Therefore the behavior of the RF classifier on different amounts of trainings observations was investigated.

5.3.2 Methods

Offline BCI simulations were performed with different amounts of trainings observations. Due to the fact that not all subjects were able to operate a BCI, only the observations of previously found good performer (ak10, al4 and al5) were used for simulation. See Section 9.1 in the appendix for coding details.

5.3.3 Results

Figure 5.26 shows the MCEs of subject ak10 in the three class problem of left hand vs right hand vs feet.

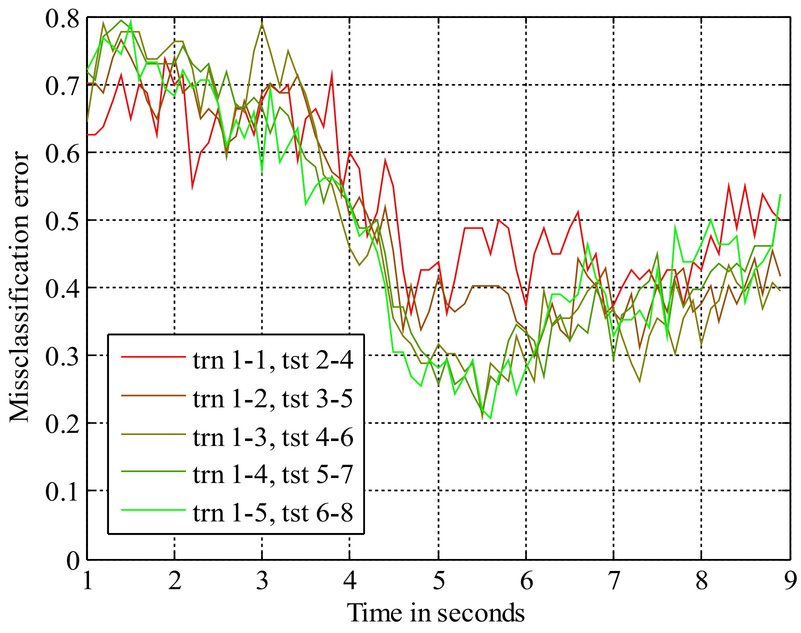

Figure 5.26: MCE over time for data set ak10.

Table 5.11 shows the peak, mean and standard deviation values of the accuracies, corresponding to the BCI simulation results shown in Figure 5.26.

3 classes ak10	Observation for training						
L vs R vs F	30	60	90	120	150	mean	std
peak during feedback period	0,64	0,70	0,79	0,78	0,79	0,74	0,07
mean during feedback period	0,55	0,62	0,66	0,64	0,63	0,62	0,04
std during feedback period	0,05	0,04	0,05	0,07	0,09	0,06	0,02

Table 5.11: Peak, mean und standard deviation of the accuracies for different amounts of trainings observations.

Figure 5.27 shows the MCEs of subject al4 in the three class problem of left hand vs right hand vs feet.

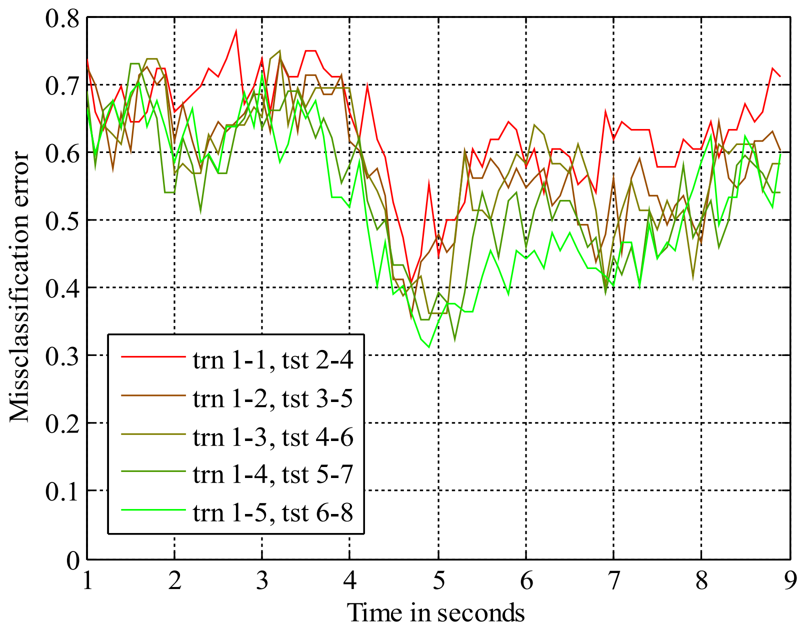

Figure 5.27: MCE over time for data set al4.

Table 5.12 shows the peak, mean and standard deviation values of the accuracies, corresponding to the BCI simulation results shown in Figure 5.27.

3 classes al4	Observation for training						
L vs R vs F	30	60	90	120	150	mean	std
peak during feedback period	0,55	0,56	0,64	0,68	0,69	0,62	0,06
mean during feedback period	0,40	0,46	0,47	0,52	0,54	0,47	0,05
std during feedback period	0,05	0,05	0,07	0,07	0,08	0,06	0,01

Table 5.12: Peak, mean und standard deviation of the accuracies for different amounts of trainings observations.

Figure 5.28 shows the MCEs of subject al5 in the three class problem of left hand vs right hand vs feet.

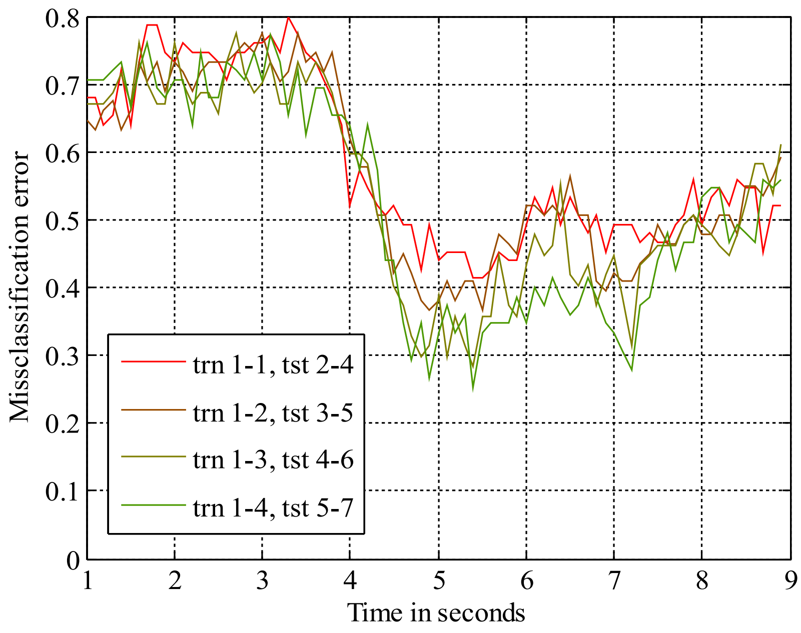

Figure 5.28: MCE over time for data set al5.

Table 5.13 shows the peak, mean and standard deviation values of the accuracies, corresponding to the BCI simulation results shown in Figure 5.28.

3 classes al5	Observation for training					
L vs R vs F	30	60	90	120	mean	std
peak during feedback period	0,59	0,63	0,72	0,75	0,67	0,07
mean during feedback period	0,51	0,53	0,56	0,59	0,55	0,04
std during feedback period	0,04	0,06	0,08	0,08	0,07	0,02

Table 5.13: Peak, mean und standard deviation of the accuracies for different amounts of trainings observations.

5.3.4 Discussion

The results in Figures 5.26, 5.27, 5.28 and Tables 5.11, 5.12, 5.13 show that previously determined good performers benefited from using more training observations. The accuracies improved to 15%, 14% and 16%, respectively, between using 30 and 120 trainings observations (Table 5.11, 5.12 and 5.13). But saturation effects were showing up. Table 5.11 shows that the peak accuracy was rising by 6% from using 30 to 60 trainings observations, but was only rising by 1% from using 120 to 150 trainings observations. This saturation effects are showing up for each tested subject.

5.4 Summary

The results suggests that RF classifiers are able to detect the correct MIs in offline cross-validation tests and in offline BCI simulations. More training observations lead to higher classification accuracies (with upcoming saturation effects). The provided OOB errors are a good estimation of the accuracies. The provided importance estimations are showing which features are crucial for classification. Moreover they conform with literature [11].

6 Online BCI using Random Forests

After successful demonstration of the ability of RF to classify MIs in cross-validation and offline BCI simulations, the last Section of this work covers the online detection abilities of RF classifiers.

6.1 Scope

Although the offline abilities of RF classifiers were demonstrated in Section 5, the online abilities cannot be derived from these results. But they can be considered as a precondition. A BCI with feedback after the training was set up for demonstrating the classification of MIs in online applications.

6.2 Methods

The trainings observations period was fixed at second 4.5 to 5.5. All the observations obtained in runs one to five were used for training. Feedback was given to the subject in the following three runs. Therefore, a classification was performed every tenth of a second. If more than five of the last ten classifications were accurate the feedback bar grew. Due to the rapid sequence of classification, the observation periods were overlapping.

The BCI system was created in MATLAB Simulink, using the Tobi SignalServer [24] and the Graz-BCI libraries. Details on the Simulink model can be found in the appendix, Section 9.2.

6.2.1 Addition on experimental paradigm

In the case of the paradigm used for the EEG measurements in this Section, which were especially recorded for this work, the design was slightly different.

Three able bodied subjects, all male, aged between 26 and 28, one without experience in using a BCI, were seated in a chair in front of an LCD screen, with a distance of one meter, on which randomized instructions for different MIs were showing up. A single trial was built up as follows (see also Figure 1.2):

- Second 0: A green cross faded up.
- Second 3: One of two cues was displayed for 1.25 seconds. The two cues were: Right pointing arrow for right hand MI and down pointing arrow for feet MI.
- Second 4: The subject was hold on to start the MI after the cue was displayed and stop the MI if the green cross disappeared. During the testing phase (runs six to eight) feedback was provided with a horizontal bar. The bar grew, if the results of the classification were correct.
- Second 13: End of MI phase. The fixation cross disappeared and a break with a variable length (between two and three seconds) followed.

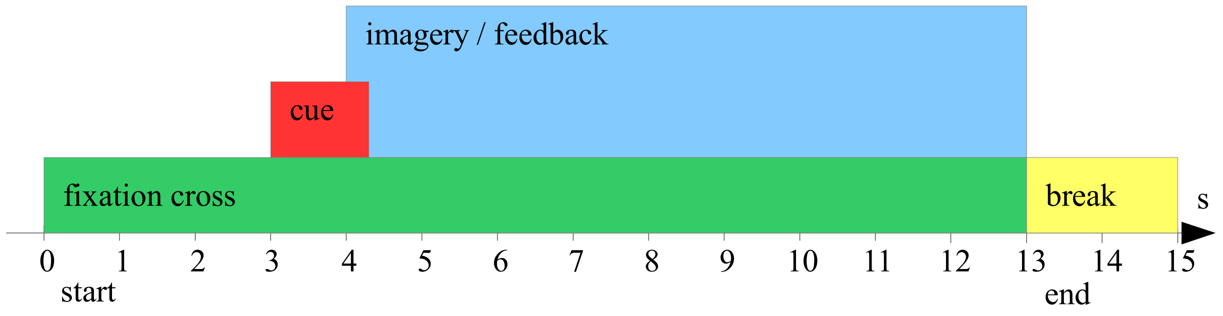

Figure 6.1: Recording paradigm for EEG measurements used in Section 6.

A run consisted of 20 trials, ten for each class, a session consisted of 8 runs. Therefore, all in all 160 trials were recorded. Each session was divided into two phases. The first phase (runs one to five) was used to record EEG data during the MI. In the second phase (runs six to eight), the classifiers results were used to provide feedback on the classification accuracy.

6.2.2 Addition on EEG measurement

In the case of the EEG measurements used in this Section the sensorimotor area over C3, Cz and C4 was covered with 15 Ag/AgCl electrodes, the reference electrode was mounted on the left mastoid and the ground electrode was mounted on the right mastoid. The mounting followed the international 10-20-system (see Figure 6.2). For the recording of the EEG, a bio-signal USB amp build by the company g.tec (g.USBamp, g.tec Guger Technologies, Graz, Austria) was used with the following settings: Sampling rate at 512 Hz, a bandpass filter (chebyshev filter of 8th order and cut off frequencies at 0.1 and 200 Hz) and a notch filter at 50 Hz.

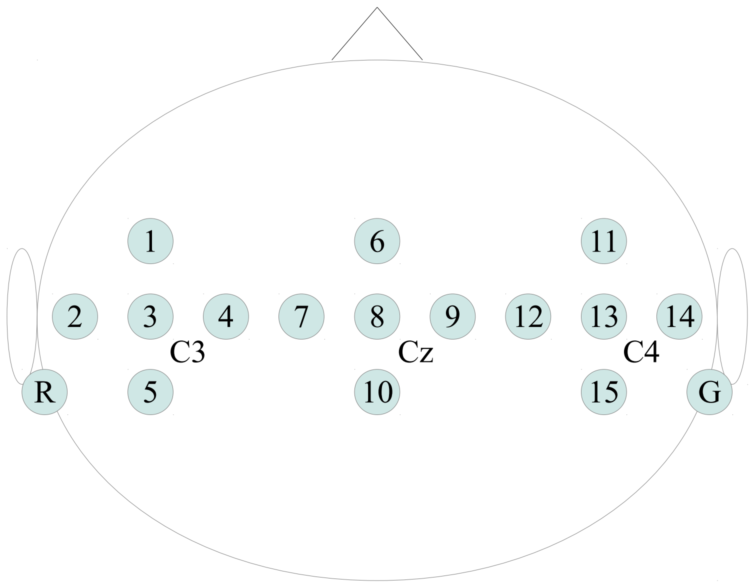

Figure 6.2: Mounting of the electrodes according to the 10-20-system.

6.2.3 Additional signal preprocessing

In this Section the EEG measurements were processed online. Therefore there was no possibility to perform a visual screening for artifacts in online data and there was no artifact rejection in this Section.

6.2.4 Addition on software & tools

In addition, for constructing the BCI Simulink model in this Section, the TOBI SignalServer (http://bci.tugraz.at/downloads.html) developed under the TOBI project (http://www.tobi-project.org) and the Graz-BCI libraries, primarily for giving feedback, were used.

6.3 Results

Figure 6.3 shows the MCEs over the test trials of the three subjects.

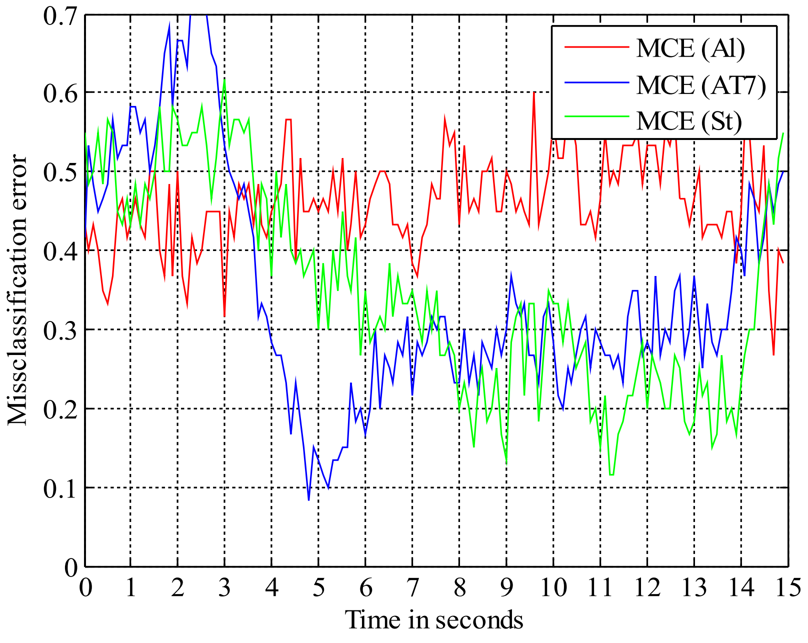

Figure 6.3: Online MCE over all test trials.

Table 6.1 shows the peak, mean and standard deviation values of the accuracies, corresponding to the BCI results shown in Figure 6.3.

2 classes	Subjects				
R vs F	Al	AT7	ST	mean	std
peak during feedback period	0,73	0,92	0,88	0,84	0,10
mean during feedback period	0,53	0,72	0,71	0,65	0,11
std during feedback period	0,06	0,08	0,09	0,08	0,02

Table 6.1:Peak, mean and standard deviation of the accuracies of two class BCI.

Figure 6.4 shows the OOB errors of the three subjects over the number of trees for the two class problem of right hand vs feet.

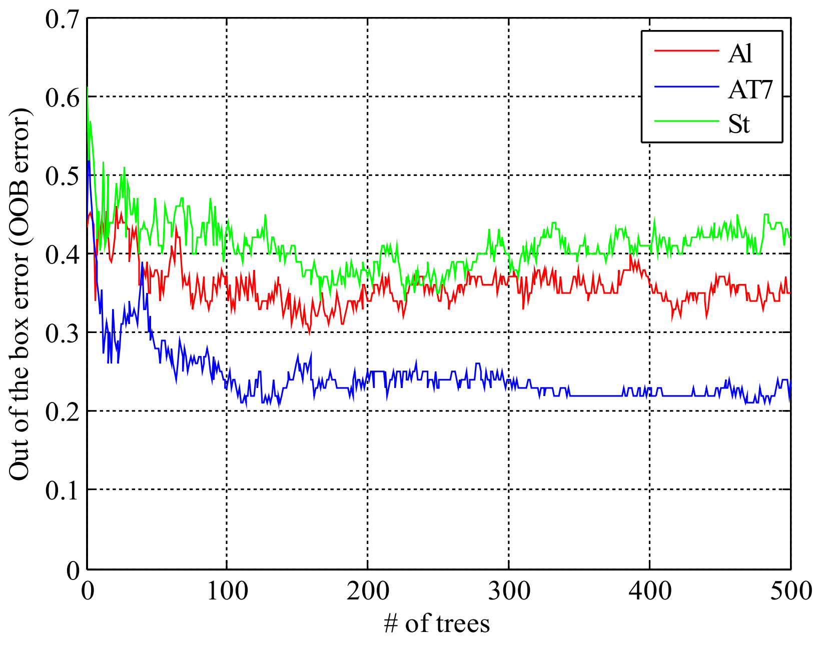

Figure 6.4: OOB error of the used RFclassifiers in online application.

Figure 6.5 shows the importance of the features of subject Al in the two class problem of right hand vs feet.

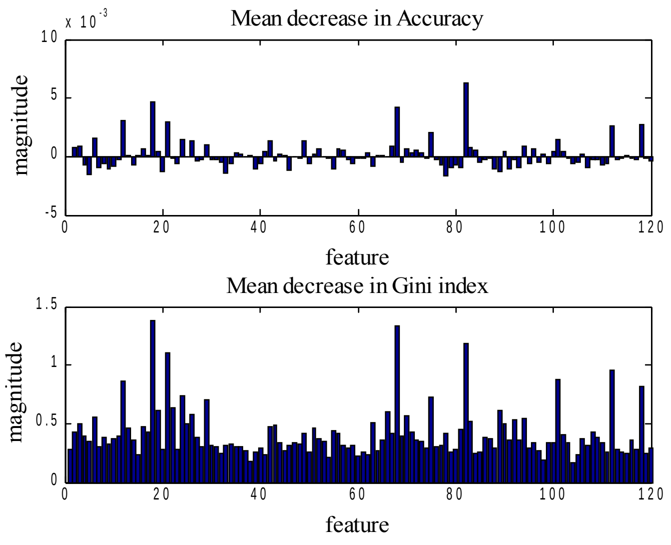

Figure 6.5: Importance of the features. Feature sectioning: 1 to 40 log power of frequencies of C3, 41 to 80 log power of frequencies of Cz, 81 to 120 log power of frequencies of C4. Two classes (right hand vs feet).

Figure 6.6 shows the importance of the features of subject AT7 in the two classes problem of right hand vs feet.

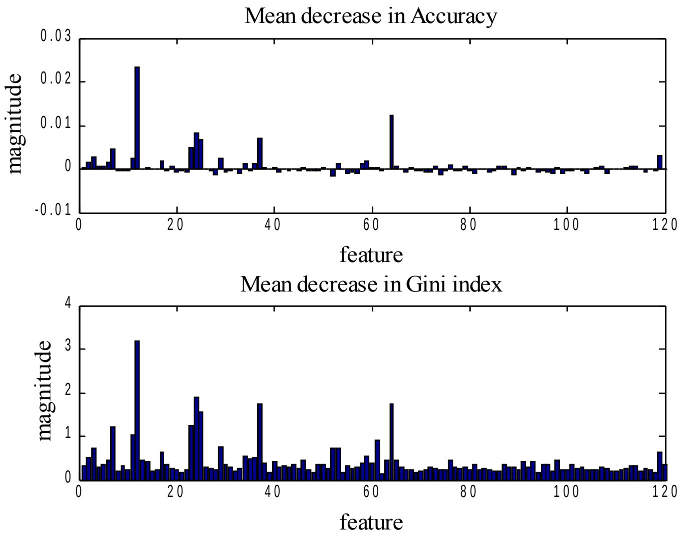

Figure 6.6: Importance of the features. Feature sectioning: 1 to 40 log power of frequencies of C3, 41 to 80 log power of frequencies of Cz, 81 to 120 log power of frequencies of C4. Two classes (right hand vs feet).

Figure 6.7 shows the importance of the features of subject ST in the two class problem of right hand vs feet.

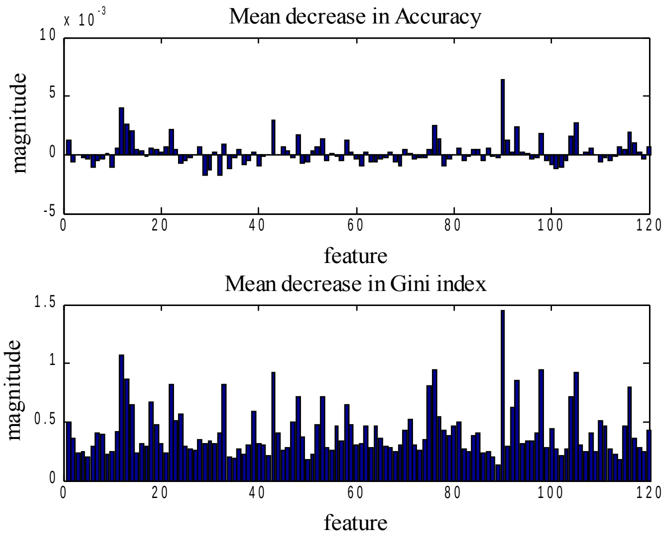

Figure 6.7: Importance of the features. Feature sectioning: 1 to 40 log power of frequencies of C3, 41 to 80 log power of frequencies of Cz, 81 to 120 log power of frequencies of C4. Two classes (right hand vs feet).

6.4 Discussion

Figure 6.3 and Table 6.1 show the MCE over all test trials of the three subjects. Two of the three subjects were able to operate the RF BCI. These two subjects were previously known to be good performer in BCI interaction. The peak accuracy was up to 92%. An interesting detail about subject St is, that he comes up with his peak performance relatively late in the feedback period. This is not a common result, but in other studies, he shows a similar characteristic. Subject AT7 shows a more common result. Best performance was reached shortly after the cue and degraded over time [25]. The naïve subject (Al) was not able to use the BCI. His results were around chance level.

In the case of online data, the OOB error does not seem to be a good estimation of the MCE. Figure 6.4 shows the OOB errors of the two subjects. This error estimation is not totally wrong for the observed time interval from second 4.5 to 5.5 (compare with Figure 6.3), but the behavior of the MCE of the subjects shifts completely over time. Thus, the OOB error becomes a bad estimation for the MCE later on. There are two possible reasons for this behavior: On the one hand, there was no outlier rejection. This could have a strong impact on the OOB error, because some of the observations could have been inconsistent with others. On the other hand, the possible adaptation of the brain to the behavior of the RF classifier. Further investigations would be necessary.

The results in MCEs are also reflected in the corresponding importance plots (Figure 6.5, 6.6 and 6.7). Subject Al does not show important features in the alpha band (where Mu rhythms would be expected [11]) (Figure 6.5). On the contrary, subjects AT7 and St were show important features in the alpha band and beta band of C3 and C4, respectively.

6.5 Summary

This Section of the work has demonstrated that RF classifiers are able to detect mental imagery online.

7 Conclusion and perspective

The aim of this work was to investigate the suitability of the RF classifier to detect MI in EEG. For that purpose, a multiplicity of topics were covered.

In Section 2 a brief historical overview and theoretical background on RF were presented. Section 3 addressed the used methods and tools. Following, it was shown in Section 4 that standard parameters are proper for MI classification too. An individual search for each parameter was performed. A multidimensional search was due to the complexity not performed and could bring up a different conclusion. The first part of Section 5 covered extensive cross-validation calculations and showed the principal ability of RF for the classification of MIs. In part two, an offline BCI simulation was successfully performed to show the classification on unseen runs of the data. A third part covered the behavior of the RF classifier on rising amounts of trainings observations. It turned out, that the behavior is as expected. Higher amounts of training observations lead to higher classification accuracies, but saturation effects were observed. Analysis of the saturation effects could lead to a recommended amount of trainings observations in further research. And finally in Section 6 the successful application of the RF classifier in a real BCI was demonstrated.

The additional options of the RF classifier (OOB error, importance of the features) were investigated too and the results suggest their validity for providing accurate estimations.

The applicability of the RF classifier in real BCI environments, such as asynchronous BCIs, would be an interesting research topic for the future.

8 References

[1] S. G. Mason, A. Bashashati, M. Fatourechi, K. F. Navarro, and G. E. Birch: *A Comprehensive Survey of Brain Interface Technology Designs.* Annals of Biomedical Engineering, Vol. 35, p. 137–169, 2007.

[2] J. J. Vidal: *Toward direct Brain-Computer communication.* Annual Review of Biophysics and Bioengineering, 2, 157-180, 1973.

[3] N. Birbaumer, N. Ghanayim, T. Hinterberger, I. Iversen, B. Kotchoubey, A. Kübler, J. Perelmounter, E. Taub, H. Flor: *A spelling device for the paralysed.* Nature 398, p. 197-298, 1999.

[4] F. Nijboer, E. W. Sellers, J. Mellinger, M. A. Jordan, T. Matuz, A. Furdea, S. Halder, U. Mochty, D. J. Krusienski, T. M. Vaughan, J. R. Wolpaw, N. Birbaumer, A. Kübler: *A P300-based brain-computer interface for people with amyotrophic lateral sclerosis.* Clin. Neurophysiol. 119, p. 1909-1916, 2008.

[5] A. Kübler, A. Furdea, S. Halder, E. M. Hammer, F. Nijboer, B. Kotchoubey: *A brain-computer interface controlled auditory event-related potential (p300) spelling system for locked-in patients.* Ann. NY Acad. Sci. 1157, p. 90-100, 2009.

[6] J. R. Wolpaw, N. Birnbaumer, D. J. McFarland, G. Pfurtscheller, T. M. Vaughan: *Brain-Computer interfaces for communication and control.* Clin. Neurophysiol. 113, p. 767-791, 2002.

[7] G. Pfurtscheller, G. R. Müller-Putz, R. Scherer, C. Neuper: *Rehabilitation with Brain-Computer Interface Systems.* IEEE: Computer, Volume 41, p. 48-65, 2008.

[8] R. Rao, R. Scherer: *Brain-Computer Interfacing.* IEEE: Signal Processing Magazine, July 2010, p. 147-150, 2010.

[9] H. Berger: *Über das Elektroenkephalogramm des Menschen.* Arch. Psychiat. Nervenkr, vol. 99, no. 6, p. 555–574, 1933.

[10] G. Pfurtscheller, C. Neuper: *Motor imagery and direct brain-computer communication.* Proc. IEEE 89, p. 1123-1134, 2001.

[11] G. Pfurtscheller, F. H. Lopes da Silva: *Event-related EEG/MEG synchronization and desynchronization: basic principles.* Clin. Neurophysiol. 110, p. 1842-1857, 1999.

[12] B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, K. R. Müller: *Optimizing Spatial Filters for Robust EEG Single-Trial Analysis.* IEEE Signal Processing Magazine January 2008, p. 41-56.

[13] F. Lotte, M. Congedo, A. L´ecuyer, F. Lamarche, B. Arnaldi: *A review of classification algorithms for EEG-based brain–computer interfaces.* Journal of Neural Engineering: 4, R1–R13, 2007.

[14] G. R. Müller-Putz, R. Scherer, G. Pfurtscheller, C. Neuper: *Temporal coding of brain patterns for direct limb control in humans.* Frontiers in neuroscience: June 2010, Volume 4, Article 34, 2010.

[15] T. Hastie, R. Tibshirani, J. Friedman: *The Elements of Statistical Learning*. Springer: Springer Series in Statistics, Second Edition, 2009.

[16] L. Breiman: *Random Forests*. Kluwer Academic Publishers, Machine Learning, 45, p. 5-32, 2001.

[17] L. Breiman: *Bagging Predictors*. Kluwer Academic Publishers, Machine Learning, 24, p. 123-140, 1996.

[18] T. K. Ho: *Random Decision Forests*. AT&T Bell Laboratories: 1995.

[19] T. K. Ho: *The random subspace method for constructing decision forests*. Bell Laboratories, Lucent Technologies: 1998.

[20] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone: *CART: Classification and Regression Trees*. Wadsworth: Belmont, CA, 1983.

[21] H. H. Jasper: *The ten-twenty electrode system of the International Federation*. Electroencephalography and clinical neurophysiology 10, p. 371–375, 1958.

[22] G. R. Müller-Putz, R. Scherer, C. Brunner, R. Leeb, G. Pfurtscheller: *Better than Random? A closer look on BCI results.* International Journal of Bioelektromagnetism: 2008, Volume 10, No. 1, p. 52 – 55, 2008.

[23] V. Kaiser, C. Brunner, R. Leeb, C. Neuper, G. Pfurtscheller: *Investigation of cue-based vertical and horizontal eye movements with electroencephalographic and eye-tracking data.* Clin. Neurophysiol.: 120(11), p. 1988-93, 2009.

[24] C. Breitwieser and C. Eibel: *TiA-Documentation of TOBI Interface A.* ArXiv e-prints, Mar. 2011.

[25] R. Scherer, G. Pfurtscheller, C. Neuper: *Motor imagery induced changes in oscillatory EEG components: speed vs accuracy.* Proceedings of the 4th International Brain-Computer Interface Workshop and Training Course 2008, p. 186-190, 2008.

9 Appendix

This section provides all used MATLAB scripts and MATLAB / Simulink models.

9.1 Appendix on Section 4 and 5

9.1.1 MATLAB code for offline calculations

The Matlab script offline_new.m was written to perform the preprocessing (feature extraction), the tests for finding proper parameter, the cross-validations tests, the offline BCI simulations and the tests with different amounts of samples. The functional core is shown below:

```
%% Data preprocessing %%
%%%%%%%%%%%%%%%%%%%%%%%%

% parameters
channel_c3=1; %set to 1 if you want this channel (0 else)
channel_cz=1; %set to 1 if you want this channel (0 else)
channel_c4=1; %set to 1 if you want this channel (0 else)
class_1=1; %set to 1 if you want this class (0 else)
class_2=2; %set to 2 if you want this class (0 else)
class_3=0; %set to 3 if you want this class (0 else)
frequence_selection=2:41; %vector of selected frequences (entry 1 is frequence 0)
no_noised_samples=0; %how many noised copies of the data should be made?
percent_noise=0; %percent of noise 0 bis 100% (percent of maximum value in this sample as
standard deriviation of the noise)

% define sizes of data (X) and classlabels (Y)
no_channels=channel_c3+channel_cz+channel_c4;
no_frequences=length(frequence_selection)*no_channels;
no_timepoints=LengthOfSample/time_offset*SampleRate;
no_samples=0;
for ind1=1:length(data)
    TRIG=data(ind1).h.TRIG;
    CL=data(ind1).h.Classlabel;
    if al6==1 && ind1==5 || al10==1
        ArtifactSelection=zeros(1,length(TRIG));
    else
        ArtifactSelection=data(ind1).h.ArtifactSelection;
    end
    for ind2=1:length(TRIG)
        if ArtifactSelection(ind2)==0 && (CL(ind2)==class_1 || CL(ind2)==class_2 ||
CL(ind2)==class_3)
            no_samples=no_samples+1;
        end
    end
    if ind1==1
        no_samples_1=no_samples;
    elseif ind1==2
        no_samples_2=no_samples;
    elseif ind1==3
        no_samples_3=no_samples;
    elseif ind1==4
        no_samples_4=no_samples;
    elseif ind1==5
        no_samples_5=no_samples;
    elseif ind1==6
        no_samples_6=no_samples;
    elseif ind1==7
        no_samples_7=no_samples;
    elseif ind1==8
        no_samples_8=no_samples;
    end
end
% define size of X
X=zeros(no_frequences,no_timepoints,no_samples,no_noised_samples+1);
% define size of Y
Y=zeros(1,no_timepoints,no_samples,no_noised_samples+1);

% processing
ind4=0;
for ind1=1:length(data)
    data_c3=data(ind1).s(:,1)';
    data_cz=data(ind1).s(:,2)';
    data_c4=data(ind1).s(:,3)';
    TRIG=data(ind1).h.TRIG;
    CL=data(ind1).h.Classlabel;

    if al6==1 && ind1==5 || al10==1
        ArtifactSelection=zeros(1,length(TRIG));
    else
        ArtifactSelection=data(ind1).h.ArtifactSelection;
    end

    for ind2=1:length(TRIG)
        if ArtifactSelection(ind2)==0 && (CL(ind2)==class_1 || CL(ind2)==class_2 ||
CL(ind2)==class_3)
            ind4=ind4+1;
            for ind3=0:(8/time_offset*SampleRate-1)
                features=zeros(length(frequence_selection),channel_c3+channel_cz+channel_c4);
                if channel_c3==1

data_c3_part=data_c3(TRIG(ind2)+ind3*time_offset:TRIG(ind2)+ind3*time_offset+SampleRate-1);
                    data_c3_part=fft(data_c3_part);
                    data_c3_part=data_c3_part(frequence_selection);
                    data_c3_part=log(abs(data_c3_part).^2);
                    features(:,1)=data_c3_part;
                end
                if channel_cz==1

data_cz_part=data_cz(TRIG(ind2)+ind3*time_offset:TRIG(ind2)+ind3*time_offset+SampleRate-1);
                    data_cz_part=fft(data_cz_part);
                    data_cz_part=data_cz_part(frequence_selection);
                    data_cz_part=log(abs(data_cz_part).^2);
                    if channel_c3==1
                        features(:,2)=data_cz_part;
                    elseif channel_c3==0
                        features(:,1)=data_cz_part;
                    end
                end
                if channel_c4==1

data_c4_part=data_c4(TRIG(ind2)+ind3*time_offset:TRIG(ind2)+ind3*time_offset+SampleRate-1);
                    data_c4_part=fft(data_c4_part);
                    data_c4_part=data_c4_part(frequence_selection);
                    data_c4_part=log(abs(data_c4_part).^2);
                    if channel_c3==1 && channel_cz==1
                        features(:,3)=data_c4_part;
                    elseif channel_c3==0 && channel_cz==1
```

```
                        features(:,2)=data_c4_part;
                    elseif channel_c3==0 && channel_cz==0
                        features(:,1)=data_c4_part;
                    end
                end
                features=features';
                X(:,ind3+1,ind4,1)=features(:)';
                Y(:,ind3+1,ind4,1)=CL(ind2);

                if no_noised_samples>0
                    for ind5=2:no_noised_samples+1

features=zeros(length(frequence_selection),channel_c3+channel_cz+channel_c4);
                        if channel_c3==1

data_c3_part=data_c3(TRIG(ind2)+ind3*time_offset:TRIG(ind2)+ind3*time_offset+SampleRate-1);

data_c3_part=data_c3_part+max(data_c3_part)*percent_noise/100*randn(1,length(data_c3_part));
                            data_c3_part=fft(data_c3_part);
                            data_c3_part=data_c3_part(frequence_selection);
                            data_c3_part=log(abs(data_c3_part).^2);
                            features(:,1)=data_c3_part;
                        end
                        if channel_cz==1

data_cz_part=data_cz(TRIG(ind2)+ind3*time_offset:TRIG(ind2)+ind3*time_offset+SampleRate-1);

data_cz_part=data_cz_part+max(data_cz_part)*percent_noise/100*randn(1,length(data_cz_part));
                            data_cz_part=fft(data_cz_part);
                            data_cz_part=data_cz_part(frequence_selection);
                            data_cz_part=log(abs(data_cz_part).^2);
                            if channel_c3==1
                                features(:,2)=data_cz_part;
                            elseif channel_c3==0
                                features(:,1)=data_cz_part;
                            end
                        end
                        if channel_c4==1

data_c4_part=data_c4(TRIG(ind2)+ind3*time_offset:TRIG(ind2)+ind3*time_offset+SampleRate-1);

data_c4_part=data_c4_part+max(data_c4_part)*percent_noise/100*randn(1,length(data_c4_part));
                            data_c4_part=fft(data_c4_part);
                            data_c4_part=data_c4_part(frequence_selection);
                            data_c4_part=log(abs(data_c4_part).^2);
                            if channel_c3==1 && channel_cz==1
                                features(:,3)=data_c4_part;
                            elseif channel_c3==0 && channel_cz==1
                                features(:,2)=data_c4_part;
                            elseif channel_c3==0 && channel_cz==0
                                features(:,1)=data_c4_part;
                            end
                        end
                        features=features';
                        X(:,ind3+1,ind4,ind5)=features(:)';
                        Y(:,ind3+1,ind4,ind5)=CL(ind2);
                    end
                end
            end
        end
    end
end

%% n times k fold cross validation %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% parameters
no_used_samples=1:no_samples; %number of samples for cv (no_samples for all samples,
no_samples_x for samples until trial x)
n=10; %number of reiterations
k=10; %number of folds
```

```
number_of_trees=0; %0 mean 500 trees, is standart value (you can use a vector of number of
trees for searching best number of trees)
m_try=0; %0 mean sqrt(number of data), is standart value (you can use a vector of m for
searching best m)
date=0:0.1:14.9; %date of trainings dataset in s after trigger (you can use a vector of dates
for searching the best date)
use_noised_samples=0; %eg. [0,1,2] max no_noised_samples (you can use a vector of number of
noised samples for searching best number of noised samples)
number_of_lin_comb=0; %how many linear combinations do you want? (use 0 if you don't want
linear combinations) (you can use a vector of number of linear combinations for searching best
number of linear combinations)
clear extra_options; clear model;
extra_options.importance=0; %0=(Default) Don't, 1=calculate importance
nodesize=1; %1=Default for classification, bigger nodesize leads to smaller trees (you can use
a vector of nodesizes for searching best nodesize)


% number of used samples
X_use=X(:,:,no_used_samples,:);
Y_use=Y(:,:,no_used_samples,:);

% cross validation
results=[];
for ind10=1:length(m_try)
for ind11=1:length(date)
for ind12=1:length(nodesize)
for ind13=1:length(number_of_trees)
for ind14=1:length(use_noised_samples)
for ind16=1:length(number_of_lin_comb)
    if number_of_lin_comb(ind16)>0 %linear combinations

X_lin=zeros(number_of_lin_comb(ind16),no_timepoints,length(no_used_samples),no_noised_samples+
1);
        rand1=randi(no_frequences,1,number_of_lin_comb(ind16));
        rand2=randi(no_frequences,1,number_of_lin_comb(ind16));
        for ind21=1:no_noised_samples+1
            for ind22=1:length(no_used_samples)
                for ind23=1:no_timepoints
                    for ind24=1:number_of_lin_comb(ind16)
                        X_lin(ind24,ind23,ind22,ind21)=X_use(rand1(ind24),ind23,ind22,ind21)-
X_use(rand2(ind24),ind23,ind22,ind21);
                    end
                end
            end
        end
        X_CV=X_lin;
        Y_CV=Y_use;
    else
        X_CV=X_use;
        Y_CV=Y_use;
    end
    no_errors=0; %cross validation
    no_tests=0;
    for ind17=1:n %n times
        ind=crossvalind('Kfold',no_used_samples, k);
        for ind18=1:k %k fold
            [no_features,~,~,~]=size(X_CV);
            X_CV_trn=X_CV(:,int32(date(ind11)/time_offset*SampleRate+1),ind~=ind18,1);
            X_CV_trn=reshape(X_CV_trn,no_features,sum(ind~=ind18))';
            Y_CV_trn=Y_CV(:,int32(date(ind11)/time_offset*SampleRate+1),ind~=ind18,1);
            Y_CV_trn=reshape(Y_CV_trn,1,sum(ind~=ind18))';
            X_CV_tst=X_CV(:,int32(date(ind11)/time_offset*SampleRate+1),ind==ind18,1);
            X_CV_tst=reshape(X_CV_tst,no_features,sum(ind==ind18))';
            Y_CV_tst=Y_CV(:,int32(date(ind11)/time_offset*SampleRate+1),ind==ind18,1);
            Y_CV_tst=reshape(Y_CV_tst,1,sum(ind==ind18))';
            if use_noised_samples(ind14)>0
                for ind19=1:use_noised_samples(ind14)

X_CV_trn_noise=X_CV(:,int32(date(ind11)/time_offset*SampleRate+1),ind~=ind18,ind19+1);
                    X_CV_trn_noise=reshape(X_CV_trn_noise,no_features,sum(ind~=ind18))';
                    X_CV_trn=[X_CV_trn;X_CV_trn_noise];
```

```
Y_CV_trn_noise=Y_CV(:,int32(date(ind11)/time_offset*SampleRate+1),ind~=ind18,ind19+1);
                    Y_CV_trn_noise=reshape(Y_CV_trn_noise,1,sum(ind~=ind18))';
                    Y_CV_trn=[Y_CV_trn;Y_CV_trn_noise];
                end
            end
            extra_options.nodesize=nodesize(ind12);
            model =
classRF_train(X_CV_trn,Y_CV_trn,number_of_trees(ind13),m_try(ind10),extra_options);
            Y_hat = classRF_predict(X_CV_tst,model);
            no_errors=no_errors+sum(Y_hat~=Y_CV_tst);
            no_tests=no_tests+length(Y_CV_tst);
        end
    end
    results=[results,no_errors/no_tests];
end
end
end
end
end
end


%% Test time series %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% parameters
no_used_samples_training=1:no_samples_5; %number of samples for training (no_samples for all
samples, no_samples_x for samples until trial x)
no_used_samples_testing=no_samples_5+1:no_samples_8; %number of samples for testing
(no_samples for all samples, no_samples_x for samples until trial x)
number_of_trees=0; %0 mean 500 trees, is standart value
m_try=0; %0 mean sqrt(number of data), is standart value
date=4.5; %date of trainings dataset in s after trigger
use_noised_samples=0; %eg. 0,1,2 max no_noised_samples
number_of_lin_comb=0; %how many linear combinations do you want? (use 0 if you don't want
linear combinations)
clear extra_options; clear model;
extra_options.importance=0; %0=(Default) Don't, 1=calculate importance
nodesize=1; %1=Default for classification, bigger nodesize leads to smaller trees

% number of used samples for training
X_use=X(:,:,no_used_samples_training,:);
Y_use=Y(:,:,no_used_samples_training,:);

% train
if number_of_lin_comb>0 %linear combinations

X_lin=zeros(number_of_lin_comb,no_timepoints,length(no_used_samples_training),no_noised_sample
s+1);
    rand1=randi(no_frequences,1,number_of_lin_comb);
    rand2=randi(no_frequences,1,number_of_lin_comb);
    for ind21=1:no_noised_samples+1
        for ind22=1:length(no_used_samples_training)
            for ind23=1:no_timepoints
                for ind24=1:number_of_lin_comb
                    X_lin(ind24,ind23,ind22,ind21)=X_use(rand1(ind24),ind23,ind22,ind21)-
X_use(rand2(ind24),ind23,ind22,ind21);
                end
            end
        end
    end
    X_CV=X_lin;
    Y_CV=Y_use;
else
    X_CV=X_use;
    Y_CV=Y_use;
end
[no_features,~,no_trials,~]=size(X_CV);
X_CV_trn=X_CV(:,int32(date/time_offset*SampleRate+1),:,1);
X_CV_trn=reshape(X_CV_trn,no_features,no_trials)';
```

```
Y_CV_trn=Y_CV(:,int32(date/time_offset*SampleRate+1),:,1);
Y_CV_trn=reshape(Y_CV_trn,1,no_trials)';
if use_noised_samples>0
    for ind19=1:use_noised_samples
        X_CV_trn_noise=X_CV(:,int32(date/time_offset*SampleRate+1),:,use_noised_samples+1);
        X_CV_trn_noise=reshape(X_CV_trn_noise,no_features,no_trials)';
        X_CV_trn=[X_CV_trn;X_CV_trn_noise];
        Y_CV_trn_noise=Y_CV(:,int32(date/time_offset*SampleRate+1),:,use_noised_samples+1);
        Y_CV_trn_noise=reshape(Y_CV_trn_noise,1,no_trials)';
        Y_CV_trn=[Y_CV_trn;Y_CV_trn_noise];
    end
end
extra_options.nodesize=nodesize;
model = classRF_train(X_CV_trn,Y_CV_trn,number_of_trees,m_try,extra_options);

% number of used samples for testing
X_use=X(:,:,no_used_samples_testing,:);
Y_use=Y(:,:,no_used_samples_testing,:);

% test
if number_of_lin_comb>0 %linear combinations
    X_lin=zeros(number_of_lin_comb,no_timepoints,no_used_samples,no_noised_samples+1);
    for ind21=1:no_noised_samples+1
        for ind22=1:length(no_used_samples)
            for ind23=1:no_timepoints
                for ind24=1:number_of_lin_comb
                    X_lin(ind24,ind23,ind22,ind21)=X_use(rand1(ind24),ind23,ind22,ind21)-
X_use(rand2(ind24),ind23,ind22,ind21);
                end
            end
        end
    end
    X_CV=X_lin;
    Y_CV=Y_use;
else
    X_CV=X_use;
    Y_CV=Y_use;
end
mce=zeros(1,no_timepoints);
for ind10=1:no_timepoints
[no_features,~,no_trials,~]=size(X_CV);
X_CV_tst=X_CV(:,ind10,:,1);
X_CV_tst=reshape(X_CV_tst,no_features,no_trials)';
Y_CV_tst=Y_CV(:,ind10,:,1);
Y_CV_tst=reshape(Y_CV_tst,1,no_trials)';
Y_hat = classRF_predict(X_CV_tst,model);
mce(ind10)=sum(Y_hat~=Y_CV_tst)/length(Y_CV_tst);
end
```

9.2 Appendix on Section 6

9.2.1 MATLAB / Simulink model for recording

The BCI MATLAB model shown in Figure 9.1 was used for data recording in runs one to five.

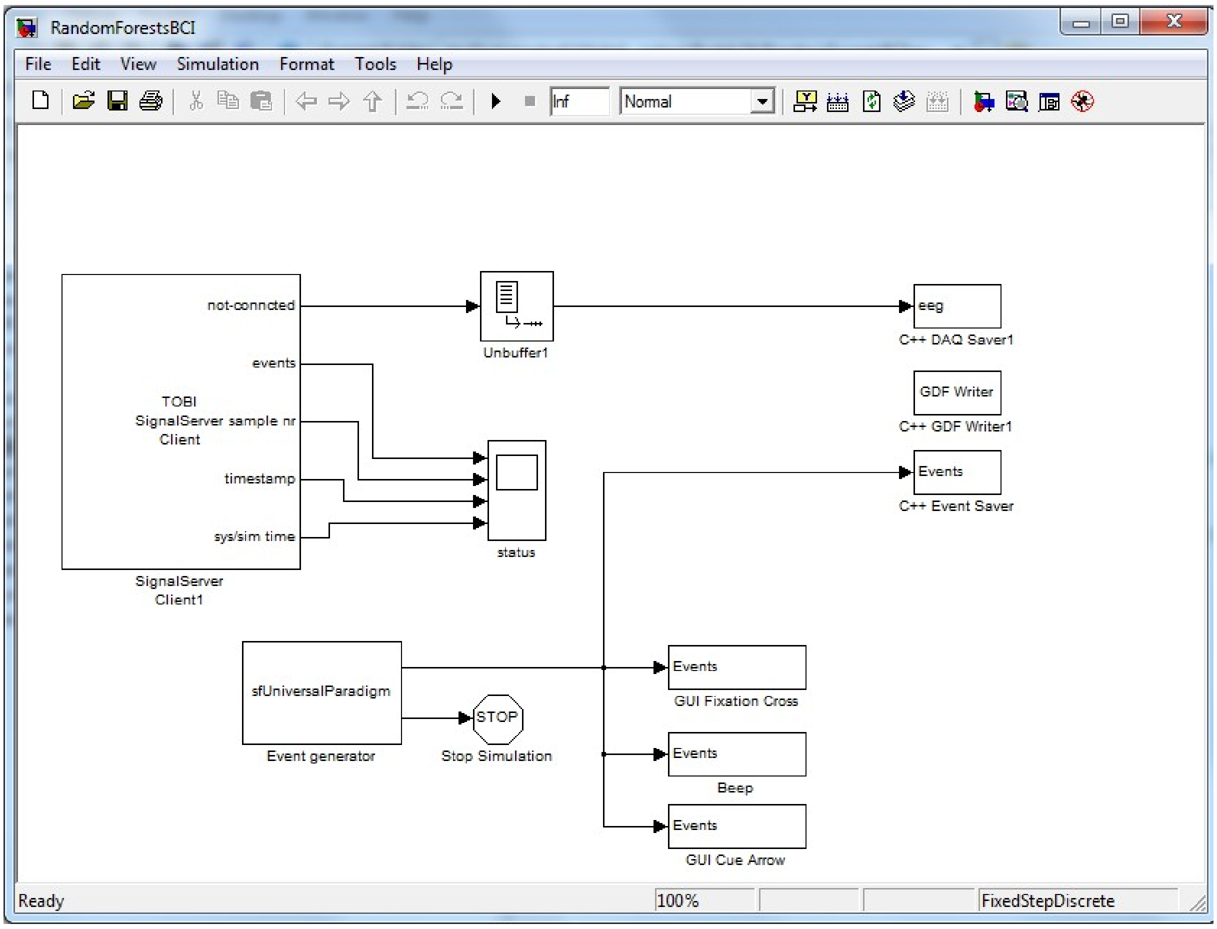

Figure 9.1: MATLAB / Simulink model used for recording the trainings data.

The MATLAB script trainRandomForests.m contains code for extracting and calculating the features of the recorded EEG data and training a RF classifier. The core of this script is shown below:

```
% Calculate Laplace derivation
s1=signals{1,1}(:,:);
s2=signals{2,1}(:,:);
s3=signals{3,1}(:,:);
s4=signals{4,1}(:,:);
s5=signals{5,1}(:,:);
```

```
s6=signals{6,1}(:,:);
s7=signals{7,1}(:,:);
s8=signals{8,1}(:,:);
s9=signals{9,1}(:,:);
s10=signals{10,1}(:,:);
s11=signals{11,1}(:,:);
s12=signals{12,1}(:,:);
s13=signals{13,1}(:,:);
s14=signals{14,1}(:,:);
s15=signals{15,1}(:,:);

Lap_C3=s3-1/4*(s1+s2+s4+s5);
Lap_CZ=s8-1/4*(s6+s7+s9+s10);
Lap_C4=s13-1/4*(s11+s12+s14+s15);

% Make classlabels
time=[];
classlabel=[];

for i=1:length(events.event_code)
    if events.event_code(i)==770
        time=[time,events.position(i)];
        classlabel=[classlabel,1];
    elseif events.event_code(i)==771
        time=[time,events.position(i)];
        classlabel=[classlabel,2];
    end
end

% Make training samples
sample_rate=512;
features=2:41;

for i=1:length(classlabel)
    part_C3_1=Lap_C3(int32(time(i)+1.5*sample_rate):int32(time(i)+2.5*sample_rate-1))';
    part_C3_2=fft(part_C3_1);
    part_C3=((abs(part_C3_2(features))).^2);

    part_CZ=Lap_CZ(int32(time(i)+1.5*sample_rate):int32(time(i)+2.5*sample_rate-1))';
    part_CZ=fft(part_CZ);
    part_CZ=((abs(part_CZ(features))).^2);

    part_C4=Lap_C4(int32(time(i)+1.5*sample_rate):int32(time(i)+2.5*sample_rate-1))';
    part_C4=fft(part_C4);
    part_C4=((abs(part_C4(features))).^2);

    X_trn=[X_trn;part_C3,part_CZ,part_C4];
end

Y_trn=[Y_trn;classlabel'];

%% Train a Random Forests classifier
extra_options.importance=1;

modelRF=classRF_train(X_trn,Y_trn,0,0,extra_options);

save('modelRF','modelRF');
```

9.2.2 MATLAB / Simulink model for feedback

The MATLAB / Simulink model shown in Figure 9.2 was used for data recording and for giving feedback on the online classification accuracy.

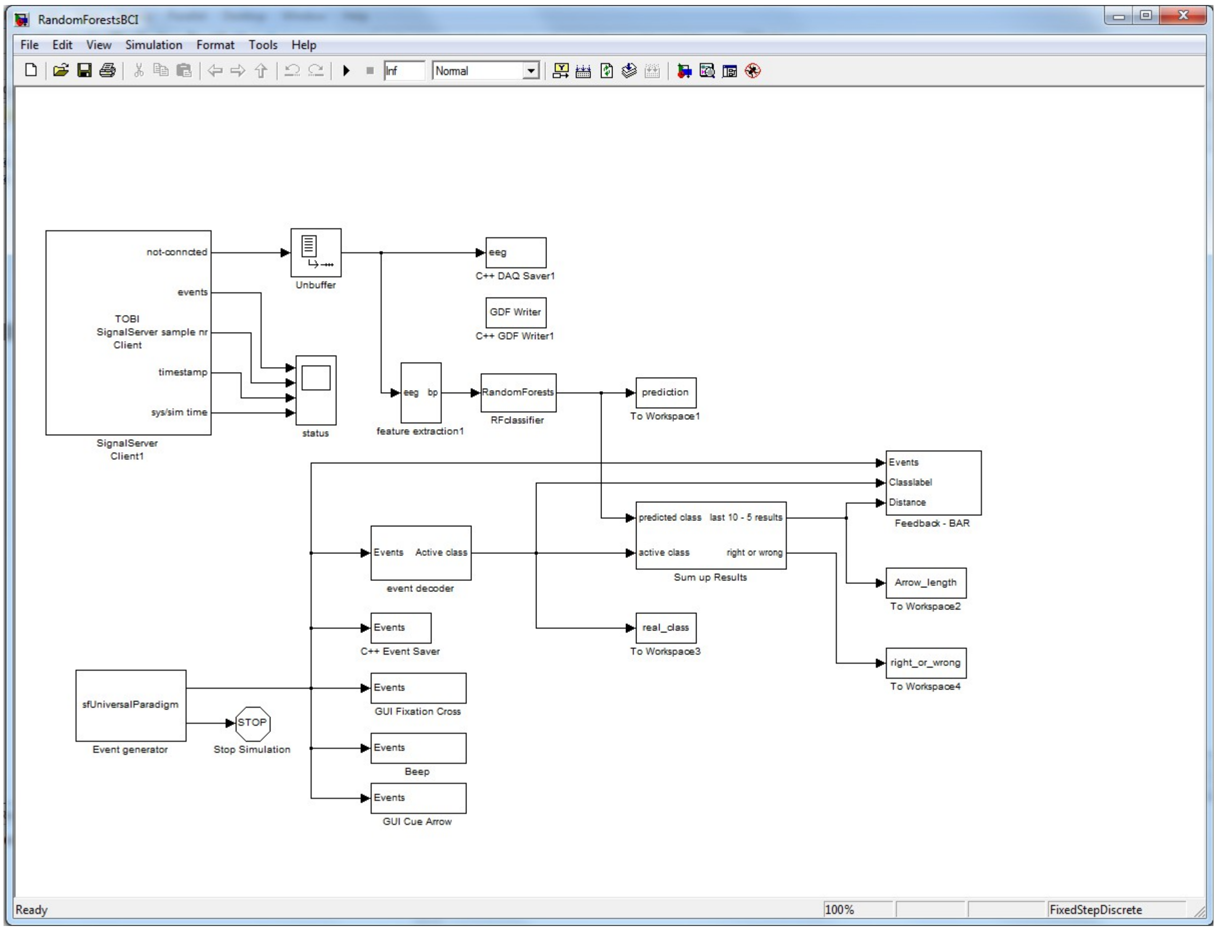

Figure 9.2: MATLAB / Simulink model built and used for recording and giving feedback on the accuracies of the online classifications.

Relevant part of the MATLAB code for s-function “RandomForests” in Figure 9.2:

```
function [sys,x0,str,ts] = RandomForests(t,x,u,flag)
.
.
.
case 3
      sys=classRF_predict(u',rtBCI.modelRF);
.
.
.
```

Figure 9.3 shows the inner life of block “feature extraction” used in Figure 9.2.

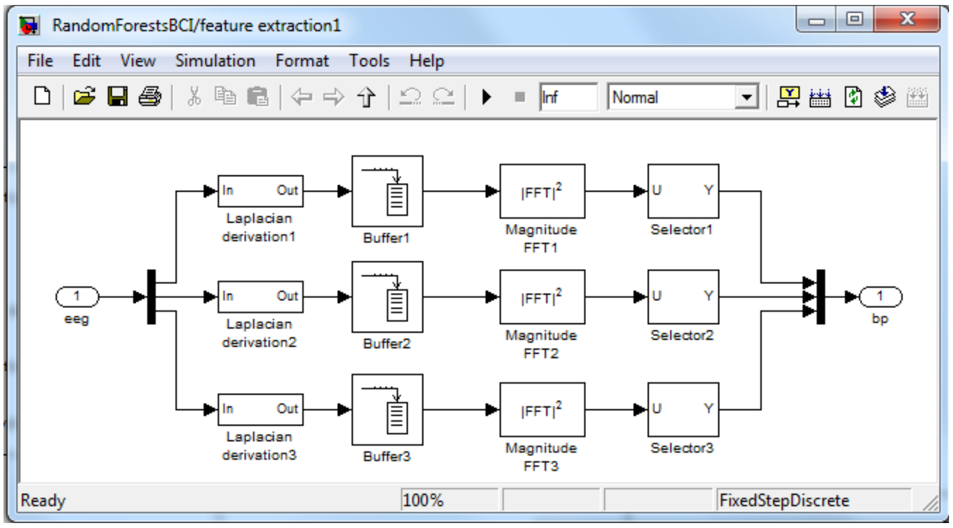

Figure 9.3: MATLAB / Simulink model of the feature extraction (powers of the frequencies of the Laplace derivations of C3, Cz and C4).

Figure 9.4 shows the inner life of block “event decoder” used in Figure 9.2.

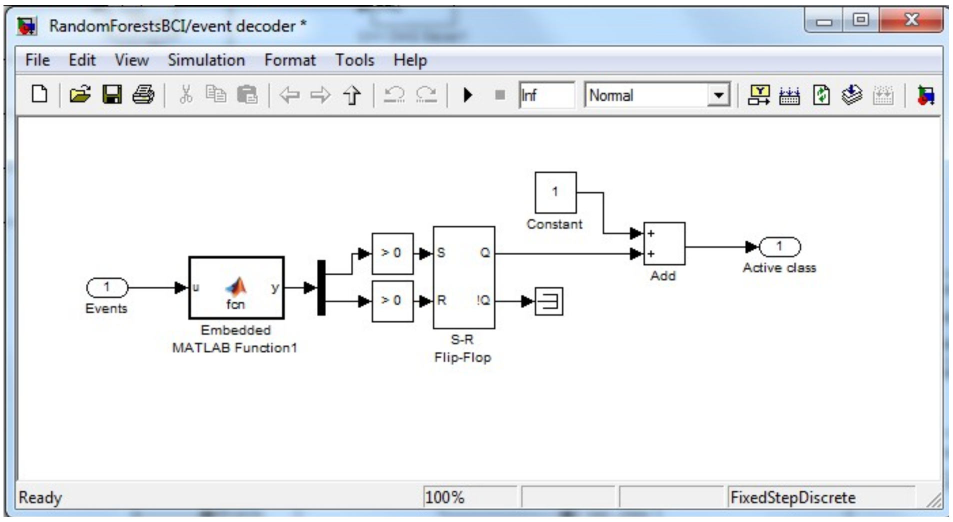

Figure 9.4: MATLAB / Simulink model for the extraction of the active class.

MATLAB code of the embedded function in Figure 9.4:

```
function y = fcn(u)

y = zeros(2,1);

if u(1) > 0
    for k=2:1+u(1)
        if u(k) == hex2dec('302') % class 1
            y(1) = 0;
            y(2) = 1;
        elseif u(k) == hex2dec('303') % class 2
            y(1) = 1;
            y(2) = 0;
        end
    end
end
```

Figure 9.5 shows the calculations for feedback. Feedback was only provided, if at least 50% of the last ten samples were correctly classified. If this was the case, the amount of correctly classified samples was normalized to values between zero and one.

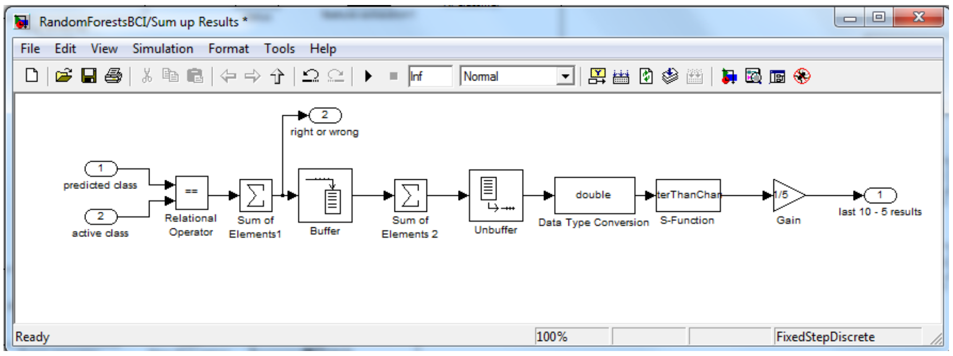

Figure 9.5: MATLAB / Simulink model for summing up the last results on classification.

Relevant part of the MATLAB code for s-function “Better than chance” in Figure 9.5:

```
function [sys,x0,str,ts] = BetterThanChance(t,x,u,flag)
.
.
.
case 3
      if u>=5
          sys=u-5;
      else
          sys=0;
      end
.
.
.
```