Freie Universität Berlin

Department of Mathematics and Computer Science

Max Planck Institute for Human Development

Center for Lifespan Psychology

Combination of Classifiers to Increase Accuracy of EEG Classification

April 19, 2012

Author Julian Karch
Advisor Prof. Dr. Timo von Oertzen
Supervisor Prof. Dr. Raúl Rojas

Declaration of Academic Integrity

I hereby confirm that I prepared this diploma thesis independently and on my own, by exclusive reliance on the tools and literature indicated therein. The thesis has not been submitted to any other examination board.

Berlin, April 19, 2012 _____

Acknowledgments

I want to thank everybody who has directly or indirectly contributed to the success of this thesis.

Especially I want to thank:

Prof. Dr. Timo von Oertzen, from the University of Virginia, for his excellent mentoring throughout this thesis. It has been an honor and a great pleasure to work with him.

Prof. Dr. Raúl Rojas, from the Freie Universität Berlin, for making this diploma thesis possible by supervising it.

Dr. Andreas Brandmaier for being supportive all the time and for answering my questions with an extraordinary patience.

Dr. Timothy Brick for introducing me to the wonderful world of punctuation and for sharing his remarkable knowledge about statistical inference.

Dr. Markus Rampp, from the Rechenzentrum Garching, for assisting me with the cluster computer.

Dr. Myriam Sander for providing the *Attention* data sets.

Dr. Markus Werkle-Bergner and *Michael Schellenbach* for providing the *Oddball* data sets.

Zander et al. (2011), from the Berlin Institute of Technology, for providing the *Motor Imaginary* data sets.

Michael Beckmann, former student assistant of the Max Planck Institute for Human Development, for recording the *Memory* data set with me.

If not indicated otherwise, the affiliation of these persons is the Max Planck Institute for Human Development.

Abstract

The successful classification of single-trial *Electroencephalography* (EEG) signals enables paralyzed people to communicate and can be employed as analysis tool. This thesis investigates the possibility to increase the accuracy of EEG classification systems by combining classifiers that are based on different feature extraction and classification methods that are employed for the classification of EEG signals. This is achieved by comparing multiple classifiers that are based on a combination of classifiers against the best single classifier on data sets originating from four different EEG studies. The results show that a combination of classifiers is able to increase the accuracy by more than 7%. This implies that the general direction in EEG classification research should be changed from "finding the best single classification method" to "finding the best combination of classification methods".

Contents

1. Introduction

Recently, the automatic classification of *Electroencephalography* (EEG) signals via *Pattern Recognition Systems* (PRSs) has gained attention. One main motivation behind this is that the automatic classification of EEG signals enables communication for paralyzed people. Given that there are differences between *classes,* a PRS hypothesizes a *model*, based on a *labeled* data set that captures these differences and can reliably classify a *novel sample* based on that model. PRSs can be used as an EEG data analysis tool by interpreting the separating model a PRS infers from a data set. In contrast to the conventional analysis techniques, which are mostly univariate approaches, the employment of a PRS as analysis tool enables the detection of differences that are based on interactions between multiple variables (van Gerven et al., 2009).

In the last 10 years, researchers proposed a variety of different feature extraction and classification methods for the classification of EEG signals (see Sections 2.2.3 and 2.2.4). Most EEG-PRS are based on one particular combination of feature extraction and classification method (Lotte et al., 2007). It is promising to combine the different feature extraction and classification methods to potentially create a PRS that is, for any given *Pattern Recognition* (PR) task, more accurate than the best PRS that is based on one particular combination of feature extraction and classification method. Because the best out of all possible classifiers can not be obtained, I use the so called ORACLE classifier as base-line comparison. If the ORACLE is asked, it returns the classifier, out of a candidate set of classifiers, that achieves the highest mean accuracy over all data sets for one particular PR task. Due to the employment of various different feature extraction and classification methods, this PRS might be able to perform well on a large variety of EEG data sets. Hence, it produces a separating model, which can be interpreted, on a large set of different data sets. Thus, the first hypothesis of this thesis is: A combination of the different feature extraction and classification methods that are employed for the classification of EEG signals improves the accuracy of the resulting classifier compared to ORACLE and results in a PRS that performs well on a variety of EEG data sets.

The most popular approach to combine feature extraction and classification methods is the employment of so called *Multiple Classifier System* (MCS). A MCS consists of a set of *base-level classifiers* and a *combiner*. All base-level classifier are trained for the same PR task, but each base-level classifier differs from the other base-level classifiers. The combiner combines the decisions from all base-level classifiers to one overall ensemble decision. The resulting classifier is called *ensemble classifier*.

One of the most famous combiners is the simple selection of the best classifier, as estimated on a part of the training set. This combiner is called *Select the Best* (SelectBest) in the remainder. The second hypothesis of this thesis is that a combination of the decisions of the base-level classifiers leads to a more accurate ensemble classifier than the

selection of the best classifier by SelectBest.

An even simpler approach is to only combine the different feature extraction methods, and to employ a single classification method on the concatenation of the outputs of all feature extraction methods. This approach is called *Concatenation* (CONCAT) throughout this thesis. The last hypothesis of this thesis is that the employment of a MCS leads to a more accurate classifier than the CONCAT approach.

When ORACLE is compared against a MCS, the set of candidate classifiers is identical to the set of base-level classifiers. MCSs have been applied successfully for many diverse EEG-PR tasks. Furthermore, it was shown that the combination of multiple feature extraction methods is able to boost the accuracy compared to ORACLE. In all previous studies, MCSs outperformed the simple CONCAT approach. The previous comparisons were all made on one particular type of EEG data sets. However, a systematic comparison on a large set of many different data sets is missing. In this thesis, I compare several different MCSs against CONCAT, ORACLE, SelectBest, and each other on a large set of different EEG data sets. The different MCSs only differ in the combiner they employ. They are all based on the same diverse and broad base-level classifiers (see section 3.4).

A majority of the combiners that I compare have not yet been applied to the classification of EEG signals. Furthermore, I propose several new combiners, which are not limited to the application to EEG-PRSs. While all previous studies used the classification of EEG signals to build a PRS that works well on one particular type of EEG data sets, my goal is to build a PRS that works well on a variety of different EEG data sets.

I apply a subset of the MCS that I propose to an EEG classification problem for that successful classification has not yet been achieved. The motivation behind this is to examine if one MCS is powerful enough to infer a separating model for that problem.

1.1. Outline

The remainder of this thesis is laid out as follows.

In Chapter 2, the mathematical and psychophysiological foundations will be introduced. It will start with a short introduction to PR. After that, the applications of the classification of EEG signals will be introduced in detail. Then, various feature extraction and classification methods that have been employed in previous studies will be introduced. The chapter ends with an introduction to MCS, with an emphasis on different combiners.

Chapter 3 will start with a detailed review of previous work. After that, the learning algorithm that is used to train the ensemble classifiers will be introduced. Also, the newly proposed combiners and the settings for the existing combiners will be presented. Chapter 3 also contains the description of the employed set of base-level classifiers. It will conclude with the details of the implementation.

In Chapter 4, the results of the comparison of the different methods will be presented. The methods will be compared on four different EEG classification tasks and on simulated data sets. After the methods and the implementation details will have been presented, the results on the simulated data sets will be introduced. After that, the results on the

EEG data sets will be shown. Chapter 4 concludes with a summary of the results.

In Chapter 5 a summary of this thesis will be presented and conclusions based on the results will be drawn. It will also contain an Outlook that identifies further possible improvements.

2. Foundations

This section introduces the foundations that are necessary for the understanding of this thesis. It will start with a brief introduction to *Pattern Recognition* (PR), including a treatment of the proper comparison of classifiers. Thereafter, the foundations of, the application of, and the methods for the classification of *Electroencephalography* (EEG) signals will be introduced. The last section will introduce the combination of classifiers.

2.1. Pattern Recognition

This section contains a short introduction to the field of *Pattern Recognition* (PR). A more extensive introduction can, e.g., be found in Duda et al. (2000). PR is a sub-field of machine learning, which in turn is a sub-field of artificial intelligence.

Assume that someone asked you to build a system that separates hippos and giraffes based on their height and weight. To fulfill this task you collect a *data set* that contains the weight and height for each member of a set of hippos and giraffes. One approach to fulfill this task would be to look at the data set and define a separating model based on what *you* have learned about the differences between the two classes, hippos and giraffes.

Supervised learning aims at transferring this learning process, which is necessary to hypothesize a model of the differences, to a computer. Given that there are differences between *classes,* a supervised learning algorithm is an algorithm that hypothesizes a *model*, based on a *labeled* data set, reflecting these differences and classifies a *novel sample* based on that model. The labeled data set contains a number of samples for which the class membership is given by an external source. In our example, the classes are hippos and giraffes. The model could, for example, suggest that if an animal has a height of less than 4 meters it is an hippo, otherwise it is a giraffe. An animal that was not included in the data set can now be automatically classified using this model.

A *Pattern Recognition System* (PRS) is a system that employs a supervised or unsupervised learning algorithm to infer a separating model, which is then employed to classify novel samples. In the remainder of this section I will explain the functionality of a PRS by describing the components of that a PRS typically consists. These components represent the solutions of the different problems one has to solve when designing a PRS. For this thesis, I am concerned only with those PRSs that employ a supervised learning approach.

2.1.1. Components

The components of a PRS are usually sequentially processed. I will introduce the components in their processing order.

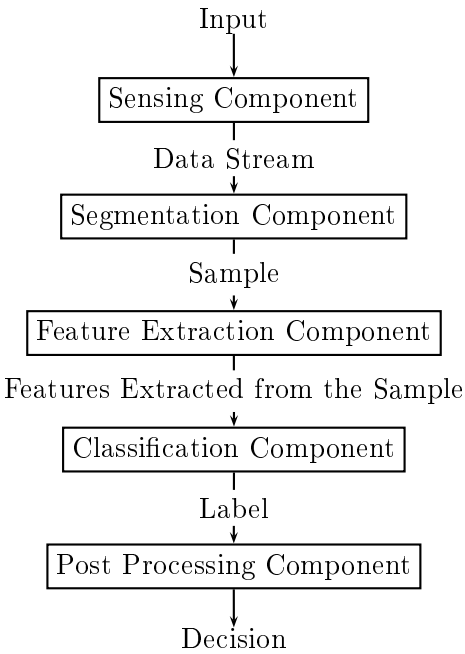


Figure 2.1.1.: Illustration of the data flow between the typical components of a PRS.

Sensing Component

Because a PRS works on a computer, it is only able to process digital data. The task of the first component, named *sensing component*, is to transform chosen aspects of the reality into a format that is readable by a computer.

The sensing component should sense those aspects of the reality that reflect the differences between the classes. Hence, the choice of an appropriate sensing component is crucial for the success of a PRS.

For our hippo and giraffe example, the sensing device might be a camera. For other domains, a microphone or *Electroencephalography* (EEG) electrodes might be used.

Segmentation Component

When using a microphone as sensor for a speech recognition PRS, the computer gets a constant data stream as input. However, most supervised learning algorithms are only able to handle discrete samples as input. Therefore, the constant data stream has to be segmented into samples. The decision how to segment the data results in the *segmentation component*.

Depending on the domain of the classification problem the segmentation component may be crucial to the success of the PRS, or may be completely unnecessary. For a speech PRS the design of a good segmentation component is crucial, as opposed to an e-mail spam filter, where the data is naturally segmented into e-mails. A set of multiple

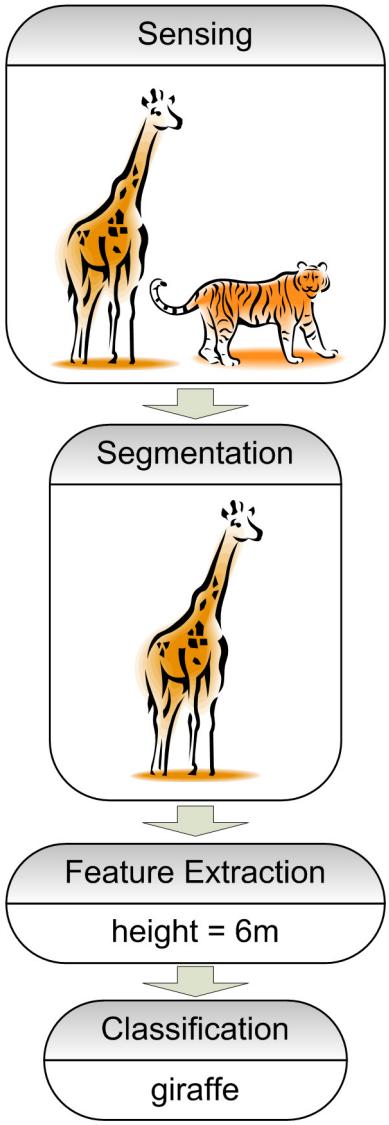


Figure 2.1.2.: Illustration of the outputs of each component. The sensing components returns a picture from which the segmentation component extracts animals. In this case the giraffe. The feature extraction component extracts the height from the picture of an animal and the classification component classifies animals based on their height.

Segmentation A	I	like	this	
Segmentation B	Il	ik	eth	is

Table 2.1.: Illustration of the importance of proper segmentation

samples is called raw *data set.*

Definition 1. Let $X \subseteq K^r$ be the set of unlabeled samples from multiple classes, called *measurement space*, where K is a field. Let $Y = \{y_1, \ldots, y_L\}$ be the finite set of possible labels representing the classes. Furthermore, let $M \subseteq X \times Y$ be the set of samples that are labeled correctly. Then a finite subset $D \subseteq M$ is called *data set* and $(x, y) \in D$ is called *labeled sample.* $N = |D|$ denotes the number of samples in the data set D and y_l the lth label out of the L possible labels.

Feature Extraction Component

In the introduction of this section it was assumed that the data set contains height and weight as measure. This is true if one wants to build a PRS that works on data sets that are made by zoologists. In this case neither a sensing nor a segmentation component is needed as sensing and segmentation is performed by humans.

However, if one wants to build a PRS that enables a robot in the wilderness to distinguish giraffes from hippos, the data set will more likely be a collection of images. If the camera resolution is 640×480, each image is represented by a $3 \cdot 640 \cdot 480 = 921,600$ dimensional vector x. While it is possible to build a successful PRS on a raw data set, with high feature dimensions like this, the approach to transform the samples into a better discriminating and meaningful space is more common. This process is called *feature extraction.* The goal of the feature extraction component is to extract features that differ largely for samples from different classes and are very similar for samples from the same class.

Definition 2. A feature extraction function ϕ is a function that maps samples from the measurement space X to a new measurement space X_{feat}. $\phi_{\theta,\tau} : X \to X_{\text{feat}}$. θ are parameters that are learned from the data set D, and τ are parameters that have to be chosen by the designer, often called hyper-parameters. I call

$$D_{\text{feat}} = \{(x_{\text{feat}}, y) : (x, y) \in D \wedge x_{\text{feat}} = \phi_{\theta,\tau}(x)\}$$

the feature data set.

The feature extraction component can consist of the composite of arbitrary many feature extraction functions $\phi_n \circ \phi_{n-1} \circ \ldots \phi_1$. Each feature extraction function ϕ_i has parameters τ_i and , where θ_i is learned using the data set transformed by ϕ_{i-1}. With that in mind I will just speak of the feature extraction function in the remainder

$$\phi_{\theta,\tau} \quad = \quad \phi_n \circ \phi_{n-1} \circ \ldots \phi_1 \tag{2.1.1}$$

There are two approaches to obtain a feature extraction function. The first approach is the incorporation of prior knowledge about the underlying problem. It is, for example, known that on average giraffes are taller than hippos. Therefore, the height of an animal should be a feature that enables a good distinction between giraffes and hippos. The extraction of the height as feature also reduces the number of feature dimensions from the 921600 pixels of the picture to 1 height value. Therefore, it drastically reduces demands on memory and computation time. This approach simplifies the learning task for the PRS by delegating part of the learning to the designer. The designer specifies and implements the, in this case at least very complex, feature extraction function. If this approach is chosen, no parameters have to be learned from the data set; $\theta = \emptyset$.

The second approach relies less on prior knowledge. It uses a so-called *learning algorithm* to induce the feature extraction function from the data set or in other words to populate θ.

Definition 3. An algorithm $I_{\phi_{?,\tau}}(D) = \phi_{\theta,\tau}$ that takes as input an untrained feature extraction function $\phi_{?,\tau}(x)$ and a data set D and returns the trained feature extraction function $\phi_{\theta,\tau}(x)$ is called a *learning algorithm*

Example algorithms that fall into this category are: Principal Component Analysis (Duda et al., 2000, pp. 568), Independent Component Analysis (Duda et al., 2000, pp. 570), and *Common Spatial Patterns* (CSP) (see Section 2.2.3). These algorithms consist of an untrained feature extraction function $\phi_{?,\tau}$ and the corresponding learning algorithm $I_{\phi_{?,\tau}}$. I will call such algorithms *feature extraction methods* in the remainder. When a learning algorithm is used to populate θ, the designer of the PRS still heavily influences the hypothesized model by choosing the feature extraction method and its hyper-parameters.

Classification Component

The heart of each PRS is the *classification component*. The choice of the classification component is crucial.

Definition 4. A trained classifier $\Psi_{\theta,\tau}$ is a function that maps samples from the feature space X_{feat} to labels $\Psi_{\theta,\tau} : X_{\text{feat}} \to Y$.

Analogous to the feature extraction component, there are two ways to build a classifier, one can define a static classifier based on prior knowledge or use a learning algorithm to induce a classifier from a data set. In contrast to the feature extraction method, no state-of-the-art general purpose PRS exists, to my knowledge, that does not employ a learning algorithm for training its classifier. The classification component takes the feature data set as input.

Note that a classifier could also be defined as the feature extraction function for that the new measurement space is the label set Y. This implies that θ is also populated by the corresponding learning algorithm and τ are hyper-parameters, which must be chosen by the designer. The population of θ by a learning algorithm, which leads to a classifier, is also referred to as the learning algorithm I_Ψ induces a classifier Ψ. I will

call the combination of an untrained classifier and its corresponding learning algorithm *classification method.*

An example classification method is the *mean classifier.* The learning algorithm of the mean classifier calculates the mean for each class

$$m_{y_l} = \frac{1}{|D_{y_l}|} \sum_{(x,y) \in D_{y_l}} x$$

where $D_{y_l} = \{(x, y) \in D_{\text{feat}} : y = y_l\}$. The trained mean classifier assigns a new sample x to the class to whose mean it has the smallest distance, with respect to some distance measure d. Therefore, the discriminating model consists of the mean of every class and the distance function d that is employed . The set of parameters learned from the data consists of the class-wise means $\theta = \{m_{y_1}, \ldots, m_{y_L}\}$ and the hyper-parameter reflects the choice of the distance function $\tau = d$. A new sample $x \in X_{\text{feat}}$ is classified according to the following formula

$$\Psi_{\theta,r}(x) = \arg\min_{y_l \in Y}(d(m_{y_l}, x))$$

Note that the concatenation of the classifier and the feature extraction function

$$\Psi \circ \phi : X \quad \rightarrow \quad Y \tag{2.1.2}$$

also results in a classifier. Therefore, I will speak of a classifier $\Psi_{\theta,\tau}$ for the concatenation of the feature extraction and the classification function in the remainder of this thesis. Furthermore, I will summarize the learning algorithm of the feature extraction method and the learning algorithm of the classification method as $I_{\Psi_{?,\tau}}(D)$. Remember that it is a short for

$$I_{\Psi_{?,\tau}}(D) := I_{\Psi_{?,\tau}}(\phi_{\theta,\tau}(D)) = I_{\Psi_{?,\tau}}(I_{\phi_{?,\tau}}(D)(D)) = \Psi_{\theta,\tau}$$

where $I_{\phi_{?,\tau}}$ is the learning algorithm of the feature extraction function, $I_{\Psi_{?,\tau}}$ the learning algorithm of the classifier, and $\phi_{\theta,\tau}(D)$ denotes that the raw data set is mapped to the feature data set by applying $\phi_{\theta,\tau}$ to each sample in D.

Post Processing Component

The *post processing* component is defined as everything that is done with the classification of a sample.

Most PRS perform some action that is dependent on the classification decision. For example, an iris scanner could open a door if the classifier decided that the iris put in front of the sensors belongs to a person who has access rights to the room.

The post processing component might also be able to add context to a classification. If, for example, a letter recognition system is unsure if a picture of a letter represents a c or an o, but the same system classified the context of the letter with high certainty as "f?r", the post processing component may decide that, based on the context, o has a much higher a-priori likelihood and, hence, assign the sample to the class o.

Another function of the *post processing* component can be the integration of multiple classifiers working on multiple aspects of the input to one decision. This will be presented

in more detail in Section 2.3. But first, I explain how to get a valid measure of the performance of a classifier.

2.1.2. Notation

Throughout this work I will use the following notation, originating from the previous section. D will be the raw data set, containing samples (x_i, y_i), with cardinality N. D is assumed to be a representative subset of the whole population M. $Y = \{y_1, \ldots, y_L\}$ describes the set of labels with cardinality L, X denotes the measurement space, and Ψ a classifier induced by the learning algorithm I_Ψ. y_i refers to the label of the ith sample in the data set D, while y_l refers to the lth out of the possible labels Y.

2.1.3. Estimation of the Performance of a Classifier

Loss and Risk

After creating a classifier, its performance is usually of interest. Questions related to the performance are typically: Is the performance of a classifier sufficient for a given task? Is it performing better than another classifier? The performance of a classifier is usually quantified with the help of a *loss function*.

Definition 5. A *loss function* L is a function that maps a labeled sample $(x, y) \in$ M and a classifier Ψ to a cost term. $L(x, y, \Psi) \in \mathbb{R}_{\geq 0}$

The most basic and most often employed loss function is the zero-one loss function.

Definition 6. Let x, y,Ψ be as in Definition 5. The zero-one loss function is then defined as

$$L_{01}(x, y, \Psi) = \begin{cases} 0 & \text{if, } \Psi(x) = y \\ 1 & \text{otherwise} \end{cases}$$

The zero-one loss function assigns, independently from the true label y, every misclassification the cost 1. Correct classifications are assigned zero cost. Other loss functions are, for example, used if the cost of a misclassification depends on the true class. A typical example for unequal misclassification costs are medical tests. In most cases the consequences are less severe if a medical test classifies a patient as sick who is not sick compared to the situation when the test classifies a patient as healthy who is sick. Therefore, for a medical test mostly asymmetric loss functions are used, which assign the misclassification of a sick patient a high cost.

The *risk* of a classifier is the expected loss. It is the most common performance measure for classifiers.

Definition 7. The *risk of a classifier* is defined as

$$R(\Psi, p) = \int\limits_{x \in X, y \in Y} L(x, y, \Psi) p(x, y) dx dy$$

where L is a loss function and $p(x, y)$ the joint probability mass function of X and Y.

When using the zero-one loss function, an equivalent measure for the risk is the accuracy.

Definition 8. Let $R_{01}(\Psi, p)$ be the risk calculated with L_{01} as loss function. Then

$$\mathrm{acc}(\Psi, p) = 1 - R_{01}(\Psi, p)$$

is called *accuracy* of classifier Ψ.

Note that the accuracy represents the probability that a classifier Ψ predicts the true class, $\mathrm{acc}(\Psi, p) = P(\Psi = \text{correct})$. If $p(x, y)$ were known, the accuracy could be calculated directly .

Also, if $p(x, y)$ were known, the classification task would become trivial. It can be shown that the Bayes classifier

$$\mathrm{Bayes}(x) = \arg\max_{y_l \in Y} \frac{p(x|Y = y_l)P(Y = y_l)}{p(x)}$$

is the classifier with the highest accuracy (Duda et al., 2000, pp. 24). No classifier is able to achieve a higher accuracy than Bayes.

But when building a classifier, the joint probability mass function $p(x, y)$ is rarely known. Only a finite subset D of the whole population M is available. Hence, the design of other classifiers than Bayes is reasonable, and the accuracy can only be estimated on the finite data set D.

Estimation Methods

There are several methods for estimating the accuracy, which I will introduce in the remainder of this section. Independently of the method one chooses to estimate the accuracy, it is defined as follows.

Definition 9. Let D be a data set, the estimated accuracy of the classifier Ψ on that data set is

$$\mathrm{acc}_{\mathrm{es}}(\Psi, D) = 1 - \frac{1}{N} \sum_{(x,y) \in D} L_{01}(x, y, \Psi)$$

Recall that $N = |D|$.

Note that one usually is not interested in the performance of a static classifier Ψ but in the performance of the classifier that is induced by a learning algorithm I_Ψ from the data set D. The most naive method to estimate the accuracy is to use the same data set for estimating the accuracy as for inducing the classifier.

Definition 10. When estimating the accuracy on the same data set that was used for inducing the classifier, the estimated accuracy is called *training accuracy* and calculated as follows

$$\mathrm{acc}_{\mathrm{train}}(I_\Psi, D) = \mathrm{acc}_{\mathrm{es}}(I_\Psi(D), D)) = 1 - \frac{1}{N} \sum_{(x,y) \in D} L_{01}(x, y, I_\Psi(D))$$

The training accuracy is not a good estimate of the accuracy. It is a biased estimate. The training accuracy is usually significantly higher than the true accuracy because the data on that the accuracy is estimated is not independent from the data that was used to induce the classifier.

The general solution to that problem is to divide the data set D into two disjoint data sets, D_h and D_t. The inducer I_Ψ employes D_t to induce the classifier Ψ. The accuracy of Ψ is then estimated on D_h. Since one part of the data set is held out from training, this method is called *holdout method*.

Definition 11. The accuracy estimated by the holdout method is defined as

$$\mathrm{acc}_{\mathrm{es}}(I_\Psi, D) = \mathrm{acc}_{\mathrm{es}}(I(D_t), D_h) = 1 - \frac{1}{N} \sum_{(x,y)\in D_h} L_{01}(x, y, I_\Psi(D_t))$$

where $D_h = D/D_t$.

It is common to use $\frac{2}{3}$ of the data set for the training set D_t and $\frac{1}{3}$ for the holdout set D_h. Assuming that the learning algorithm I_Ψ gets better with a bigger data set, the accuracy estimated by the holdout method, in opposition to the training accuracy, yields an underestimation of the accuracy (Kohavi, 1995). This problem is severe when the data set is small.

To utilize the complete data set for the estimation of the accuracy, a method called *n-fold cross-validation* is often employed. This method basically repeats the holdout method with different holdout sets. The data set is separated into n disjunctive sub sets, each containing N/n samples. For each subset D_i the learning algorithm is trained with the remainder of the data set D/D_i, and the accuracy of the resulting classifier is estimated on D_i. The accuracy estimated by n-fold cross-validation is the summed loss over the n folds divided by the number of samples N.

Definition 12. The accuracy estimated by the n-fold cross-validation method is defined as

$$\mathrm{acc}_{\mathrm{cv}}(I_\Psi, D) = 1 - \frac{1}{N} \sum_{i=1}^{n} \sum_{(x,y)\in D_i} L_{01}(x, y, I_\Psi(D/D_i))$$

where $D = \cup_{i\in\{1,..,n\}} D_i$ and $D_i \cap D_j = \emptyset$ if $i \neq j$.

When estimating the accuracy using n-fold cross-validation, one has to decide how many disjunctive data sets to create and how to create them. Kohavi (1995) showed that 10 data sets (folds) are a good trade-off between bias and variance of the resulting estimator and that stratification leads to a decrease of both variance and bias of the estimated accuracy. Stratification means that the folds D_i contain roughly the same proportions of the classes as the original data set D.

When the data set D is imbalanced, i.e., D does not contain equal proportions of all classes, a better measure of the performance of a classifier than the accuracy is the *Balanced Accuracy* (BAC) (Brodersen et al., 2010).

Definition 13. The BAC is defined as the average per class accuracy

$$\mathrm{bac}(\Psi, p) = \frac{1}{L} \sum_{l=1}^{L} \mathrm{acc}(\Psi, p_{|M_{y_l}}(x, y))$$

where $M_{y_l} := \{(x, y) \in M : y = y_l\}$

Analogous to the accuracy, the BAC can also be estimated using one of the introduced methods.

2.1.4. Comparison of Classifiers

Independent of the method used for the estimation of the performance of a classifier, the performance measure depends on the data set. As long as $D \neq M$, it is a random variable that potentially changes if a different data set is drawn from M. Thus, methods from the field of statistical inferences need to be applied to answer questions like, "does a classifier perform better than chance" or "does a classifier perform better than another one".

The general procedure used in statistical inference is to formulate a hypothesis. Hypotheses are expressed in so called test statistics. Test statistics are certain characteristics of the data. E.g., a test statistic is the estimated BAC of a classifier.

To substantiate that the hypothesis is true, the contrary of the hypothesis, called null hypothesis, is assumed, and the probability under the null hypothesis of obtaining test statistics that are at least as extreme as those observed is calculated. This probability is often called *p-value*. If the p-value falls below a certain threshold, which is often called α, the null hypothesis is rejected, and the original hypothesis is believed to be true.

When the hypothesis is a difference hypothesis, e.g., classifier A has a higher accuracy than classifier B, the difference is called *statistically significant* at the 5% level, which is often simply referred to as significant if the corresponding p-value falls below 5%.

2.1.4.1. Comparing Against Random Guessing

When comparing a classifier Ψ against random guessing using the BAC as test statistic, the null hypothesis is bac(Ψ,p) = 0.5. Under the null hypothesis and independent samples, the estimated accuracy $\mathrm{acc}_{\mathrm{es}}$ for each class is distributed as

$$\frac{1}{N_{y_l}} B(N_{y_l}, 0.5)$$

where N_{y_l} is the number of samples in the data set D with label y_l, and $B(n, p)$ is the binomial distribution with n trials and success probability p per trial. Note that p is different from the probability mass function. The estimated BAC is distributed as

$$\frac{1}{L} \sum_{l=1}^{L} \frac{1}{N_{y_l}} B(N_{y_l}, 0.5)$$

The Binomial distribution converges to the normal distribution for large n, approximately in the range of $n > 30$. The data sets that are used for estimating the BAC usually contain more than 30 samples per class. Therefore, the sum converges to a normal distribution with mean 0.5 and variance

$$\begin{aligned}\mathrm{var}(\frac{1}{L}\sum\frac{1}{N_{y_l}}B(N_{y_i},0.5)) &= \frac{1}{L^2}\sum\mathrm{var}(\frac{1}{N_{y_l}}B(N_{y_l},0.5)) = \frac{1}{L^2}\sum\frac{1}{N_{y_l}^2}\mathrm{var}(B(N_{y_l},0.5)) \\ &= \frac{1}{L^2}\sum\frac{1}{N_{y_l}^2}\frac{1}{4}N_{y_l} = \frac{1}{L^2}\sum\frac{1}{4N_{y_l}}\end{aligned}$$

Let b be the BAC achieved on D. The probability that the estimated BAC b or a higher BAC is achieved by a classifier that independently guesses is therefore:

$$P(\mathrm{bac}_{es}\ (\Psi,\mathrm{D}) > b|\mathrm{bac}(\Psi,p) = 0.5) = \int_b^1 \mathcal{N}(x;0.5,\frac{1}{L^2}\sum\frac{1}{4N_{y_l}})dx \tag{2.1.3}$$

where $N(x;\mu,\sigma^2)$ is the likelihood of x under the univariate normal distribution with mean μ and variance σ^2. If this probability is smaller than 5%, we say that Ψ performs significantly better than random guessing.

Comparing two Classifiers on a Single Data Set

For comparing two classifiers, Ψ_1 and Ψ_2, Salzberg (1997) suggests a different procedure.

He proposes to use the following test statistics. The number of samples in the data set where Ψ_1 predicts the correct class and Ψ_2 predicts the wrong class.

$$s = |\{(x,y) \in D : \Psi_1(x) = y \wedge \Psi_2(x) \neq y\}|$$

Analogous to that, the number of samples in the data set where Ψ_2 predicts the correct class and Ψ_1 predicts the wrong class.

$$f = |\{(x,y) \in D : \Psi_2(x) = y \wedge \Psi_1(x) \neq y\}|$$

The null hypothesis says that no classifier performs better and guesses are independent. Thus, $E(s) = 0.5(s+f) = E(f)$, where $E(s)$ denotes the expected value of the random variable s. Let s_{emp} and f_{emp} be the values observed for a certain data set. Furthermore let s_{emp} be greater than f_{emp}. Then, under the null hypothesis, the probability that a value for s that is at least as high as s_{emp} is observed is

$$P(s > s_{\mathrm{emp}}|\Psi_1 = \Psi_2) = \sum_{k=s_{\mathrm{emp}}}^{s_{\mathrm{emp}}+f_{\mathrm{emp}}} B(k;s_{\mathrm{emp}}+f_{\mathrm{emp}},0.5)$$

where $B(k;n,p)$ denotes the probability of k successes under the binomial distribution with n trials and success probability p. Thus, if this value falls below 5%, it is believed that Ψ_1 performs better than Ψ_2. If the original hypothesis was "$\Psi_1 > \Psi_2$ or $\Psi_1 < \Psi_2$", the p-value has to be multiplied by two to correct for the two comparisons.

2.1.4.2. Comparing Multiple Classifiers on Multiple Data Sets

For comparing multiple classifiers $\Psi_1, \ldots, \Psi_K$ on multiple data sets $D_1, \ldots, D_n$ Demšar (2006) proposes to use the Friedman test.

The Friedman test computes for every classifier Ψ_k and data set D_i the rank, based on an arbitrary performance measure, compared to the other classifiers. It is not important if the best or the worst performing learning algorithms gets assigned rank 1 or rank K. I will assume that the best classifier gets assigned rank K in the remainder. Let r_k^i be the rank of the classifier Ψ_k on the data set D_i. The Friedman test employs the mean rank over all data sets

$$R_k = \frac{1}{n} \sum_{i=1}^{n} r_k^i$$

per classifier as test statistics. The test statistics and distribution for the null hypothesis that all averaged ranks R_k are equal can be found in Demšar (2006).

After the hypothesis that there are no differences between the classifiers was falsified, a post-hoc procedure can be applied to test which pairs of classifiers differ. Under the null hypothesis, no difference between the two learning algorithms Ψ_o and Ψ_m, the difference of the two ranks is mapped to a *z-value* by the following formula

$$z = \frac{R_o - R_m}{\sqrt{\frac{K(K+1)}{6n}}}$$

The z-value can be transformed to a p-value as follows

$$p = \int_{-z}^{z} N(x; 0, 1)\, dx$$

When comparing a set of classifiers over a set of data sets, usually two typical question are of interest. Which classifier out of a set of classifiers $T := \{\Psi_1, \ldots, \Psi_K\}$ performs better than a base-line classifier Ψ, and which classifier out of a set of classifiers performs best.

For comparing a set of classifiers against a base-line classifier a p-value for each classifier from the set can be obtained by comparing the classifier against the base-line method, using the aforementioned method. But the threshold α has to be decreased.

Let H be the null hypothesis that there are no differences between any classifier from the set T and the base-line classifier, and let H_k be the null hypothesis that the classifier Ψ_k is identical to the base-line classifier. Note that the null hypothesis $H_1, \ldots, H_K$ are independent. Thus, under the null hypothesis H, the expected number of rejected null hypothesis H_k is $K\alpha$. To correct for this inflated α error, the α threshold for each hypothesis $H_i \in \{H_1, \ldots, H_k\}$ is divided by K. This procedure is called *Bonferroni correction.*

Static Shaffer Procedure

If comparing all pairs of classifiers, it is also possible to use the Bonferroni correction. However, the Bonferroni correction assumes that each comparison made is completely independent of the others. Because this assumption is not met, the Bonferroni correction is overly conservative.

García and Herrera (2008) compared correction schemes that exploit dependencies between the hypothesis with regard to their suitability for the comparison of multiple classifiers. They concluded that the Bergmann-Hommel procedure performs best but is also computational complex and hard to understand. *Shaffer's Static Procedure* (SSP) has almost equivalent power but is much simpler.

When comparing all pairs out of K classifiers, there exists a total of $\frac{K(K-1)}{2} := G$ difference hypotheses. Each differences hypothesis corresponds to a hypothesis $\Psi_m \neq \Psi_o$. For each hypothesis the p-value for the corresponding null hypothesis can be obtained by employing the post-hoc Friedman test, introduced in the previous section. The first step of the SSP is to sort the null hypothesis by their p-values. Let $H_1, \ldots, H_G$ be the null hypotheses sorted by their corresponding p-value. In general, the α value corresponding to the ith hypothesis is corrected by the number of hypotheses that can be true given that $(i-1)$ hypotheses are false.

Hence, H_1 is rejected if $p \leq \alpha/G$. Note that each null hypothesis corresponds to the proposition that one pair of classifiers, Ψ_m and Ψ_o, performs the same $\Psi_o = \Psi_m$. If H_1 is rejected, $\exists m, o \in \{1, \ldots, K\} : \Psi_o \neq \Psi_m$. Therefore, for all other classifier $\forall k \in \{1, \ldots, K\}/\{m, o\} \Psi_k \neq \Psi_m \vee \Psi_k \neq \Psi_o$. Hence, if H_1 is wrong, at least $K-1$ additional null hypothesis have to be wrong. Thus, the correction term t_2 for the second hypothesis is $t_2 = G - (K-1)$. The algorithm to calculate the correction term for every stage and a more extensive description of SSP can be found in Shaffer (1986).

A final remark

In this section, for simplicity, I always assumed to compare classifiers. In this thesis I will rather compare learning algorithms. I will do so by estimating the BAC of the learning algorithms on each data set, using 10-fold stratified cross-validation. The introduces test can then be applied on these estimates in the same way.

2.2. Classification of Electroencephalographic Signals

The classification of *Electroencephalography* (EEG) signals is an application area, out of many, of *Pattern Recognition* (PR). In the remainder of this section I will first introduce the basics of EEG. After that, I will give an overview about why the classification of EEG is useful and what its applications are. The last subsections will focus on the feature extraction and classification methods that are usually employed to build EEG-*Pattern Recognition Systems* (PRSs).

2.2.1. Electroencephalography

EEG is a neuroimaging method based on electrical fields generated by neural activity. The field potentials are measured by electrodes at different locations on the scalp, at a certain sampling rate. Each electrode provides a time series of electrical potentials.

Classification of EEG signals aims at separating different brain states. While resting state classification is also of interest, most EEG-PRSs try to separate signals that are induced by certain events. This signals are called *Event Related Potentials* (ERP). For example, one of the data sets used in this thesis comes from an experiment where the participants heard a high- or a low-pitched tone. The potentials that were generated by the processing of the tone are the ERP. The remaining brain activity is considered noise. The main task of a EEG-PRS is to separate the ERP from the noise.

To generate the data sets that are needed for supervised learning, typically for every to-be-separated class multiple repetitions are recorded. For example, multiple repetitions of the presentation of a high- and a low-pitched tone. In the EEG context these repetitions are called *trials*. The samples for the data sets are usually generated by cutting a fixed sized time interval out of each trial. Thus, each element of the raw data set typically is of the form

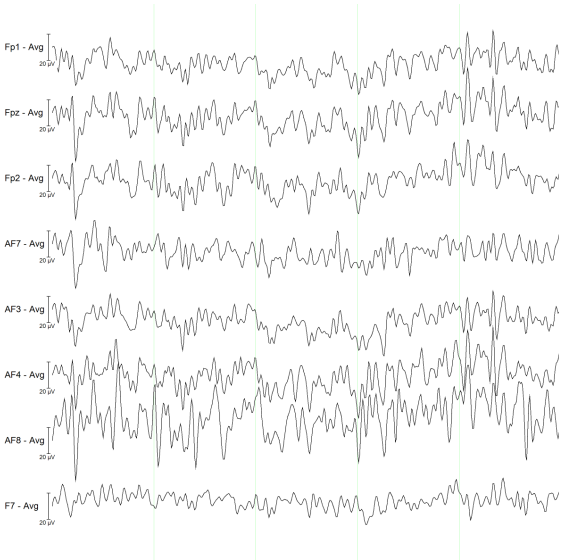
$$x \in \mathbb{R}^{C \cdot T}$$

where C is the number of channels (electrodes) and T the number of time points that are extracted. Because there usually is a one to one relationship between experimental trials and samples in the data set, I will also call samples trials in the remainder of this thesis.

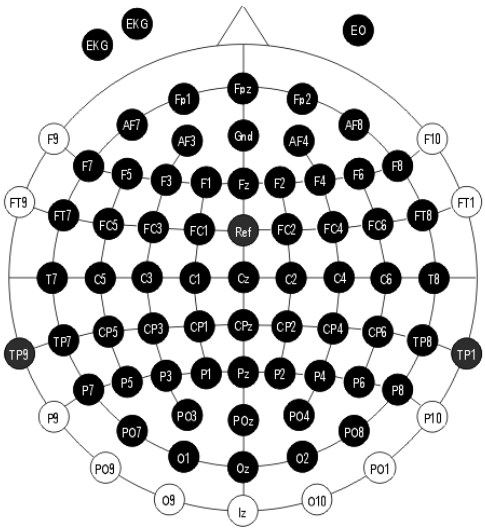
The ERP are typically weaker than the ongoing brain activity. Even worse, the EEG signal is additionally disturbed by external noise sources (Lotte et al., 2007). The most prominent noise sources are electrical fields induced by eye movements, muscle activity, and electrical devices / power jams. Typically, the magnitude of those noise signals is of orders higher than the magnitude of the brain signal. Additionally, because recording EEG signals is comparatively time consuming and the to-be-performed tasks are often monotonous and, hence, exhausting, the data sets typically contain less than thousand trials; often substantially less. One trial is usually around one seconds long and sampled at 1000Hz. Thus, for one trial the dimensionality of the raw amplitude data is $C \cdot T = 60 \cdot 1000 = 60,000$. Most EEG-PRSs reduce the number of data dimensions by extracting features from the raw data. But still, it is often the case that the number of features is higher than the number of trials. This problem is often referred to as curse-of-dimensionality.

To summarize this: The three major challenges when building an EEG-PRS are small training data sets, high dimensionality, and a low signal to noise ratio. As we will see in Section 2.2.3, researchers came up with various methods to deal with these problems. I will first introduce the major applications of the classification of EEG signals in the following section.

(a)



(b)



(c)

Figure 2.2.1.: (a) Picture of a participant during an EEG Experiment. (b) EEG signal from 8 channels during 1 second. (c) Example of a 64 channel electrode cap montage (Brain Products, 2012).

2.2.2. Applications

There are two types of major applications of EEG classification: Enabling direct brain computer communication and the utilization of EEG classification as an analysis method in neuroscience.

Brain Computer Interfaces

A *Brain Computer Interface* (BCI) is a PRS that enables the communication between a person and a computer via brain signals. Most current BCI systems use EEG as sensor because, compared to other neuroimaging methods, such as *functional magnetic resonance imaging* (fMRI) and *Magnetoencephalography* (MEG), EEG is cheap, has a good time resolution and is portable.

The most prominent application of BCIs is the enabling of communication for paralyzed patients (Sellers et al., 2007, Hinterberger et al., 2007, Pfurtscheller et al., 2007, Blankertz et al., 2007). Another popular application of BCIs is the control of artificial limbs (Pfurtscheller et al., 2007). Both applications are realized by distinguishing brain patterns that can willfully be generated by the patients.

Most current BCIs are only able to distinguish two classes and work in a synchronous mode. Synchronous mode hereby refers to the fact that the classification has to be triggered by external cues. A patient can not send commands to the computer spontaneously. These are the two main reasons why the information transfer rate of current BCI systems is less than 0.5bit/s. However, for people who are not able to communicate at all even this small information transfer rate means a tremendous improvement of their situation.

One major outstanding issue of BCI research is that, while it was proven that most healthy subjects and also patients with only little residual muscular control are able to control a BCI, no research lab has yet reported the successful control of BCI by a completely locked-in patient (Kübler et al., 2007). In the completely locked-in state no muscular control and, thus, no communication is possible. The tragedy of that circumstance is that the completely locked-in patients would benefit most of a BCI. It is simply the only chance for them to communicate. All other patients are also able to communicate with the help of their muscles.

This tragic situation might be one cause why recently the use of BCIs for healthy subjects has gained attention. It was, e.g., used as controlling device for computer games (Blankertz et al., 2010b), an autonomous car (Autonomos Labs, 2011), and a pinball machine (Tangermann et al., 2009).

BCIs can be subdivided into active and passive BCIs (Zander and Kothe, 2011). Active BCIs are characterized by the fact that the brain activity that is classified is willfully generated by the user. The application examples above all belong to the group of active BCIs. In contrast to that, passive BCIs aim at classifying different brain patterns that are not willfully generated by the user. For example, a passive BCI was used as a tool in a neuroscience study (Jensen et al., 2011) to introduce brain-state dependent stimuli.

Single-trial Analysis

The classification of EEG signals can be employed as so called single-trial analysis method. A classifier is trained on a particular EEG data set, and the model that is hypothesized by the classifier is interpreted. This differs from the conventional analysis of EEG signals, which reduces the noise by averaging over trials and subjects. In contrast to the conventional analysis technique, single-trial analysis is able to detect differences that are based on interactions between multiple variables (features) (van Gerven et al., 2009). Additionally, it accounts for the per-subject and the per-trial variance. In spite of these advantages, there are relatively few publications that apply single-trial analysis to EEG data (Parra et al., 2002, Blankertz et al., 2011, van Gerven et al., 2009). In contrast to that, the employment of PR methods for data analysis is widely spread in the fMRI community.

2.2.3. Feature Extraction Methods

Notation

In the remainder of this chapter I will use the following notation. A data set consists of N trials. A trial from the raw data set consists of an element $E_i \in \mathbb{R}^{C \times T}$ from the raw input space and the corresponding label $y_i \in Y$. Each trial from the feature data set consists of a feature vector $x_i \in X$ and the corresponding label y_i. When the index is not needed, it is omitted. X is called the feature space. The number of different classes is denoted by L. r denotes the dimensionality of the feature space X.

Raw Electroencephalography Signals

The most straightforward feature extraction method for the classification of EEG data is to employ the raw field potentials. The feature vector x consists of the concatenation of the time series of all channels $x \in \mathbb{R}^{C \cdot T}$. Indeed, as Lotte et al. (2007) describe, several successful BCIs used the raw field potentials as input for their classification component.

Spatio-temporal features

To reduce the dimensionality of the feature vector, in comparison to the feature extraction method that simply employs the raw amplitude data, Blankertz et al. (2011) suggest to average the time series from each channel in certain intervals. Let $I = \{I_1, \ldots, I_K\}$ be sets of time points of interest. $I_k \in I$ is typically an interval. For every channel c the method generates K features

$$x_c(I) = [\text{mean}(\{E(c,t)\}_{t \in I_1}), \ldots, \text{mean}(\{E(c,t)\}_{t \in I_K})]$$

where $E(c,t)$ refers to the data point in the cth channel at time point t in trial E. The final feature vector x is the concatenation of the feature vectors x_c from all channels. In general, this methods leads to so called *Spatio Temporal Features* (STF).

This approach is, of course, not limited to the usage of the mean as aggregation method. Indeed, all feature extraction methods presented in the following can also by applied to a set of intervals.

Bandpass Filter

For EEG signals several frequency bands were established in which characteristic dominant brain rhythms can be found. The bands are referred to as α (8-12 Hz), β (12-30 Hz), γ(30-80 Hz), δ(0-4 Hz) and θ(4-8 Hz) (Herrmann et al., 2004).

A bandpass filter can be used to extract the signal of these bands. It transforms a signal such that it only contains frequency components of the specified band. All other oscillatory components are removed. For an extensive introduction to bandpass filters refer to Shenoi (2005).

Bandpass filters are often used as prepossessing step for successive feature extraction methods. But they are also used as the only feature extraction method. There exist several BCIs that employ a bandpass filter as feature extraction method (Lotte et al., 2007).

Log Band-power

Another feature extraction method that is, for example, employed by Pfurtscheller and Neuper (2001) is the logarithm of the band power.

Band-power hereby refers to the power of the signal in a given frequency band. The power of a time varying signal $f(t)$ is defined as

$$\lim_{R\to\infty} \int_{-\frac{R}{2}}^{\frac{R}{2}} f(t)^2 dt$$

Hence, it is the average squared mean deviation from zero.

One method to estimate the band power is to first bandpass-filter the data and then to calculate the power using the variance (Pfurtscheller and Neuper, 2001). This results in the following feature per channel

$$x_c = \log(\mathrm{var}(E(c)))$$

where $E(c)$ refers to all time points from channel c.

Common Spatial Patterns

The original *Common Spatial Patterns* (CSP) algorithm was introduced for binary classification tasks. The goal of CSP is to find a transformation matrix that transforms each trial such that the variances of the resulting time series are optimal for discriminating the two classes (Ramoser et al., 2000). Recall that the variance of a bandpass-filtered signal is a good estimator of its power. Hence, CSP can be seen as a more advanced method then the log band-power to extract power differences.

More formally, CSP seeks $W \in \mathbb{R}^{C \times C}$ such that $Z = WE$ has high variance in the first rows for trials from class y_1 and low variance for trials from class y_2. Analogous to that, the last rows of Z should contain high variance for trials from class y_2 and low variance for trials from class y_1. This goal is archived by the simultaneous diagonalization of two covariance matrices (Fukunaga, 1990).

The CSP algorithm assumes that each channel in each trial has zero mean, $\text{mean}(E(c)) = 0, \forall c \in \{1, .., C\}$. The normalized spatial covariance matrix of each trial is then

$$\Sigma = \frac{EE^T}{trace(EE^T)} \tag{2.2.1}$$

For both classes the mean of their per trial covariance matrices is calculated, resulting in covariance matrices Σ_{y_1} and Σ_{y_2}.

The composite covariance matrix is obtained by

$$\Sigma_{\text{co}} = \Sigma_{y_1} + \Sigma_{y_2}$$

Because Σ_{co} is non-singular and symmetric, Σ_{co} can be decomposed, by an eigenvalue decomposition, into

$$\Sigma_{\text{co}} = Q\lambda Q^T \tag{2.2.2}$$

where λ is a diagonal matrix and contains the eigenvalues and Q contains the eigenvectors. Based on that equation, the whitening transformation matrix for Σ_{co} can be calculated

$$P = \sqrt{\lambda^{-1}} Q^T$$

The whitening transformation matrix fulfills the following property

$$P \Sigma_{\text{co}} P^T = I$$

Fukunaga (1990) showed that if Σ_{y_1} and Σ_{y_2} are transformed by P

$$\begin{aligned} S_{y_1} &= P \Sigma_{y_1} P^T \\ S_{y_2} &= P \Sigma_{y_2} P^T \end{aligned}$$

S_{y_1} and S_{y_2} share the same eigenvectors, and the sum of their corresponding eigenvalues is one. More formally, if S_{y_1} is decomposed to

$$S_{y_1} = B \lambda_{y_1} B^T$$

S_{y_2} is diagonalized by

$$S_{y_2} = B \lambda_{y_2} B^T$$

and $\lambda_{y_1} + \lambda_{y_2} = I$. What follows is that the eigenvector corresponding to the biggest eigenvalue in λ_{y_1} is the eigenvector that explains the most variance of the EEG trials

from class y_1 and the least variance of the trials from class y_2. If B is sorted by its eigenvalues λ_{y_1} in descending order, a transformed trial of class y_1

$$Z = (B^T P)^T E$$

has high variance for the first rows and low variance for the last rows. The opposite applies for trials from class y_2. Hence, the transformation fulfills the desired properties.

The last step of the CSP learning algorithm is to select rows from both ends of $(B^T P)^T$. Usually an equal number of rows is selected from both sides of the matrix. So the final transformation matrix is

$$W = \begin{pmatrix} 1_1 & 0 & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & \ddots & \ddots & & & & & & \vdots \\ \vdots & \ddots & 1_k & \ddots & & & & & \vdots \\ \vdots & & \ddots & 0 & \ddots & & & & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & & & \ddots & 0 & \ddots & & \vdots \\ \vdots & & & & & \ddots & 1_1 & \ddots & \vdots \\ \vdots & & & & & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & \dots & \dots & \dots & \dots & 0 & 1_k \end{pmatrix} (B^T P)^T$$

where the number of ones on each sides of the diagonal is even and must be selected as the hyper-parameter k. Note that $(B^T P)^T$ are the model parameters θ, which are learned from the data.

The logarithm of the variance of the resulting k time series is usually employed as feature. CSP is currently one of the most used feature extraction methods for the extraction of power differences.

Permutation Entropy

The *Permutation Entropy* (PE) was introduced by Bandt and Pompe (2002) as a complexity measure for time series. The overall idea is to reduce a time series to an order pattern between m neighbors. m is called embedding dimension in the remainder.

Definition 14. Let $\{f(t)\}_{t=1..T}$ be a time series. The permutation distribution of embedding dimension m is then defined as

$$p_m(\pi) = \frac{|\{t : 0 \leq t \leq T - m, (x(t+1), \dots, x(t+m)) \text{ has type } \pi\}|}{T - m + 1} \tag{2.2.3}$$

where π represents one permutation of the $m!$ possible permutations.

The value $p_m(\pi)$ represents the probability of the occurrence of the ordering that is represented by the permutation π. I clarify the definition of the permutation distribution with a simple example.

Example 15. Assume that the embedding dimension m is 2, and the time series for that we want to calculate the permutation distribution is

$$(1, 2, 20, 30, 2, 1)$$

There are two possible orderings of two unequal elements $x(t)$ and $x(t+1)$. Either $x(t)$ is greater than $x(t+1)$ or $x(t+1)$ is greater than $x(t)$. The first ordering can be represented by the permutation 10 and the second ordering by 01. In the example time series there are three occurrences of $x(t) < x(t+1)$ and two occurrences of $x(t) > x(t+1)$. Therefore, the permutation distribution with embedding size 2 for that time series is

$$\begin{aligned} p_2(10) &= 3/5 \\ p_2(01) &= 2/5 \end{aligned}$$

The permutation distribution itself could be used as feature. However, Bandt and Pompe (2002) suggest to aggregate it to the *Permutation Entropy* (PE).

Definition 16. The PE of embedding dimension $m \geq 2$ is defined as the Shannon entropy of the permutation distribution of embedding dimension m,

$$H(m) = -\sum p_m(\pi) \log p(\pi)$$

Example 17. The permutation entropy of the permutation distribution from Example 15 is

$$-(\frac{3}{5}\log(\frac{3}{5}) + \frac{2}{5}\log(\frac{2}{5})) = 0.971$$

When using the PE as feature extraction method for EEG classification, the PE is calculated separately for each channel. This results in a feature vector of the form $x \in \mathbb{R}^C$ for each trial. There is one hyper-parameter that has to be chosen by the designer, the embedding dimension m. Recently, Brandmaier (2012) described a heuristic for choosing the embedding dimension automatically.

The Permutation Entropy was introduced as a feature extraction method for BCIs (Nicolaou and Georgiou, 2010). Furthermore, Brandmaier (2012) demonstrated that divergence measures based on the permutation distribution perform well in clustering EEG trials.

2.2.4. Classification Methods

Ledoit's Regularized Linear Discriminant Analysis

Ledoit's Regularized Linear Discriminant Analysis (LRLDA) is based on *Linear Discriminant Analysis* (LDA). The idea behind LDA is to adjust a normal distribution for each class. A new trial is assigned to the class for that the a-posteriori likelihood is the highest (von Oertzen, 2011).

Assume that for every class the class conditional probability distribution $P_l(X) = P(X|Y = y_l)$ is known and normal, i.e., $P_l \sim \mathcal{N}(\mu_l, \Sigma_l)$. Additionally, the a-priori likelihoods for every class $P(Y = y_l)$ are the same $P(Y = y) = 1/L$. Furthermore, the

variances and covariances are the same for each class $\Sigma_l = \Sigma, \forall l \in \{1, \ldots, L\}$. Under these assumptions, the classifier that classifies a new sample to the class for that the a-posteriori likelihood is the highest is called LDA.

$$\mathrm{LDA}(x) = y_l$$

where

$$l = \arg\max_{l=1}^{L}(\mathcal{N}(x; \mu_l, \Sigma)) \tag{2.2.4}$$

where $\mathcal{N}(x; \mu, \Sigma)$ is the likelihood of x under the multivariate normal distribution with mean μ and variance σ. Equation 2.2.4 can be simplified to

$$\begin{aligned}
\arg\max_{l=1}^{L}(\mathcal{N}(x; \mu_l, \Sigma)) &= \arg\max_{l=1}^{L}(\log(\mathcal{N}(x; \mu_l, \Sigma))) \\
&= \arg\max_{l=1}^{L}(\log((2\pi)^{-\frac{r}{2}}|\Sigma|^{-\frac{1}{2}}e^{-\frac{1}{2}(x-\mu_l)^T\Sigma^{-1}(x-\mu_l)})) \\
&= \arg\max_{l=1}^{L}(-\frac{1}{2}(x-\mu_l)^T\Sigma^{-1}(x-\mu_l))
\end{aligned}$$

Furthermore,

$$-\frac{1}{2}(x-\mu_l)^T\Sigma^{-1}(x-\mu_l) = x^T\Sigma^{-1}\mu_l - \frac{1}{2}\mu_l^T\Sigma^{-1}\mu_l - \frac{1}{2}x^T\Sigma^{-1}x$$

For simplicity I will continue the treatment for the binary classification task, with $y_1 = 1$ and $y_2 = 2$.

$$\begin{aligned}
\mathrm{LDA}(x) &= 1 \\
\Leftrightarrow x^T\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 - \frac{1}{2}x^T\Sigma^{-1}x &\geq x^T\Sigma^{-1}\mu_2 - \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 - \frac{1}{2}x^T\Sigma^{-1}x \\
\Leftrightarrow x^T\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 &\geq x^T\Sigma^{-1}\mu_2 - \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2
\end{aligned}$$

$$\Leftrightarrow (\Sigma^{-1}(\mu_1 - \mu_2))^T x - \frac{1}{2}(\mu_1 + \mu_2)\Sigma^{-1}(\mu_1 - \mu_2)^T \geq 0 \tag{2.2.5}$$

Equation 2.2.5 is of the form

$$w^T x + c \geq 0 \tag{2.2.6}$$

with $w = \Sigma^{-1}(\mu_1 - \mu_2)$ and $c = -\frac{1}{2}(\mu_1 + \mu_2)\Sigma^{-1}(\mu_1 - \mu_2)^T$. That means that the decision surface learned by LDA between two classes is a hyperplane. If a classifier fulfills this property, it is called *linear classifier*.

In practice the means $\mu_1, \ldots, \mu_L$ and the covariance matrix Σ^{-1} are usually not known. These values have to be estimated from the data set by the learning algorithm I_{LDA}. The estimation of the means is straightforward

$$\hat{\mu}_l = \frac{1}{|D_l|}\sum_{(x,y)\in D_i} x$$

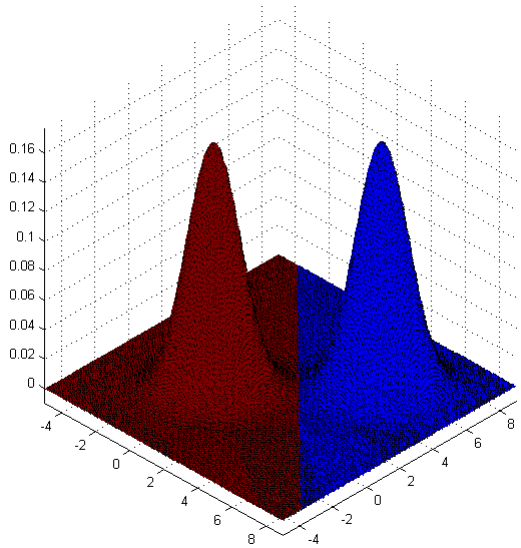


Figure 2.2.2.: Illustration of the separating model that is learned by LDA. For each class a multivariate normal distributions is estimated. A novel sample is assigned to the class with the highest a-posteriori likelihood. The figure displays the a-posteriori likelihood for a sample to be in class y_1 or y_2 for different values. The different colors illustrate the regions in which a point is classified as class y_1 (red) or class y_2 (blue). The decision surface between the two classes is a hyperplane.

where $D_l := \{(x, y) \in D : y = y_l\}$. LDA is known to be relatively robust to the violation of the assumption that the per-class covariances matrices are the same. Thus, the standard LDA learning algorithm estimates the common covariance matrix by

$$\hat{\Sigma} = \frac{1}{2}(\hat{\Sigma}_1 + \hat{\Sigma}_2) \tag{2.2.7}$$

where $\hat{\Sigma}_l$ is the empirical covariance matrix of class y_l

$$\hat{\Sigma}_l = \frac{1}{|D_l|} \sum_{(x,y)\in D_l} (x - \hat{\mu}_l)(x - \hat{\mu}_l)^T \tag{2.2.8}$$

When the covariance matrix is estimated separately for each class, the decision surface becomes quadratic. Consequently, the corresponding classification method is called Quadratic Discriminant Analysis.

Under the usual conditions in statistical analysis, the number of samples per class $|D_l|$ is large compared to the dimensionality r of the feature space, the empirical covariance matrix is an unbiased estimate of the true covariance matrix. But if $|D_l|$ is not significantly larger than the dimensionality of the feature space r, it is known that the empirical covariance is systematically biased. Large Eigenvalues of Σ are estimated too large and small eigenvalues are estimated too small (Friedman, 1989). An approach to correct for this systematic bias is to replace $\hat{\Sigma}_l$ by

$$\hat{\Sigma}_l^* = (1 - \gamma)\hat{\Sigma}_l + \gamma \frac{tr(\hat{\Sigma}_l)}{r} I \tag{2.2.9}$$

where $tr(A)$ refers to the trace of matrix A, I is the identity matrix and $\gamma \in [0, 1]$ is a hyper-parameter. Equation 2.2.9 regularizes $\hat{\Sigma}_l$ towards the multiple of the identity matrix. Therefore, larger eigenvalues are decreased and smaller eigenvalues are increased (Friedman, 1989); correcting for the systematic bias.

When estimating the covariance matrix according to Equation 2.2.9, the resulting classification method is called *Regularized Linear Discriminant Analysis* (RLDA). An open question was how to choose the hyper-parameter γ. Ledoit and Wolf (2004) presented an analytic solution for choosing the optimal γ. Their method estimates γ such that $||\hat{\Sigma}_l^* - \Sigma_l||_r^2$ is minimized, where $||A||_r$ is defined as $||A||_r = \sqrt{tr(AA^T)/r}$. $||A||_r$ is equivalent to the *Frobenius norm* divided by r. Remember that r denotes the dimensionality of the feature space X. According to their results, the optimal γ is

$$\gamma^* = \frac{b^2}{d^2} \tag{2.2.10}$$

where

$$d^2 = ||\hat{\Sigma}_l - \frac{tr(\hat{\Sigma}_l)}{r} I||_r^2$$

and

$$b^2 = \min\left(\frac{1}{|D_l|^2} \sum_{(x,y)\in D_l} ||(x - \hat{\mu}_l)(x - \hat{\mu}_l)^T - \hat{\Sigma}_l||_r^2, d^2 \right)$$

d^2 describes the deviation of the sample covariance from the scaled identity matrix. b^2 describes the deviation of the per-trial covariance matrices $(x - \hat{\mu}_l)^T(x - \hat{\mu}_l)$ from their mean Σ_l. The minimum operator ensures that the shrinking parameter γ^* stays below 1.

In the literature the resulting classification method is also called RLDA when estimating γ according to 2.2.10. To distinguish this method from the classical RLDA method, for which γ has to be chosen by the designer, I will call this method *Ledoit's Regularized Linear Discriminant Analysis* (LRLDA).

LRLDA is the classification method that leading BCI groups use and advocate (Blankertz et al., 2011).

Support Vector Machines

*Support Vector Machine*s (SVMs) were invented by Cortes and Vapnik (1995). My description of SVMs is inspired by Ng (2011).

Without loss of generality, $Y = \{-1, 1\}$. In analogy to Equation 2.2.6, I define a linear classifier

$$\Psi_{\text{lin}}(x; w, b) = \begin{cases} 1 & \text{if, } w^t x + b \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{2.2.11}$$

The basic form of SVMs assumes that the training data set D is linearly separable, i.e., there exists a linear classifier that perfectly separates the two classes

$$\exists w, b : \forall (x_i, y_i) \in D \begin{cases} w^t x_i + b > 0 & \text{if } y_i = 1 \\ w^t x_i + b < 0 & \text{if } y_i = -1 \end{cases}$$

Notice that this condition is equivalent to

$$\exists w, b : \forall (x_i, y_i) \in D \; y_i(w^t x_i + b) > 0 \tag{2.2.12}$$

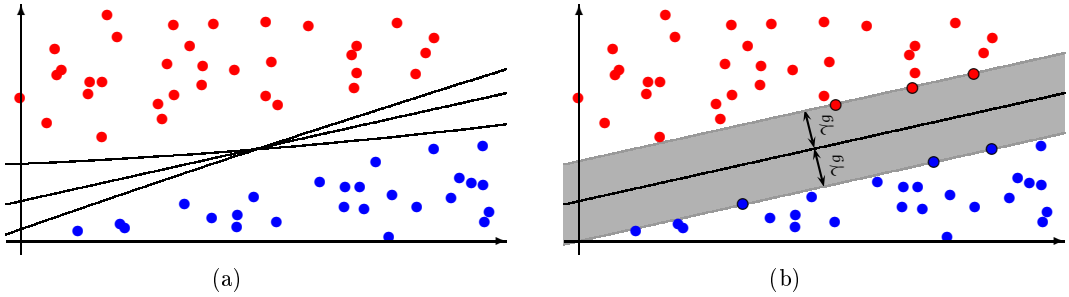
In general, for a linearly separable data set there are many choices for w and b that satisfy Equation 2.2.11. The idea behind SVMs is to choose w and b such that the geometric margin is maximized. The geometric margin is the smallest distance between the hyperplane described by w and b and any point in the training set D. The motivation behind that is that maximizing the margin should be a good strategy to maximize the accuracy as it decreases the risk of a new point to be at the wrong side of the decision plane.

What follows is a formalization of this idea. The geometric margin is defined as

$$\gamma_g(D; w, b) = \min_{i=1}^{N} \frac{y_i(w^T x_i + b)}{||w||}$$

Hence, the optimization problem of the SVM is: Find $w, b, \hat{\gamma}_f$ such that the geometric margin is maximized

$$\begin{aligned} &\arg\max_{\hat{\gamma}_f, w, b} \frac{\hat{\gamma}_f}{||w||} \\ \text{subject to constraints} \quad & y_i(w^T x_i + b) \geq \hat{\gamma}_f, \forall i \in \{1, \ldots, N\} \end{aligned} \tag{2.2.13}$$



(a) (b)

Figure 2.2.3.: (a) Multiple decision planes that perfectly separate the data set. (b) The decision plane that maximizes the geometric margin γ_g. The marked points are the support vectors. Adapted with permission from Schilling (2012).

Unfortunately, this optimization problem is hard to solve as $\frac{\hat{\gamma}_f}{||w||}$ is non-convex. Thus, the problem has to be transformed into an easier, equivalent optimization problem.

The value $\hat{\gamma}_f$ is often called functional margin. For given w and b it can be expressed as a function, which additionally depends on the training data set D.

$$\gamma_f(D; w, b) = \min_{i=1}^{N} y_i(w^T x_i + b)$$

where $\gamma_g = \gamma_f / ||w||$. Notice that

$$\delta(w^T x + b) = \delta w^T x + \delta b, \forall \delta \in \mathbb{R}^+$$

Therefore,

$$\forall \varepsilon \in \mathbb{R}^+ / \infty \exists \delta \in \mathbb{R}^+ / \infty : \gamma_f(D; \delta w, \delta b) > \varepsilon$$

is true for every w, b for that Equation 2.2.12 holds. Furthermore, scaling w, b like that does not change the classification function 2.2.11 as it only depends on the sign. Thus, for every linear separating classifier an arbitrary functional margin can be achieved by simply scaling w and b without changing the geometric margin or the actual classification function. Therefore, by setting $\hat{\gamma}_f$ to 1 the space in which the classifiers are searched is not decreased. Moreover,

$$\arg\min_w \frac{1}{||w||} = \arg\max_w ||w|| = \arg\max_w \frac{1}{2}||w||^2$$

Thus, the optimization problem 2.2.13 can be rephrased as

$$\arg\max_{w,b} \frac{1}{2}||w||^2 \tag{2.2.14}$$

$$\text{subject to constraints} \quad y_i(w^t x_i + b) \geq 1, \forall i \in \{1, \ldots, N\} \tag{2.2.15}$$

This optimization problem can be solved using quadratic programming. Let w_*, b_* the optimal values for w and b. Each sample (x_i, y_i) for that

$$y_i(w_*^T x_i + b_*) = 1$$

is then called support vector. The values of w_*, b_* only depend on these samples. They support the decision plane.

The assumption that the training data set is linearly separable makes this classification method applicable to only a small subset of all existing data sets. Cortes and Vapnik (1995) enhanced their method to make it applicable to arbitrary data sets. For each violation of the original constraints 2.2.15 a cost term is added to the objective function 2.2.14.

$$\begin{aligned} & \arg\min_{w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to constraints} \quad & y_i(w^t x_i + b) \geq 1 - \xi_i,\ \forall i \in \{1,\ldots,N\} \\ & \xi_i \geq 0,\ \forall i \in \{1,\ldots,N\} \end{aligned}$$

where $C \geq 0$ is a hyper-parameter. Each trial x_i that lies inside the margin gets assigned cost ξ_i. Notice that if $\xi_i > 1$, x_i is on the wrong side of the decision plane.

In this section I introduced linear SVMs, which induce linear classifiers. Linear SVMs can be extended to nonlinear classifiers using the so called *kernel trick* (Cortes and Vapnik, 1995). SVMs have been successfully employed as classification method in various BCIs (Lotte et al., 2007).

k-Nearest Neighbors

The *k-Nearest Neighbor* (k-NN) classification method is one of the most simple classification methods. It assigns a new trial x to the class to that the simple majority of the k-nearest training trials belong to.

Let $d(x, y)$ be a distance function defined over the feature space X. For a novel trial x let $Q_k(x) \subseteq D$ be the k-nearest neighbors of x in the training data set D, that is

$$Q_k(x) = S \subseteq D : |S| = k \wedge \forall (x_i, y_i) \in S \nexists (x_j, y_j) \in D/S : d(x_j, x) < d(x_i, x)$$

The k-nearest neighbor classifier assigns then x to the class that gets the most votes

$$\text{kNN}(x; D, k) = \arg\max_{l=1}^{L} \sum_{(x_i,y_i)\in Q_k(x)} \delta(y_i, y_l)$$

where

$$\delta(a, b) = \begin{cases} 1 & \text{if, } a = b \\ 0 & \text{otherwise} \end{cases} \tag{2.2.16}$$

k-NN has been used as a classification method in multiple EEG classification setups (Lotte et al., 2007).

2.3. Combination of Classifiers

The term combination of classifiers refers to the process of combining multiple classifiers for the same problem to a new classifier, an *ensemble classifier*.

Definition 18. Let $\Psi_1, \ldots, \Psi_J$ be classifiers for one particular pattern recognition task. Let $r_{\theta,\tau}(\Psi_1(\mathbf{x}), \ldots, \Psi_J(\mathbf{x}), \mathbf{x}) = \Psi_{\text{ens}}$ be a rule that combines the outputs of the classifiers to a new classifier Ψ_{ens}. The classifiers $\Psi_1, \ldots, \Psi_J$ are then called *base-level classifiers*, the rule $r_{\theta,\tau}$ *combination rule*, and the classifier Ψ_{ens} *ensemble classifier*. Analogous to the feature extraction function, the combination rule has parameters θ that have to be learned by the corresponding learning algorithm and parameters τ that have to be specified by the designer. The combination of learning algorithm and combination rule is called *combiner*. A *Pattern Recognition System* (PRS) that employes an ensemble classifier is called *Multiple Classifier System* (MCS). The learning algorithms that induce the base-level classifiers are called *base-level learners*.

I will start the following treatment of the combination of classifiers with the introduction of a taxonomy. It will include a categorization of the different approaches to build different base-level classifiers and a classification of different types of combiners. After that, I will present a selection of combiners for the combination of labels. I will conclude this section with a treatment why and under what conditions an ensemble classifier is more accurate than the most accurate base-level classifier.

2.3.1. Taxonomy

When creating an ensemble classifier, one has to fulfill two tasks: The creation of accurate and diverse base-level classifiers and the appropriate combination of the base-level classifier.

Kuncheva (2004, chapter 3) identifies three approaches that are used to generate diverse base-level classifiers:

1. The employment of different classification methods
2. The employment of different feature extraction methods
3. The employment of different subsets of the data set as input for the learning algorithm

These three methods can be arbitrarily combined.

Kuncheva (2004, chapter 3) classifies the different combiners based on two properties: The type of the input on that they operate and if they are trainable or nontrainable.

She distinguishes between three types of base-level classifier outputs and, hence, combiner inputs.

- Type 1 (The Abstract level): Each base-level classifier returns a label for each sample. There is no information about the certainty of the classification.
- Type 2 (The Rank level): The output of each base-level classifier is an ordered subset of Y. It is ordered by a-posteriori likelihood.
- Type 3 (The Measurement level). Every base-level classifier produces a L-dimensional vector $[s_1, \ldots, s_L]$, where s_l represents the likelihood that the sample x belongs to class y_l.

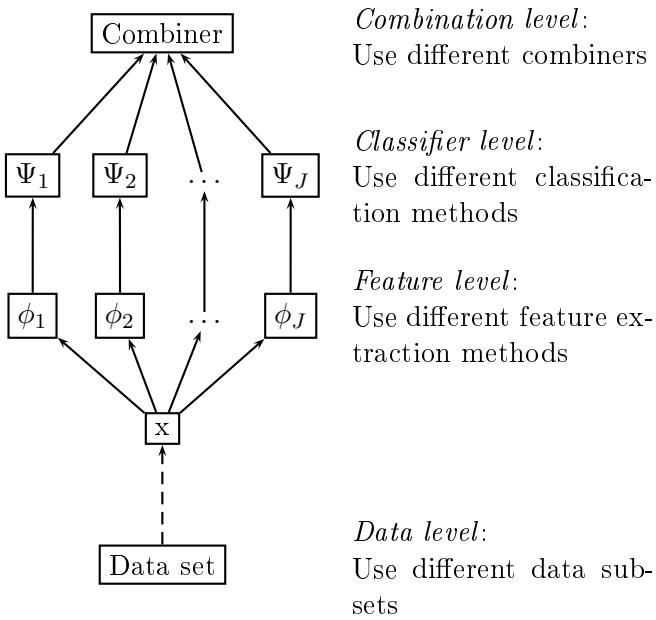


Figure 2.3.1.: Approaches to build ensemble classifiers. Adapted from Kuncheva (2004, p 105).

Nontrainable combiners combine the output of the base-level classifiers using fixed rules.

Definition 19. A *nontrainable combiner* consist of a combination rule comb that combines the outputs of the base-level classifiers. $\Psi_{\text{ens}} = r_{\theta,\tau}(\Psi_1(\mathbf{x}), \ldots, \Psi_J(\mathbf{x}))$. The combination rule is static and independent of the data set. Hence, $\theta = \emptyset$.

Trainable combiners, in contrast, use learning algorithms to induce the combination rule from the data. The trainable combiners are further distinguished in implicit and explicit trainable combiners. The learning algorithm of implicit trainable combiners induces one combination rule for all samples, i. e., the combination rule is independent of the to-be-classified sample x. Explicit trainable combiners, in contrast, induce a combination rule that can potentially be different for every sample x.

Definition 20. A *trainable combiner* consists of a learning algorithm that induces a combination rule based on a data set $I_{r_{?,\tau}}(D) = r_{\theta,\tau}$. If the combination rule has the following signature $r_{\theta,\tau}(\Psi_1(x), \ldots, \Psi_J(x))$, hence, it does not dependent on x itself, the combiner is called *implicit trainable*. If the combination rule directly depends on x, $r_{\theta,\tau}(\Psi_1(x), \ldots, \Psi_J(x), \mathbf{x})$, it is called *explicit trainable*.

2.3.2. Abstract Level Combiners

In this section I will review nontrainable and implicit trainable combiners that combine the outputs at the abstract level. This is called combination of labels in the remainder of this thesis.

The situation is as follows: A variety of base-level learners $I_{\Psi_1}(D_{\text{train}}), \ldots, I_{\Psi_J}(D_{\text{train}})$ have been trained on the data set D_{train} and produced base-level classifiers $\{\Psi_1, \ldots, \Psi_J\} =: B$. The task of the combiners is to build an ensemble classifier based on the base-level classifiers. For that they may employ a separate data set D_{comb}, sampled independently from the same distribution as D_{train}.

2.3.2.1. Majority Voting

Majority Voting (MV) is perhaps the most simple combiner. It is a nontrainable combiner. Therefore, it does not consume D_{comb} to generate the combination rule. The combination rule is fixed and defined as follows.

Definition 21. The combination rule of majority voting is defined as

$$\text{mv}(\Psi_1(x), \ldots, \Psi_J(x)) = \arg\max_{y_l \in Y} \sum_{j=1}^{J} \delta(\Psi_j(x), y_l)$$

where δ is defined as in Equation 2.2.16.

MV assigns $\mathbf{x}$ to the class y_l for that most of the base-level classifiers Ψ_j voted. Ties are resolved arbitrarily. Despite of its simplicity or maybe because of its simplicity, MV is one of the most used combiners.

When certain assumptions are made about the base-level classifiers, the accuracy of the ensemble classifier created by the MV rule can be calculated.

p	$J = 7$	$J = 15$	$J = 51$	$J = 101$	$J = 301$	$J = 500$
0.45	0.3917	0.3465	0.2359	0.1562	0.0409	0.0124
0.55	0.6083	0.6535	0.7641	0.8438	0.9591	0.9876

Table 2.2.: Accuracy of the ensemble classifier built by MV for different numbers of base-level classifiers, under the assumption of independence. p denotes the accuracy of the base-level classifiers.

Theorem 22. *Let the number of base-level classifiers J be odd and the accuracy of every base-level classifier be p. Furthermore, let the outputs of the base-level classifiers be independent. That means that for each subset $\{\Psi_1, \ldots, \Psi_K\} \subseteq B$ the joint probability $P(\Psi_1 = y_1, \ldots, \Psi_K = y_K)$ equals $\prod_{k=1}^{K} P(\Psi_k = y_k)$. Then the accuracy of the ensemble classifier built by employing MV as combiner is*

$$p_{mv} = \sum_{o=\frac{J+1}{2}}^{J} \binom{J}{o} p^o (1-p)^{J-o}$$

Proof. The ensemble classifier built by MV classifies a sample $\mathbf{x}$ correctly if at least $\frac{J+1}{2}$ base-level classifiers Ψ_j classify $\mathbf{x}$ correctly. Hence, if we assume that the accuracy of each base-level classifier is p, the accuracy of the ensemble classifier built by MV is as claimed. □

The following results require the same assumptions as Theorem 22. Table 2.2 shows how the accuracy of the ensemble classifier built by MV changes when the number of base-level classifier increases for $p = 0.45$ and $p = 0.55$. It can, furthermore, be shown that

$$\lim_{J\to\infty} p_{\text{mv}} = \begin{cases} 1 & \text{if, } p > 0.5 \\ 0 & \text{if, } p < 0.5 \end{cases}$$

Additionally, if $p > 0.5$ ($p < 0.5$), p_{mv} is monotonically increasing (decreasing) as J expands. This proof can also be extended to the case where the accuracy of the base-level classifiers are unequal. Indeed, the only necessary condition is that they are symmetrically distributed with a mean above 0.5 (see Kuncheva, 2004, p 114 and references therein) Hence, the intuition that an ensemble classifier boosts the accuracy if the base-level classifiers are accurate and diverse is supported.

2.3.2.2. Weighted Majority Voting

Example 23. Assume that $J = 3$, $L = 2$, $p_1 = 0.4$, $p_2 = 0.4$, $p_3 = 0.65$, and independence as in Theorem 22, where p_j refers to the accuracy of base-level classifier Ψ_j. The accuracy of the ensemble classifier generated by MV is then

$$\begin{aligned} p_{\text{mv}} &= p_1p_2p_3 + (1-p_1)p_2p_3 + p_1(1-p_2)p_3 + p_1p_2(1-p_3) \\ &= 0.4 \cdot 0.4 \cdot 0.65 + 0.6 \cdot 0.4 \cdot 0.65 + 0.4 \cdot 0.6 \cdot 0.65 + 0.4 \cdot 0.4 \cdot 0.35 \\ &= 0.472 \end{aligned}$$

As you may confirm, this is smaller than the accuracy of the most accurate base-level classifier Ψ_3.

In this section I will introduce a combiner that in the situation of independent base-level classifiers maximizes the ensemble accuracy. This combiner is called *Weighted Majority Voting* (WMV). I will show that, in contrast to MV, the ensemble classifier built by WMV leads to a more accurate classifier than the most accurate base-level classifier when applied to the previous example

First, I will establish *Weighted Voting* (WV) in general, then, I will show how the optimal weights are obtained by WMV.

Definition 24. A WV rule is of the following form

$$\mathrm{wv}(\Psi_1(x), \ldots, \Psi_J(x)) = \arg\max_{y_l \in Y} \sum_{j=1}^{J} w_j \delta(\Psi_j(x), y_l)$$

where $w_1, \ldots, w_J \in \mathbb{R}$ are weights for the corresponding base-level classifiers and δ is defined as in Equation 2.2.16.

Example 25. Let the base-level classifiers be as in Example 23. Let the weights for the WV rule be $w_1 = -0.4055$,$w_2 = -0.4055$ and $w_3 = 0.6190$. Then the accuracy of the ensemble classifier build by WV is

$$\begin{aligned} p_{\mathrm{wv}} &= (1-p_1)\cdot(1-p_2)\cdot(1-p_3) + (1-p_1)\cdot(1-p_2)\cdot p_3 + p_1(1-p_2)p_3 + (1-p_1)p_2p_3 \\ &= 0.672 \end{aligned}$$

Proof. Notice that $\delta(\Psi_j(x), y_l)$ is 1 for exactly one $l \in \{1, \ldots, L\}$ as every classifier predicts exactly one label because in the example $L = 2$, $\delta(\Psi_j(x), y_1) = 0 \Leftrightarrow \delta(\Psi_j(x), y_2) = 1$. Hence, if $\delta(\Psi_j(x), y_1)$ is known, $\delta(\Psi_j(x), y_2)$ is also known. Thus, if a base-level classifiers predicts the correct (wrong) class, its weight influences only the sum of the correct (wrong) class. Let y_c be the correct class and y_w be the wrong class. An ensemble classifier built by a WV rule makes the correct decision if

$$\sum_{j=1}^{J} w_j \delta(\Psi_j(x), y_c) > \sum_{j=1}^{J} w_j \delta(\Psi_j(x), y_w)$$

This is exactly the case if the voting behavior is: 000 or 001 or 101 or 011. Where a 1(0) at the jth place means that the jth base-level classifier classifies a sample correctly (incorrectly). □

Until now, we have seen that WV can drastically improve the ensemble accuracy compared to MV. The interesting question is how to choose the weights.

Theorem 26. *Consider a set of J independent classifiers that are combined using the weighted voting combination rule. Furthermore the a-priori probabilities for all classes*

are the same. The accuracy of the resulting ensemble gets maximized by assigning each classifier Ψ_j the weight

$$w_j = \log \frac{p_j}{1 - p_j}$$

Proof. See (Kuncheva, 2004, pp. 124) □

When the WV rule is employed and the weights are set as in the above theorem, the resulting combiner is called *Weighted Majority Voting* (WMV). Shapley and Grofman (1984) even showed, for binary decisions, that if the a-priori probabilities for both classes are the same, under the independence assumption, WMV is the combiner, out of all possible combiners, that maximizes the accuracy of the ensemble classifier. If the a-priori likelihoods are not the same, they have to be taken into account for the decision function. This leads to the general form of WMV.

$$\mathrm{wmv}(\Psi_1(x), \ldots, \Psi_J(x)) = \arg\max_{y_l \in Y} [P(Y = y_l) + \sum \log_{j=1}^{J} \frac{p_j}{1 - p_j} \delta(\Psi_j(x), y_l))] \quad (2.3.1)$$

Note that p_j for every classifier Ψ_j and $P(Y = y_l)$ have to be estimated using D_{comb}.

2.3.2.3. Adaptive Boosting

Adaptive Boosting (AB) is a boosting algorithm invented by Freund and Schapire (1997). It is an application of their algorithm for the on-line allocation problem. According to Freund and Schapire (1997, p 120), boosting refers to the "general problem of producing a very accurate prediction rule by combining rough and moderately inaccurate rules of thumb". The original algorithm generates arbitrarily many base-level classifiers by training a weak learner on different subsamples from the data set D.

However, AB can also be applied to the situation in which a predefined set of base-level classifiers has to be combined, as described at the beginning of this section. The AB algorithm for that situation, as described by Rojas (2009), is shown as Algorithm 2.1 and called *fixed Adaptive Boosting* (fAB) in the remainder.

fAB is also a WV combiner. WMV and fAB only differ in the way they compute the weights for the base-level classifiers. While for WMV the weight of each base-level classifier Ψ_j only depends on the performance of itself, fAB also takes into account the performance of other base-level classifier in the set. This is done by iteratively adding base-level classifier to the ensemble and using an importance for each sample. After adding a base-level classifier Ψ to the ensemble, the importance of each sample that Ψ classifies wrong is increased and the importance of samples that Ψ classifies correct is decreased. The next classifier that gets added to the set is the one with the lowest error, in respect to the importance of the samples. Thus, while WMV is optimal if the base-level classifiers are dependent, fAB potentially produces a more accurate ensemble classifier if the independence assumption is violated.

Algorithm 2.1 Pseudocode for the learning algorithm of the fAB combiner

Input:

- data set $D_{\text{comb}} = \{(x_1, y_1), ..., (x_N, y_N)\}$
- base-level classifiers $\{\Psi_1, \ldots, \Psi_J\}$

Procedure

Initialize $\forall i \in \{1, \ldots, N\}\, W_i^1 = 1$, $sel^1 = \emptyset$,
for $m = 1$ to J do

1. Select the base classifier Ψ_j that was not already selected in the iterations $1, \ldots, m-1$ with the lowest weighted error

$$\begin{aligned} s_m &= \arg \min_{j=\{1,\ldots,J\}/sel^m} \left(\sum_{i=1}^{N} L_{01}(x_i, y_i, \Psi_j) W_i^m\right) \\ &\quad \text{where } L_{01} \text{ is the zero-one loss function} \\ \text{sel}^{m+1} &= \text{sel}^m \cup s_m \end{aligned}$$

2. Calculate the relative error of the selected base-level classifier

$$\text{err}_m = \frac{\sum_{i=1}^{N} L_{01}(x_i, y_i, \Psi_{s_m}) W_i^m}{\sum_{i=1}^{N} W_i^m} \in [0, 1]$$

3. Set the weight w_{s_m} for the selected base-level classifier Ψ_{s_m} to

$$w_{s_m} = \frac{1}{2} \log \left(\frac{1 - \text{err}_m}{\text{err}_m} \right) \in \mathbb{R}$$

4. Update the importance of the samples for the next step

$$W_i^{m+1} = W_i^m \cdot \begin{cases} e^{w_{sm}} & \text{if } \Psi_{s_m}(x_i) \neq y_i \\ e^{-w_{sm}} & \text{if } \Psi_{s_m}(x_i) = y_i \end{cases}$$

end

Output: the combination rule

$$\text{ab}(\Psi_1(x), \ldots, \Psi_J(x)) = \arg \max_{y_l \in Y} \sum_{j=1}^{J} w_j \delta(\Psi_j(x), y_l)$$

		predicted class	
		Hippo	Giraffe
actual class	Hippo	10	13
	Giraffe	8	5

Table 2.3.: Example for a confusion matrix. In this case the corresponding classifier classifies 10 of the 23 hippos correctly. From the 13 giraffes 5 are classified correctly.

2.3.2.4. Bayes Combination

Under the assumption of conditional independence between the base-level classifiers, the probability of having observed a sample of class y_l after having seen x is:

$$P(y_l|x) = \frac{P(Y = y_l) \prod_{j=1}^{J} P(\Psi_j = \Psi_j(x)|Y = y_l)}{P(\Psi_1 = \Psi_1(x) \wedge, \ldots, \wedge \Psi_J = \Psi_J(x))} \tag{2.3.2}$$

The denominator is independent of the candidate class y_l, its purpose is only to scale $P(y_l|x)$ such that it fulfills the conditions of a probability measure. Hence, for classification only the nominator is needed. Therefore, the support for class y_l is

$$\sup_{y_l}(x) = P(Y = y_l) \prod_{j=1}^{J} P(\Psi_j = \Psi_j(x)|Y = y_l) \tag{2.3.3}$$

It seems reasonable to assign a new sample to the class with the highest support. This leads to *Bayes Combination* (BC).

$$\mathrm{bc}(\Psi_1(x), \ldots, \Psi_J(x)) = \arg\max_{y_l \in Y}(\sup_{y_l}) \tag{2.3.4}$$

Note that not all of the needed probabilities are known, but they can easily be estimated employing D_{comb}.

The probabilities needed for BC can be estimated by the confusion matrix.

Definition 27. Let D be a data set and Ψ a classifier. Each entry $\mathrm{cf}_{l,k}(\Psi, D)$ of the confusion matrix $\mathrm{CF}(\Psi, D)$ is then defined as

$$\mathrm{cf}_{l,k}(\Psi, D) = |\{(x, y) \in D : y = y_l \wedge \Psi(x) = y_k\}|$$

The entry $\mathrm{cf}_{l,k}$ corresponds to the number of the samples (x, y) in D with label y_l that were labeled with the label y_k by the classifier Ψ. For a perfect classifier all off diagonal elements of the confusion matrix are zero. An example for a confusion matrix can be seen in Table 2.3.

The confusion matrix for all base-level classifiers has to estimated. I will call CF^j the confusion matrix of base-level classifier Ψ_j, with entries $\mathrm{cf}^j_{l,k}$. Let y_{k_j} be the label predicted by base-level classifier Ψ_j, $\Psi_j(x) = y_{k_j}$; $\forall j \in \{1, \ldots, J\} k_j \in \{1, \ldots, L\}$. Let

N_l=$|\{(x,y) \in D : y = y_l\}|$ denote the number of samples $(x,y) \in D$ with label y_l. N_l/N can then be used as an estimate for $P(Y = y_l)$ and $\mathrm{cf}^j_{l,k_j}/N_l$ as an estimate for $P(\Psi_j = \Psi_j(x)|Y = y_l)$. Thus, the support for each class (see Equation 2.3.3) can be estimated as

$$\hat{\mathrm{sup}}_{y_l}(x) = \frac{N_l}{N}\prod_{j=1}^{J}(\mathrm{cf}^j_{l,k_j}/N_l) = \frac{1}{N_l^{J-1}N}\prod_{j=1}^{J}\mathrm{cf}^j_{l,k_j}$$

Because N is independent of the candidate class, this can be further simplified to

$$\frac{1}{N_l^{J-1}}\prod_{j=1}^{J}\mathrm{cf}^j_{l,k_j}$$

Note that if $\exists j, k_j : \mathrm{cf}^j_{l,k_j} = 0$, the support of the whole class y_l is zero. Because the probabilities are estimated and, hence, a probability that is estimated as 0 may not be 0, this is an undesired behavior. Therefore, (Kuncheva, 2004, p 127) suggests a different method to calculate the support, which she adapted from the work of Titterington et al. (1981). The resulting estimator of the support is

$$\hat{\mathrm{sup}}^*_{y_l}(x) = \left(\prod_{j=1}^{J}\frac{\mathrm{cf}^j_{l,k_j} + 1/J}{N_l + 1}\right)^B \tag{2.3.5}$$

where B is a hyper-parameter.

2.3.2.5. Stacking

Stacking refers to the procedure of applying a classification method to to the outputs of the base-level classifiers. The classifier that works on the output of the base-level classifiers is often called meta-classifier. The definition of stacking includes that the data set that is used for the induction of the base-level classifiers has to be disjoint of that used for the induction of the meta-classifier.

In our case the features for the classification methods are the labels predicted by the base-level classifiers. Every classification method may be used for stacking.

2.3.2.6. Information Theoretic Combiner

Meynet and Thiran (2010) proposed a combiner based on the mutual information. Their combiner tries to exploit the fact that ensemble classifiers tend to perform well if the base-level classifiers are diverse and accurate. I will review this fact in more detail in Section 2.3.3.

The main contribution of their work is a new score that measures the accuracy and diversity of a set of base-level classifiers on a data set simultaneously, called *information theoretic score*. But they also show how to to select the best subset of base-level classifiers out of a given set employing this score. I will start this section by introducing the

information theoretic score. After that, I will present the combiner that is based on that score.

The information theoretic score is based on the mutual information. The mutual information is a central concept in information theory.

Definition 28. The *mutual information* between two discrete random variables A, B is defined as

$$I(A;B) = \sum_{a \in A} \sum_{b \in B} P(A = a \wedge B = b) \mathrm{lb} \left(\frac{P(A = a \wedge B = b)}{P(A = a)(B = b)} \right)$$

where lb refers to the binary logarithm.

The mutual information can also be calculated using any other logarithm instead.

Definition 29. The information theoretic accuracy of the base-level classifier $\{\Psi_1, \ldots, \Psi_J\}$ on the data set D_{comb} is defined as the mean mutual information between the labels predicted by the base-level classifiers and the correct labels

$$\mathrm{ITA}(\Psi_1, \ldots, \Psi_J; D_{\text{comb}}) = \frac{1}{J} \sum_{j=1}^{J} I(L_j; \hat{Y})$$

where L_j is the random variable that represents the predictions of the base-level classifier Ψ_j on the data set D_{comb} and $\hat{Y} = \{y_1, \ldots, y_N\}$ the random variable that represents the true labels of the samples in the data set D_{comb}.

Definition 30. The information theoretic diversity between the base-level classifiers $\{\Psi_1, \ldots, \Psi_J\}$ is defined as

$$\mathrm{ITD}(\Psi_1, \ldots, \Psi_J; D_{\text{comb}}) = \frac{\binom{J}{2}}{\sum_{i=1}^{J-1} \sum_{j=i+1}^{J} I(L_i; L_j)} \tag{2.3.6}$$

Note that $\binom{J}{2}$ is the number of distinct pairs that can be built out of J base entities. The information theoretic diversity is the inverse of the mean mutual information between all pairs of base-level classifiers.

Definition 31. The information theoretic score of an ensemble of K classifiers is defined as

$$\mathrm{ITS}(\Psi_1, \ldots, \Psi_J; D_{\text{comb}}) = (1 + \mathrm{ITA}(\Psi_1, \ldots, \Psi_J, D_{\text{comb}}))^3 \mathrm{ITD}(\Psi_1, \ldots, \Psi_J, D_{\text{comb}}) \tag{2.3.7}$$

Employing this score, Meynet and Thiran (2010) propose to use the algorithm that is displayed in Algorithm 2.2 to select a subset of base-level classifiers from a given set of base-level classifiers. I will call the resulting combiner *Information Theoretic Combination* (ITC).

Algorithm 2.2 The learning algorithm of ITC

Input:

- data set $D_{\text{comb}} = (x_1, y_1), ..., (x_N, y_N)$
- set of base-level classifiers $\{\Psi_1, \ldots, \Psi_J\} := B$
- odd size of the to be selected subset K

Procedure
Initialize $k = 1$, $sel^1 = \emptyset$
Select the best individual classifier

$$\begin{aligned} \Psi_{1^*} &= \arg \max_{L_i, i=1,\ldots,J} I(L_i, \hat{Y}) \\ \text{sel}^1 &= \Psi_{1^*} \end{aligned}$$

for $k = 2$ to $(K-1)/2$ do

1. Select the two base-level classifiers Ψ_i, Ψ_k that maximize the information theoretic score

$$(\Psi_i, \Psi_k) = \arg \max_{(\Psi_i, \Psi_k) \in B/sel^{k-1} \times B/sel^{k-1}} (\text{ITS}(\text{sel}^{k-1} \cup \Psi_i \cup \Psi_k; D_{\text{comb}}))$$

2. and add them to the set of selected classifiers

$$\text{sel}^k = \text{sel}^{k-1} \cup \Psi_i \cup \Psi_k$$

end
Output: the discrimination function

$$\text{it}(\Psi_1(x), \ldots, \Psi_J(x)) = \text{mv}(\text{sel}^K)$$

where $\text{mv}(\Psi_1, \ldots, \Psi_K)$ refers to the majority voting rule as defined in Definition 21.

2.3.2.7. Select The Best

The *Select the Best* (SelectBest) combiner refers to the procedure of model selection. Instead of fusing the decision from all base-level classifiers, the most accurate base-level classifier is selected to make the decisions.

The learning algorithm for SelectBest outputs the classifier Ψ_j that has the highest accuracy on the set D_{comb}.

$$\text{sb}(\Psi_1(x), \ldots, \Psi_J(x)) = \Psi_j(x)$$

where $\forall \Psi \in \{\Psi_1, \ldots, \Psi_J\} \text{acc}_{\text{es}}(\Psi_j, D_{\text{comb}}) \leq \text{acc}_{\text{es}}(\Psi, D_{\text{comb}})$

SelectBest can also be interpreted as the WV combiner for that all but the most accurate base-level classifier get assigned zero weight. Unlike the other combiners introduced in this section, the selection of the best classifier does not reduce the risk for one particular data set compared to the best base-level classifier. But if applied to more than one data set, it can decrease the average risk tremendously by picking different base-level classifiers for different data sets.

2.3.3. Why and When do Multiple Classifier Systems Perform Better?

Why?

In the last section I reviewed a variety of combiners for the creation of ensemble classifiers. I gave examples for which the accuracy of the ensemble classifier was higher than the accuracy of any base-level classifier. These examples were theoretical in nature and did not address the question why in practice it is often possible to construct an ensemble classifier that is more accurate than the most accurate base-level classifier. Because of that, I want to introduce three reasons why in practice an ensemble classifier often outperforms classification methods that are based on a single classifier. These reasons were originally introduced by Dietterich (2000).

1. *Statistical:* A learning algorithm can be viewed as searching within a space of classifiers C for the best classifier Ψ_*. When the training data set D is to small, the learning algorithm may find many different classifiers that all achieve the same accuracy on the training data set. By averaging these classifiers, the risk to choose an inadequate classifier is reduced.

2. *Computational:* Even when the statistical problem is absent, learning algorithms that perform some local search may get stuck in local optima. Furthermore, optimal training for two very important classification methods that employ a local search, namely neural networks (Rojas, 1996) and decision trees (Quinlan, 1992), is shown to be NP-hard. An ensemble classifier consisting of base-level classifiers that are generated by running the local search using different starting points may be a better classifier than any of the base-level classifiers.

3. *Representational:* Assume that an algorithm that finds the best classifier in C is available. In this case the use of multiple classifier may still be beneficial as the

optimal classifier Ψ_* may lie outside of C. By combining classifiers from within C it may be possible to expand the space of representable hypotheses.

When?

Of course, the performance of an ensemble classifier is not independent of the performance of its base-level classifiers. It is known that a necessary condition for an increase of the accuracy of the ensemble classifier compared to the most accurate base-level classifier is that the base-level classifiers are accurate and diverse (Hansen and Salamon, 1990). An accurate classifier is a classifier that has an accuracy better than random guessing. Two classifiers are diverse if they make errors on different trials (Dietterich, 2000).

Thus, before using a combiner to fuse the decisions of the base-level classifiers makes sense, it has to be verified that the base-level classifiers fulfill this conditions. The accuracy can be estimated using one of the methods introduced in Section 2.1.3. If a base-level classifier performs better than random guessing, can be tested using the test introduced in Section 2.1.4.1.

Besides the information theoretic diversity, defined in Definition 30, various other diversity measures exist. Kuncheva (2004, chap. 10) compared many diversity measure in terms of their relationship to the final ensemble accuracy. She found that for every diversity measure the relationship between the measured diversity and the final ensemble accuracy is relatively weak. However, if the measured diversity was zero no improvement over the accuracy of most accurate base-level classifier was possible. Because the results are the same for every diversity measure, I will introduce her results in detail for one diversity measure.

One of the most intuitive diversity measures is the disagreement measure.

Definition 32. The disagreement measure between two classifiers Ψ_i, Ψ_k is defined as

$$\mathrm{Di}_{i,k} = P(\Psi_i(x) = y \wedge \Psi_j(x) \neq y) + P(\Psi_j(x) = y \wedge \Psi_i(x) \neq y) \tag{2.3.8}$$

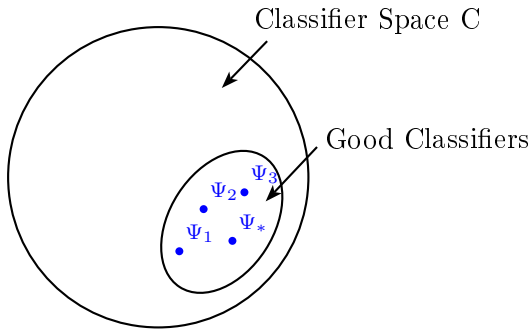
where $(x, y) \in M$

For a binary decision problem $\mathrm{Di}_{i,k}$ is the probability that Ψ_i, Ψ_k disagree. For arbitrary decision problems $\mathrm{Di}_{i,k}$ is the probability that one classifier predicts the correct class and the other classifier predicts a wrong class . The extension to a set of classifiers is straightforward.

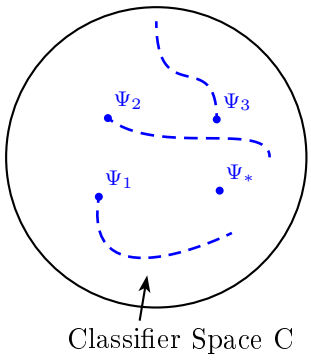
Definition 33. The disagreement measure Di for a set of J base-level classifiers is the mean disagreement measure $\mathrm{Di}_{i,k}$ between all $\begin{pmatrix} J \\ 2 \end{pmatrix}$ pairs, Ψ_i ,Ψ_k, of base-level classifiers.

The probabilities needed for the calculation of the disagreement measure have to be estimated from a data set.

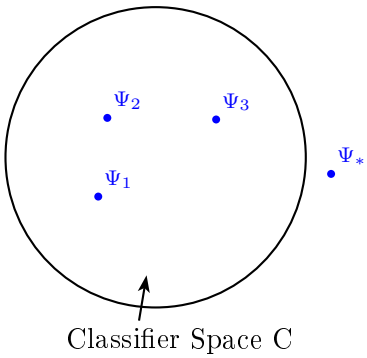
Kuncheva (2004, chap 10) showed for ensemble classifiers built by MV that the relationship between the disagreement measure and the accuracy of the ensemble classifier



(a) The ellipsoid represents the classifiers that perform well on the data set.



(b) The dashed lines represent the hypothetical trajectories of the classifiers during the local search.



(c) Ψ_* lies outside the space in which the classifiers are searched.

Figure 2.3.2.: Illustration of the (a) statistical, (b) computational and (c) representational reason why a an ensemble classifier often performs better than a classification method based on a single classifier. The classifiers Ψ_1, Ψ_2, Ψ_3 represent three classifiers that are induced on the same data set for one particular PR problem. Ψ_* is the optimal classifier. For all three illustrations, the circle represents the space C in which the classifiers are searched. Adapted from Dietterich (2000).

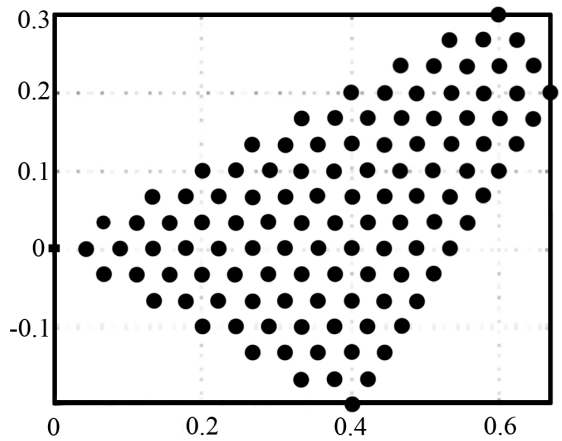


Figure 2.3.3.: Relationship between the disagreement measure and the ensemble accuracy p_{mv}. Each dot represents one ensemble classifier that was build out of the three base-level classifiers with accuracy 0.6. The x-axis describes the disagreement measure and the y-axis the accuracy improvement $p_{\text{mv}} - p$. Copied from Kuncheva (2004, chap 10) and modified for better quality with permission of the author.

is relatively weak. She compared the accuracy of the ensemble classifier p_{mv} against the accuracy of the base-level classifiers theoretically. She defined that the set of base-level classifiers consists of three classifiers that are correct on 18 out of 30 trials, leading to an accuracy of $p = 0.6$. With this constraints a total of 563 different distribution of the correct votes to the trials is possible. Each distribution leads to a different ensemble classifier, for which the accuracy p_{mv} and the disagreement measure can be calculated. The scatterplot for the accuracy improvement $p_{\text{mv}} - p$ can be seen in Figure 2.3.3. From two ensemble classifiers that are based on equally accurate but variably diverse base-level classifiers the classifier that is based on base-level classifiers with a higher disagreement measure does not have to be the classifier with the higher accuracy. Indeed, the accuracy improvement $p_{\text{mv}} - p$ of all ensemble classifier based on base-level classifiers with a disagreement measure of $\text{Di} = 0.4$ span between -0.2 and 0.2, the reason being that the accuracy largely depends on the distribution of the votes of the base-level classifiers to the trials (see Kuncheva, 2004, chap 10). However, her data show that the higher the diversity, the higher is the expected improvement. Furthermore, if $\text{Di} = 0$, no improvement over the most accurate base-level classifier is possible.

3. Combination of Classifiers to Increase Accuracy

The following list repeats the hypotheses from Chapter 1. Recall that ORACLE returns the classifier from the set of base-level classifiers that achieves the highest mean accuracy over all data sets for one particular *Pattern Recognition* (PR) task.

1. The combination of the different feature extraction and classification methods that are employed for the classification of *Electroencephalography* (EEG) signals improves the accuracy of the resulting classifier compared to the best single classifier as estimated by ORACLE and results in a *Pattern Recognition System* (PRS) that performs well on a variety of EEG data sets.

2. A combination of the decisions of the base-level classifiers leads to a more accurate ensemble classifier than the selection of the best classifier by *Select the Best* (SelectBest)

3. The employment of a *Multiple Classifier System* (MCS) leads to a more accurate classifier than the *Concatenation* (CONCAT) approach.

Using the background knowledge presented in Chapter 2, I want to present additional reasons for these hypotheses.

1+2: In Section 2.3.3, we have seen that a MCS is more accurate than the best single classifier if the base-level classifiers are diverse and accurate. Thus, the first hypothesis can only be true if the proposed set of base-level learners produce accurate and diverse base-level classifiers. However, as we saw in the Sections 2.2.3 and 2.2.4 there exists a large variety of feature extraction and classification methods that lead to accurate classifiers. Because they all rely on different characteristics of the EEG signals, there is high a chance that they are diverse.

3: The employment of CONCAT has the advantage that interactions between the different feature extraction methods can be taken into account. However, through the combination of multiple feature extraction methods, the number of features per trial is very high. Thus, I hypothesize that CONCAT will overfit the training data and not lead to a model that generalizes well. Contrary to that, for each base-level learner in the MCS the number of features per trial is comparatively small. The same is true for the combiner. Thus, analogous to *Ledoit's Regularized Linear Discriminant Analysis* (LRLDA), a MCS can be seen as regularization method (Dornhege et al., 2004). Furthermore, a MCS enables the employment of the best fitting classification method per feature extraction method.

The rest of this chapter is organized as follows. First, I will review related work. After that, I will present the methodological details for the comparison. It will include an introduction of new combiners, which I specifically invented for the comparison. The chapter will end with a description of the implementation details.

3.1. Review

3.1.1. Combination of Feature Extraction Methods

Dornhege et al. (2004) already found that a combination of base-level classifiers based on different features performed better than CONCAT and ORACLE. CONCAT performed even worse than ORACLE.

Three feature extraction methods were employed, which resulted in three base-level classifiers. The three different methods were a feature extraction method similar to the method introduced in Section 2.2.3, Autoregressive models, and *Common Spatial Patterns* (CSP). As classification method for the base-level classifiers, as well as for CONCAT, *Regularized Linear Discriminant Analysis* (RLDA) was used. The base-level classifiers were combined at the measurement level (see Section 2.3.1). The outputs of the base-level classifiers were of the form

$$g_j(x) = w_j^t x_j + c_j$$

where w_j and c_j are the parameters learned by one of the three RLDA learning algorithms, each trained on the output of one of the three feature extraction methods x_j, $j \in \{1, 2, 3\}$. The two combination methods employed were stacking with *Linear Discriminant Analysis* (LDA) as meta classifier (see Section 2.3.2.5), which Dornhege et al. (2004) called META and probabilistic voting, in this article called PROB. The ensemble decision for PROB was the average of the three RLDA instances

$$\Psi_{\text{PROB}}(x) = 1 \Leftrightarrow \sum_{j=1}^{J} g_j(x) > 0$$

for META it was

$$\Psi_{\text{META}}(x) = 1 \Leftrightarrow w_{\text{meta}}^T g(x) + c_{\text{meta}} > 0 \tag{3.1.1}$$

where $g(x) = [g_1(x), \dots g_J(x)]^T$ represent the outputs of the base-level classifiers and w_{meta}, c_{meta} are the parameters that were learned by the meta classifier.

Although META is the special case of PROB in which all weights w_{meta} and the bias c_{meta} are learned to be zero, PROB led to a better mean accuracy than META. Their comparison was based on ten subjects.

Boostani et al. (2007) found similar results. By using a combination of features extraction methods, they were able to increase the accuracy compared to ORACLE. They did not employ an ensemble classifier for the feature combination but used a genetic algorithm. They also investigated CONCAT as feature combination method and as well found it performed worse than ORACLE. As classification methods they employed

Adaptive Boosting (AB),LDA, and *Support Vector Machine* (SVM) separately. Their comparison was based on 5 subjects.

Fatourechi et al. (2008) used a two-stage combination of classifiers to build an asynchronous *Brain Computer Interface* (BCI) (see Section 2.2.2). They extracted features for three different types of neurological phenomena. For each channel and phenomenon a SVM was trained. The decision from all SVMs for one phenomenon was combined using *Majority Voting* (MV). In the second stage the outputs of the three ensemble classifiers, one for each phenomenon, was combined by one out of five fixed combination rules. The rule, as well as the features to use, and the parameters for the SVMs were simultaneously optimized using a hybrid genetic algorithm. Their resulting BCI achieved a higher information transfer rate than any existing asynchronous BCI. Their comparison was based on four subjects.

3.1.2. Combination of Predefined Base-Level Classifiers

Ensemble classifiers are not limited to the combination of different feature types. They might be beneficial in all cases were a large set of heterogeneous features is combined.

Fazli et al. (2009) were able to build a subject independent BCI system employing an ensemble classifier. The base classification method was LDA. For each session a LDA was trained on CSP power features. A session hereby refers to one recording session for one subject. A varying number of sessions per subject was recorded. Before the training of LDA, the data was bandpass-filtered in 9 different frequency bands. This led to a total of $9 \times \#sessions$ base-level classifiers. The final classification was done by a weighted sum of the continuous outputs of the LDA base-level classifiers, similar to META described above. But instead of LDA, Fazli et al. (2009) used quadratic regression with l_1 regularization to obtain w_{meta} and c_{meta}. Their ensemble classifier performed better than various baseline methods, including ORACLE, and other ensemble classifiers. Their comparison was based on four subjects.

Rakotomamonjy and Guigue (2008) used a combination of SVMs to build a BCI. It won one discipline of the BCI Competition III (Blankertz et al., 2006). For feature extraction they bandpass-filtered the data with cut-off frequencies 0.1 and 10Hz and decimated the signal to 14 samples per channel. Each of the 17 SVMs was trained on a partition of the data. They tried to choose the partitions such that they were as homogenous as possible. The final classification was done by averaging the continuous outputs of all SVMs, analogously to PROB. Their comparison was based on two subjects.

3.1.3. Methods That Generate Base-Level Classifiers

Ensemble classifiers are not limited to the combination of base-level classifiers that are defined by the designer. There are multiple methods that, besides a combiner, also include a technique to create multiple base-level classifiers from a data set.

Sun et al. (2007) showed that AB, Bagging (Breiman, 1996), and random subspace (Bryll et al., 2003) are able to boost the accuracy compared to a single classifier. They came to this conclusion after evaluating these methods on nine subjects performing a

motor imaginary task. The different base-level classifier were generated by using power spectral densities as features and the base-level classifier generating capabilities of the compared methods. They evaluated the three methods for SVM, *k-Nearest Neighbor* (k-NN), and C4.5 decision trees (Quinlan, 1992) as base classification methods separately.

Boostani and Moradi (2004) compared AB with an one hidden layer neuron (Rojas, 1996) as weak learner against LDA on three different types of features, Hjorth parameters, band power and fractal dimension. No combination of features was considered. They based their comparison on five subjects, performing a motor imaginary task. Their results show that while the combination of band power and LDA yielded the best mean accuracy (over all subjects) for two subjects the combination of fractal dimension and AB led to the best accuracy. They concluded that "for each individual, we have to find the best combination of feature and classifier or on some occasions, a combination of the features by evolutionary algorithms or a tree combination of classifiers which can lead to the best result." (Boostani and Moradi, 2004, p 217)

Sun (2007) employed an explicit trainable combiner, the so called *improved random subspace method*, for the classification of mental imagery data. The base-level classifiers were build by training a SVM with different subspaces of the feature space. The final classification decision was

$$\Psi_{\text{irsp}}(x) = \arg\max_{y_l \in Y} \sum_{j=1}^{J} w_j(x)\delta(\Psi_j(x), y_l) \tag{3.1.2}$$

where $w_j(x)$ is the fraction of correctly classified samples by Ψ_j of the k nearest neighbors, with respect to the Euclidean distance, of x. They showed that their method performs better than another similar ensemble method.

Although it seems promising, to my knowledge nobody tried to use one of the multiple classifier methods that are able to generate base-level classifiers on a combination of features.

3.2. Learning Algorithm for Ensemble Classifiers

In this section I will introduce the learning algorithm that I use to induce the different ensemble classifiers. Independently of the base-level classifiers and the combiner, the ensemble classifiers are induced as follows: The designer specifies a set of *base-level learners* $I_{\Psi_1}, \ldots, I_{\Psi_J}$. The classification behavior of each base-level learner is estimated using 10-fold stratified cross validation. The estimated behavior is fed to the combiner I_r. To clarify this: Let D be the available data set. The data set is split into 10 partitions $\{D_1, \ldots, D_{10}\}$. For each partition and base-level learner a base-level classifier is induced $\Psi_{j,n} = I_{\Psi_j}(D/D_n)$. For each base-level learner I_{Ψ_j} the combiner gets the predicted class of each trial $(x, y) \in D$ by the classifier $\Psi_{j,n}$ for that $(x, y) \notin D_n$ as input. Based on this information every combiner that I introduced can estimate the necessary properties of the base-level classifiers. After the combination method learned the combination rule, the base-level learners induce the base-level classifiers based on the complete data set D. The base-level classifiers are combined using the combination rule that was inferred in

Algorithm 3.1 The employed training algorithm for inducing the ensemble classifiers.

Input
A set of base-level learners $I_{\Psi_1}, \ldots, I_{\Psi_J}$
An untrained combiner I_r
A data set D
Procedure

1. Split D into 10 disjoint subsets $D_n, n \in \{1, \ldots, 10\}$ in respect to the stratified 10-fold cross-validation scheme
   a) For each subset D_n
      i. Train each base-level learner on the remaining data set D/D_n. $\Psi_{j,n} = I_{\Psi_j}(D/D_n)$
      ii. Calculate the prediction of each base-level classifier $\Psi_{j,n}$ for each trial $(x, y) \in D_n$, $l_{i,j} = \Psi_{j,n}(x)$, where i is the index of (x, y) in D.
   b) Train the combiner with the matrix L, with entries $l_{i,j}$. $I_r(L)$. Remember the inferred combination rule r.
2. Train all base-level learners with the complete data set D, $\Psi_j = I_{\Psi_j}(D)$
3. Create the ensemble classifier Ψ_{ens} by combining the base-level classifiers with the rule r inferred in step 2 (b).

Output
The ensemble classifier Ψ_{ens}

the previous step. This procedure is explained as pseudo code in Algorithm 3.1. It can be interpreted as a classification method with the hyper-parameters $I_{\Psi_1}, \ldots, I_{\Psi_J}$ and I_r.

3.3. Combiners

Because the combination at the abstract level (see Section 2.3.1) is the only level of combination that allows the usage of arbitrary classification methods, I only include combiners that combine the base-level classifiers at the abstract level in the comparison. In previous studies, only stacking with SVM as meta classifier (Fazli et al., 2009), MV and AB (Sun et al., 2007) have been used if the base-level classifier were combined at the abstract level. I compare all combiners that have been introduced in Section 2.3.2. In addition to the existing combiners, I invented several new combiners, mostly extensions of existing combiners, for the comparison.

This section will continue with the introduction of the combiners that I invented. Furthermore, it will contain the detailed settings for the existing combiners. Assume that the situation is as described at the beginning of Section 2.3.2.

3.3.1. Significance Majority Voting

Suppose that the base-level classifiers consist of 100 classifiers with accuracy 50% and one classifier with accuracy 100%. The accuracy of the ensemble classifier built by MV would hardly be over 50%. To make MV applicable to situations in which a majority of the base-level classifiers do not perform better than random guessing, I extend it such that only the votes of the base-level classifiers that have an estimated *Balanced Accuracy* (BAC) that is significantly higher than 0.5 are included in the decision.

For each base-level classifier, significance against random guessing is tested using the test introduced in Section 2.1.4.1. I call this extension *significance extension* and the resulting combiner *Signifigance Majority Voting* (SMV). Note that contrary to MV, SMV is a trainable combiner. This extension can also be applied to *Weighted Majority Voting* (WMV). I call the resulting combiner *Signifigance Weighted Majority Voting* (SWMV).

3.3.2. Dependent Weighted Majority Voting

While it was shown that WMV is the optimal combiner when the base-level classifiers are independent (see Section 2.3.2.1), WMV is not the optimal combiner if the independence assumption is violated .

Theorem 34. *Let* $\Psi_1, \ldots, \Psi_7$ *be base-level classifiers. Let* Ψ_1, Ψ_6, Ψ_7 *be independent. Let* $\Psi_1 = \Psi_2 \ldots = \Psi_5$*, i.e.,* $\forall i, j \in \{1, \ldots, 5\} \forall x \in X \forall y_l \in Y\, P(\Psi_i(x) = y_l | \Psi_j(x) = y_l) = 1$*. In addition,* $acc(\Psi_1, p) = 0.7$ *and* $\mathrm{acc}(\Psi_6, p) = \mathrm{acc}(\Psi_7, p) = 0.8$*. Then the accuracy* $\mathrm{acc}(\Psi_{\mathrm{wmv}}, p)$ *of the ensemble classifier created by WMV is 0.7.*

Proof. The weights as learned by WMV are $w_1 = w_2 = \ldots = w_5 = 0.8473$ and $w_6 = w_7 = 1.3863$. Because of the equality of $\Psi_1 \ldots \Psi_5$, the class for which Ψ_1 votes gets assigned weight $4 \cdot 0.8473 = 4.2365$. The remaining two classifiers Ψ_6 and Ψ_7 share a total weight of 2.27726. Hence, because $4.2365 > 2.27726$, $\mathrm{acc}(\Psi_{\mathrm{wmv}}, p) = \mathrm{acc}(\Psi_1, p) = 0.7$ □

Theorem 35. *Let* $\Psi_1, \ldots, \Psi_7$ *be as in Theorem 34 but* $w_1 = 0.8473$*,* $w_2 = \ldots = w_5 = 0$ *and* $w_6 = w_7 = 1.3863$*. Then* $\mathrm{acc}(\Psi_{\mathrm{wv}}, p) = 0.8320$

Proof. The ensemble classifier Ψ_{wv} makes the correct decision if Ψ_1, or Ψ_6 and Ψ_7 make the correct decision.

$$p_{wmv} = 0.7 \cdot 0.8 \cdot 0.8 + 0.3 \cdot 0.8 \cdot 0.8 + 0.7 \cdot 0.2 \cdot 0.8 + 0.7 \cdot 0.8 \cdot 0.2 = 0.864$$

□

The ensemble accuracy is increased by giving only one classifier out of the dependent classifiers a non zero weight. This strategy is used by *fixed Adaptive Boosting* (fAB) to correct for dependencies between base-level classifiers. A different strategy is to decrease the weight of each dependent base-level classifier. The same accuracy as in Theorem 35 can be obtained by dividing the weights $w_1 = \ldots = w_5$ by 5. If the classifiers are really identical, the two different strategies lead to the same ensemble accuracy. However, we

can only estimate the behavior of the base-level classifiers and, thus, reducing the weight for every dependent base-level classifier may be a more robust strategy than assigning one base-level classifier a large weight and the rest a small weight.

Of course, the question is how to generally correct the weights for dependencies and how to treat the most common case when classifiers are neither completely dependent nor independent. I propose the following strategy: To estimate the dependence of a base-level classifier to all other base-level classifiers the mutual information between the classifier and the remaining base-level classifiers is estimated. The corrected weight for each base level classifier is then

$$w_{\mathrm{j}} = wm_j(\sum_{i=1,i\neq j}^{J} I(\Psi_j, \Psi_i) + 1)^{-1} \tag{3.3.1}$$

where wm_j is the weight obtained by the original WMV combiner, $I(\Psi_j, \Psi_i)$ the mutual information between two classifiers as in Section 2.3.2.6. This correction procedure is motivated by the fact that the mutual information is zero for two independent classifiers and $\min(H(\Psi_j), H(\Psi_i))$ for two identical classifiers, where $H(\Psi_j)$ denotes the entropy of the classifier Ψ_j. I call this correction of the weights dependency correction and the resulting combiner *Dependent Weighted Majority Voting* (DWMV). It can, of course, also be combined with the significance extension. I call the resulting combiner *Dependent Significant Weighted Majority Voting* (DSWMV).

Another possibility would be to use the normalized mutual information as estimate of the dependencies, which leads the correction scheme

$$w_{\mathrm{j}} = wm_j(\sum_{i=1,i\neq j}^{J} \frac{I(\Psi_j, \Psi_i)}{\min(H(\Psi_j), H(\Psi_i))} + 1)^{-1}$$

and ensures that for two completely dependent classifier $\frac{I(\Psi_j,\Psi_i)}{\min(H(\Psi_j),H(\Psi_i))}$ is one. This correction scheme is not examined in this thesis.

3.3.3. Harmonic Series Weighted Voting

Another combiner that I invented for the comparison is the *Harmonic Series Weighted Voting* (HSWV) combiner. It is also a *Weighted Voting* (WV) combiner. The weight for each classifier is

$$w_j = \frac{1}{r_j}$$

where r_j denotes the rank of the corresponding base-level classifier in comparison to the remaining base-level classifiers. The rank is calculated by sorting the base-level classifiers with respect to their BACs. Hence, the base-level classifier that gets assigned weight $\frac{1}{2}$ is the base-level classifier that produces the second highest BAC on D_{comb}.

As the HSWV combiner does not take into account dependencies between base-level classifiers, the dependency extension is also applied to HSWV, leading to the *Dependent Harmonic Series Weighted Voting* (DHSWV) combiner.

3.3.4. Random Weighted Voting

As base-line method for the comparison I define *Random Weighted Voting* (RWV), also a WV combiner. The weight w_j for each base-level classifier Ψ_j is randomly picked according to the uniform distribution over the interval $(0, 1)$.

3.3.5. Details for the Existing Combiners

The details for the existing combiners are as follows. For *Bayes Combination* (BC) Equation 2.3.5 is used to estimate the support. I choose $B = 1$ for the hyper-parameter B. As meta classifiers for stacking, I employ two different classification methods, LDA and LRLDA, resulting in the two combiners, *Stacking with Linear Discriminant Analysis* (STLDA) and *Stacking with Ledoit's Regularized Linear Discriminant Analysis* (STLRLDA). For all combiners that need an estimate of the accuracy, I use the BAC as estimate as some data sets are imbalanced. As classification method for CONCAT I employ LRLDA. LRLDA gets as input the concatenation of the feature vectors originating from all unique feature extraction methods. For *Information Theoretic Combination* (ITC) I set the size K of the to be selected subset to seven.

3.4. Base-Level Learners

While the proposed ensemble learning algorithm accepts arbitrary base-level learners as input, I have to define a set of base-level learners that is used for the comparison of the different combiners. Remember that one goal of this thesis is to build a classification method that works well on a variety of different EEG data sets. I want to compare the combiners on heterogeneous data sets. If no base-level learner produces an accurate base-level classifier, a comparison of the different combiners is not possible. Because of these reasons, the set of base-level learners has to be broad and has to capture the most prominent characteristics of EEG signals. This implies that for any particular EEG data set it is very likely that some base-level learner lead to inaccurate base-level classifiers.

As classification methods only linear classifiers are employed, following the reasoning of Blankertz et al. (2010a, p 118) that in their experience "linear methods perform well, if an appropriate prepossessing of the data is performed".

Every method that I will introduce in the remainder of this section is applied to data from the following seven frequency bands separately: α (8-12 Hz), β (12-30 Hz), γ(30-70 Hz), δ(0-4 Hz), θ(4-8 Hz), *con* (1-45Hz) and *rem* (70+ Hz). This leads to a total of $\#\text{methods} \cdot 7$ base-level learners.

The CSP feature extraction method is used because it is currently the standard method in BCI research to quantify signal power changes. The hyper-parameter k is set to three as advised by Blankertz et al. (2008). The feature that is extracted of the signal transformed by the CSP patterns is the logarithm of the variance. As classification method for the base-level learners based on CSP, LRLDA is used.

To quantify amplitude changes a set of base-level learners based on *Spatio Temporal Features* (STF) (see Section 2.2.3) is employed. Three different approaches are used, *Local*

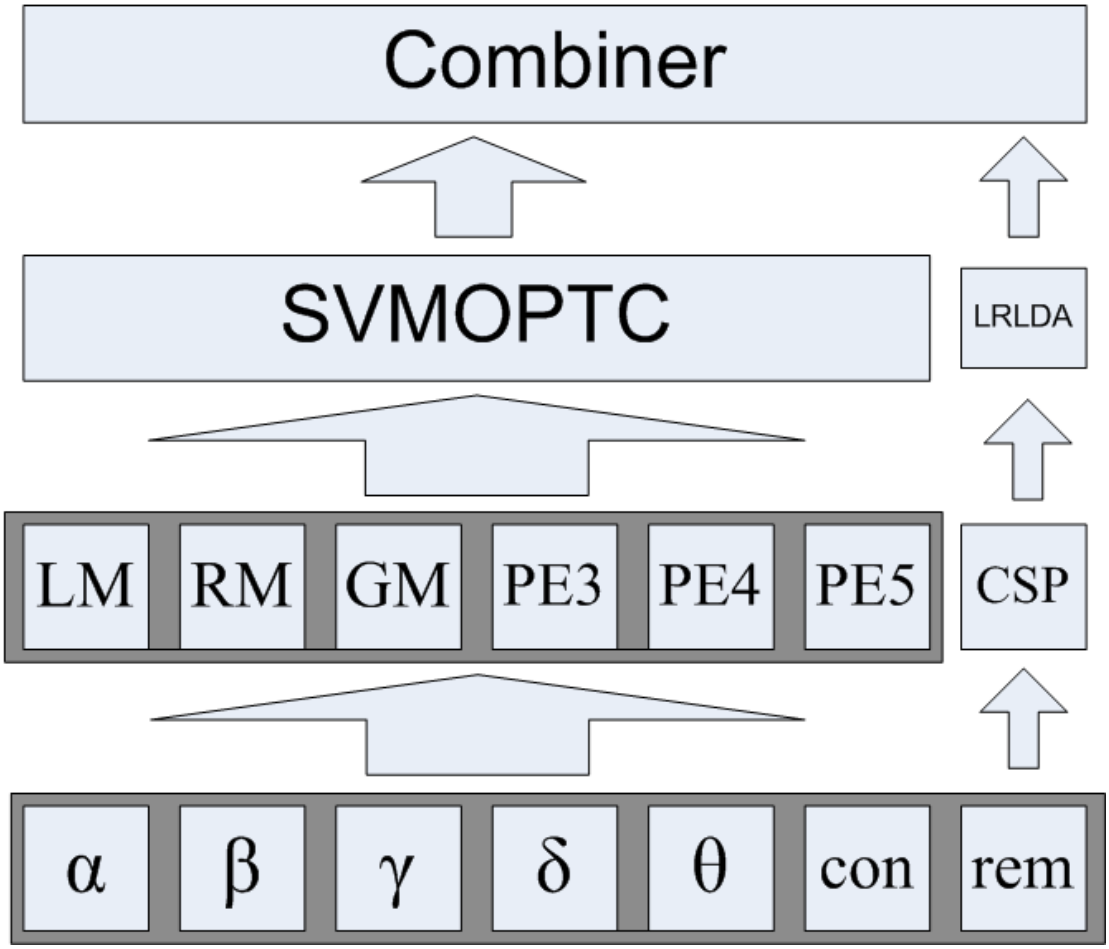


Figure 3.4.1.: Illustration of the base-level learners.

Algorithm 3.2 Learning algorithm of SVMOPTC,

Input
Data set D
Set of candidates for C, C_{can}
Procedure
maxbac = 0
maxC = 0
for each $c \in C_{\text{can}}$

1. Estimate the BAC of the SVM instance with $C = c$ using stratified 10-fold cross-validation; $\text{bac}_{\text{cv}}(\text{SVM}_{C=c}, D)$, where $\text{SVM}_{C=c}$ denotes the learning algorithm of SVM with the hyper-parameter C set to c.

2. If $\text{bac}_{\text{cv}}(\text{SVM}_{C=c}, D) > \text{maxbac}$, $\text{maxbac} = \text{bac}_{\text{cv}}(\text{SVM}_{C=c}, D)$ and $\text{maxC} = c$.

end
Output
$\text{SVM}_{c=\text{maxC}}(D)$

Means (LM), *Regional Means* (RM) and *Global Mean* (GM). The means are calculated on non-overlapping intervals of length 50ms for the LM approach. For the RM approach, the means are calculated on five non-overlapping equally sized intervals that span the whole trial. For GM, the mean is calculated over the complete trial.

The *Permutation Entropy* (PE) is used as measure of complexity. It is calculated for the embedding dimensions 3 (PE3), 4 (PE4) and 5 (PE5). For smaller embedding dimensions the PE would hardly contain any information and for larger embedding dimensions the typical number of time points per channel and trial would not be sufficient to get a reasonable estimate of the PE.

As classification method for the base-level learners based on the PE and the STF a linear SVM is used. The hyper-parameter C is optimized using stratified 10-fold cross-validation and the BAC as performance measure. Candidates for C are chosen from $\{10^i : i \in \{-5, -3, -1, 1, 3, 5, 7, 9, 11, 13, 15\}\}$. The resulting classification method will be called *Support Vector Machine with Optimization of the C hyper-parameter* (SVMOPTC) in the remainder. The learning algorithm for SVMOPTC is shown in Algorithm 3.2. A different possible approach is to include every SVM as a different base-level learner. However, pilot experiments suggested that base-level classifiers that only differ in the C hyper-parameter of their SVM are either completely dependent, or one base-level classifier performs clearly superior. Hence, the selection of the best C hyper-parameter seems more appropriate.

The overall number of base-level learners is

$$\#\text{frequencybands} \cdot (\#\text{CSPLDA} + \#\text{STFSVM} + \#\text{PESVM}) = 7 \cdot (1 + 3 + 3) = 49$$

where #frequencybands is the number of frequency band on which all feature extraction methods are applied, #CSPLDA the number of feature extraction methods that are

based on CSP, #STFSVM the number of feature extraction methods that are based on STF and, and #PESVM the number of feature extraction methods that are based on PE.

3.5. Details for CONCAT and ORACLE

As input for for CONCAT the concatenation of the feature vectors obtained by the different feature extraction methods from the base-level learners is employed. As classification method for CONCAT I choose LRLDA because it is a very powerful and regularized method. I compare the classification methods on five different simulation scenarios and on data sets originating from four different EEG studies. ORACLE returns, for each data set, the classifier that achieves the highest mean BAC over all data sets from the respective study/scenario out of all base-level classifiers. The BAC is estimated using stratified 10-fold cross-validation.

3.6. Implementation

The generic learning algorithm for ensemble classifiers, the combiners, as well as all the methods needed for the base-level learners are integrated into the multivariate toolbox of Fieldtrip (Oostenveld et al., 2011). Fieldtrip is an EEG analysis toolbox for Matlab (MathWorks, 2012). Fieldtrip's multivariate toolbox contains algorithms from the field of PR for the analysis of EEG data.

The integration of the code into an existing analysis toolbox serves multiple purposes. It makes it easier for others to verify and reproduce the results. Furthermore, the learning algorithm for ensemble classifiers can be easily used to implement and test new combiners or base-level learners. Another advantage is that many of the algorithms implemented for this work, e.g., the classification and the feature extraction methods, can be easily reused for completely different projects.

Fieldtrip was chosen over other existing toolboxes such as BioSig (Schlogl and Brunner, 2008) and BCILAB, which is included in EEGLAB (Delorme and Makeig, 2004), because it focuses more on single-trial analysis than on building BCIs, it is object oriented (at least the multivariate toolbox), and it is the major EEG analysis toolbox used at my institute.

The design of the learning algorithm for ensemble classifiers is oriented at and reuses some of the code from the ft_mv_gridsearch class of fieldtrip. Furthermore, for efficiency reasons it is implemented such that it accepts a set of combiners and returns a set of ensemble classifiers.

While the ensemble learning algorithm and the combiners are implemented by me, some methods needed for the base-level learners do already exists in fieldtrip, namely CSP and SVM. Also, the cross-validation procedure from fieldtrip is used.

The leading paradigm for the implementation is the extensive employment of automated testing to ensure the correctness of the implemented algorithms, leading to a total of 68 test cases.

4. Results

In this section I will present the results of an empirical comparison of the proposed methods. Recall that the questions I want to answer are:

1. Does a combination of base-level classifiers based on different feature extraction and classification methods improve the *Balanced Accuracy* (BAC) in comparison to the ORACLE?

2. Does a combination of base-level classifiers based on the different feature extraction and classification methods lead to a more accurate classifier than *Select the Best* (SelectBest) and *Concatenation* (CONCAT)

3. Which of the combiners produces the most accurate ensemble classifier?

4. Is the set of base-level learners that I defined sufficient? Does it produce accurate and diverse classifiers?

5. Does the best method result in a classification method that works well not only on a single but on a variety of *Electroencephalography* (EEG) data sets?

This section will start with an introduction of the methods that are used to compute the results. It will continue with the presentation of the results of a simulation study. After that, the results on the EEG data sets will be presented. I will use many abbreviations throughout this chapter. If you are reading this thesis on a computer, you may click on the abbreviation to get to the list of abbreviations (see Chapter 6). If you are reading this thesis in paper form, the list of abbreviations is provided to you as separate spreadsheet.

4.1. Methods

The *Balanced Accuracy* (BAC) (see Definition 13) for each method on each data set is estimated using stratified 10-fold cross-validation (see Definition 12). This results in a so called nested cross-validation procedure for the *Multiple Classifier System* (MCS). The outer cross-validation loop is used to estimate the accuracy of the ensemble classifiers and the inner cross-validation loop is part of the training of the ensemble classifiers (see Algorithm 3.1). Within the training of the ensemble classifiers there is even another cross-validation loop as part of the training of *Support Vector Machine with Optimization of the C hyper-parameter* (SVMOPTC).

As aggregated performance measures, over the data sets, the mean of the BACs and the ranks computed by the Friedman test (see Section 2.1.4.2) are presented.

For addressing the statistical significance of the results, depending on the situation, two different tests are employed. The differences between the different methods is translated into a p-value using the post-hoc procedure on the test statistics calculated by the Friedman test (see Section 2.1.4.2). When comparing a set of combiners against a base-line method, e.g., *Select the Best* (SelectBest), *Concatenation* (CONCAT) and ORACLE, the probability threshold α is adjusted for multiple testing using the Bonferroni correction. For the comparison of all pairs of combiners, the critical value is adjusted stepwise using the *Shaffer's Static Procedure* (SSP) (see Section 2.1.4.2). When a method is tested against random guessing, the test introduced in Section 2.1.4.1 is used. When multiple methods are compared against random guessing on the same data set, the critical value α is adjusted using the Bonferroni correction.

As characteristics of the set of base-level classifiers I report the average disagreement measure and the average number of base-level classifiers that achieve a BAC better than random guessing. The BAC is estimated using 10-fold stratified cross-validation. If a classifier performs better than random guessing, is tested using the test introduced in Section 2.1.4.1. I do not correct for multiple comparisons, because I want to test for each classifier independently if it performs better than random guessing. Hence, if all 49 base-level classifier guess randomly, this test will, on average, find $49 \cdot 0.05 = 2.45$ classifiers to perform better than random guessing.

4.2. Implementation

As with the algorithms introduced in the last section, I integrate the algorithms that are needed to generate the results, e.g., the statistical tests, into the multivariate toolbox of Fieldtrip.

Because of the large amount of data sets and the amount of methods that are compared, the total computing time to generate the results for the real data set exceeds four years. Hence, a regular computer would not be sufficient to calculate the results in a reasonable time. Therefore, I use a 180 core cluster computer to calculate the results on the real data sets, and employ a 30 core cluster computer to estimate the *Balanced Accuracy*s (BACs) on the simulated data sets.

I modify the cross-validation procedure of fieldtrip such that it accepts learning algorithms that return a set of classifiers to be compatible with the implementation of the learning algorithm for ensemble classifiers. Furthermore, because parallelism on the data set level is not sufficient to get the results in a reasonable time, I modify the cross-validation procedure such that each of the 10 folds can be processed independently on a separate machine.

4.3. Simulation

Before comparing the different combiners on real *Electroencephalography* (EEG) data sets, I compare them on simulated data sets. Besides the comparison of different combiners on data sets with known properties, the main goal of the simulation study is to reduce the

number of combiners that have to be included in the comparison on the real data sets. This is motivated by two facts: First, when testing the differences between the combiners and the base-line methods for significance, the more combiners are included in the test the less likely is it to find a significant effect. Every additional combiner increases the rank differences required for a significant effect. Second, the reduction of the number of combiner reduces the computation time.

For the simulation study, different base-level classifiers are simulated for five different scenarios. The scenarios are inspired by situations that occurred on real data sets. For every scenario, 1000 data sets are simulated. Each data set represents a binary classification problem and consists of 1000 trials per class.

4.3.1. Scenarios

Base Scenario

For the base scenario, 15 base-level classifiers are simulated. The *Balanced Accuracy*s (BACs) of the base-level classifiers are equally distributed in the interval $[0.55, 0.8]$. Hence, every base-level classifier is accurate. After ensuring that the per-class accuracies are the same, which ensures that the BAC is equivalent to the accuracy, the classifier outputs are shuffled within the class. This ensures high diversity between the classifiers.

The base scenario represents the situation when the base-level classifiers are independent and accurate. It can be seen as the optimal scenario. The remaining scenarios are extensions of the base scenario. They all contain the base-level classifiers that were generated for the base scenario.

Noise Scenario

For the noise scenario, 45 classifiers are added that arbitrarily predict class one or two, with equal probability, independently of the true label. This scenario evaluates the capacity of the combiners to deal with base-level classifiers that do not provide any information about the true label. Because I chose a broad set of base-level learners, it is very likely that such classifiers are part of the base-level classifiers set.

Doubles Scenario

For the doubles scenario, randomly one of the 15 classifiers from the base scenario is picked and duplicated five times. Each classifier has the same chance to get picked. This procedure is repeated 9 times, resulting in 45 classifiers. These 45 classifier are simply repetitions of existing classifiers. This scenario represents the case when there are strong dependencies between the base-level classifiers.

Constant Scenario

For the constant scenario, 45 classifiers that constantly predict one class are added. This scenario represents the worst case of dependent noise. It is motivated by pilot

method	base	+ noise	+ constant	+ doubles	+ all
STLRLDA	**93.77**	93.68	93.67	**93.67**	**93.41**
STLDA	93.67	93.40	93.67	**93.67**	**93.40**
fAB	93.67	93.08	93.65	**93.62**	93.03
DSWMV	93.72	93.69	93.72	92.70	92.66
DWMV	93.72	93.58	93.72	92.70	92.49
SWMV	**93.80**	**93.78**	**93.8**	86.79	86.79
WMV	**93.80**	93.71	**93.8**	86.79	86.79
BC	**93.79**	93.70	**93.79**	86.78	86.78
HSWV	88.84	88.65	80.94	82.79	82.75
SMV	92.17	90.26	92.17	83.50	83.65
DHSWV	89.80	89.37	72.55	83.42	75.35
SelectBest	77.87	77.87	77.87	77.87	77.87
ORACLE	67.79	67.79	67.79	67.79	67.79
RWV	88.44	72.05	50	83.84	57.56
MV	92.17	75.25	50	83.50	55.84
ITC	76.92	58.68	50	76.92	50.00

Table 4.1.: Mean BAC, in percent, for each method and scenario, sorted by their BACs on the all scenario. The values printed in bold letters represent the best method on the respective scenario. If in one column there is more than one value printed in bold, there was no significant difference between those methods. For the all scenario, missing column delimiters imply that no significant difference could be observed between these methods. The gray rows mark the combiners that are proposed in this thesis.

experiments, in which base-level classifiers that only differed in the C hyper-parameter of their *Support Vector Machine* (SVM) were part of the base-level classifier set. It was observed that for some C values these base-level classifiers constantly predict one class.

All Scenario

The all scenario contains the base-level classifiers from the base scenario and the base-level classifiers from all other scenarios, resulting in a total of 150 base-level classifiers. The main motivation for this scenario is to evaluate the performance of the different combiners in the case when all noise sources occur at the same time. This is believed to be the most realistic scenario.

4.3.2. Results

By construction, the diversity between the base-level classifiers from the base scenario is high. The average disagreement measure is 0.4390 with a standard deviation of 0.0132. All other scenarios include the set of base-level classifiers from the base scenario. Hence, for each scenario, there exists a subset of accurate and diverse base-level classifiers.

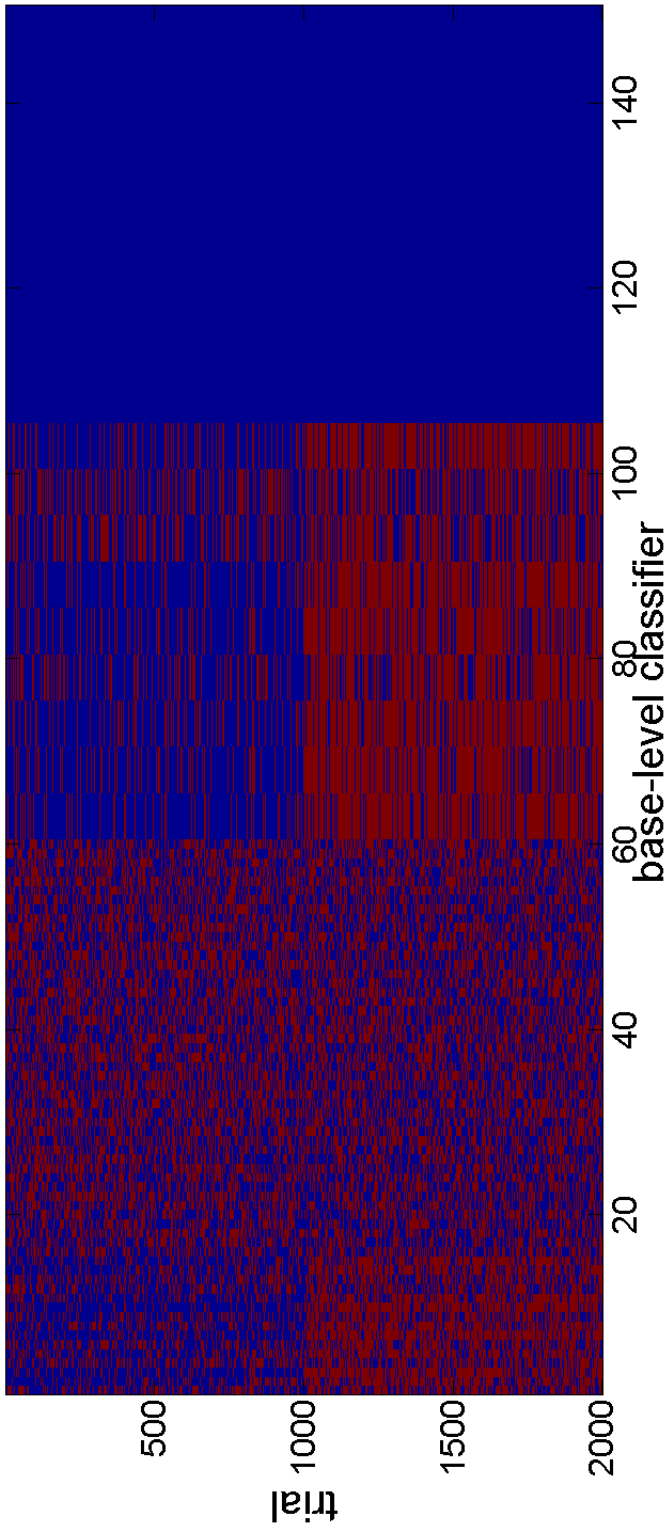


Figure 4.3.1.: Illustration of the voting behavior of the base-level classifiers for one data set from the all scenario. A red (blue) rectangle in the ith row and the jth column denotes that the jth base-level classifier classifies the ith trial as blue (red). The true class for the first 1000 trials is blue. The true class for the last 1000 trials is red. The first 15 base-level classifiers are the classifiers from the base scenario. The next 45 base-level classifiers originate from the noise scenario. The following 45 base-level classifiers are from the doubles scenario. The last 45 base-level classifiers originate from the constant scenario.

The results for all combiners and scenarios are summarized in Table 4.1. The ensemble classifiers created by almost every combiner perform better than ORACLE on all scenarios. On the base scenario the ensemble classifiers built by the best combiners achieve a mean BAC of 93.80%, while ORACLE achieves a mean BAC of 67.79%. Hence, the combination of the base-level classifiers is able to boost the mean BAC by more than 26%.

Not surprising, the ensemble classifier built by *Select the Best* (SelectBest) results in a mean BAC of 77.87% on all scenarios. Thus, the combination of base-level classifiers produces more accurate ensemble classifiers than the selection of the most accurate base-level classifier.

The ensemble classifiers build by *Stacking with Ledoit's Regularized Linear Discriminant Analysis* (STLRLDA), *Stacking with Linear Discriminant Analysis* (STLDA), *fixed Adaptive Boosting* (fAB), *Dependent Significant Weighted Majority Voting* (DSWMV), *Signifigance Weighted Majority Voting* (SWMV), *Weighted Majority Voting* (WMV), *Bayes Combination* (BC), and *Signifigance Majority Voting* (SMV) perform, with a mean BAC span of 92.17% to 93.77%, relatively similar on the base scenario. I will call this group of combiners *promising combiners* in the remainder of this work, because the remaining combiners produce tremendously less accurate ensemble classifiers on the base scenario.

On the all scenario there is one group that performs much better than the rest of the combiners. STLRLDA, STLDA, fAB, DSWMV, and *Dependent Weighted Majority Voting* (DWMV) produce mean BACs that are higher than 94.48%, while the mean BACs achieved by the remaining combiners are below 86.8%. I will call this group of combiners *winning combiners* in the remainder of this work. Note that the winning combiners are a subset of the promising combiners.

Of course, the question is what is the reason for the big differences between the winning combiners and the rest of the combiners. When comparing the learning algorithms of the winning combiners against the learning algorithms of the promising combiners, one big difference becomes apparent. The winning combiners create the combination rule such that it takes into account dependencies between the base-level classifiers. Furthermore, there is empirical evidence that the proper handling of dependent classifiers leads to the fact that the winning combiners perform best. On the noise and the constant scenario the promising combiners perform almost on the same level than on the base scenario. Contrary to that, on the doubles scenario only the winning combiners yield a similar mean BAC compared to the base scenario. The mean BACs for the rest of the combiners on the doubles scenario is considerably smaller than their mean BACs on the base scenario.

After having identified the combiners that produce the most accurate ensemble classifiers on the all scenario, I will continue this section with a detailed performance analysis for every combiner. Based on that analysis the combiners that will be included in the comparison on the real EEG data sets are selected.

The stacking combiners STLRLDA and STLDA share the first rank on the all scenario with mean BACs of 93.41% and 93.4%. They perform significantly better than all other combiners. Over all scenarios the performance of these two combiners is promising. STLRLDA performs better than STLDA on each scenario. Because of that, from the

stacking combiners, only STLRLDA will be included in the comparison on the real data sets, although per scenario the difference is negligible.

With a mean BAC of 93.03%, fAB achieves the second rank for the "all scenario" and also performs well over all scenarios. Therefore, fAB will be included in the real data comparison.

The combination methods from the *Weighted Majority Voting* (WMV) family also produce promising results. For the base, the noise and the constant scenario at least one combiner from that family ranks first. For the all scenario DSWMV (mean BAC 92.66%) and DWMV (mean BAC 92.49%) share the fourth place. SWMV (mean BAC 86.79%) and WMV (mean BAC 86.79%) follow on the shared fifth place. The application of the significance correction, which I introduced in Section 3.3.1, to WMV constantly boosts BACs of the resulting ensemble classifiers. SWMV always performs better or equally well than WMV. The same is true when comparing DSWMV and DWMV. But the differences are very small. However, because the significance correction led to a more accurate ensemble classifier on every scenario, WMV and DWMV will not be included in the final comparison.

What follows is the evaluation of the dependency extension (see Section 3.3.2). On the base, noise and constant scenario SWMV performs significantly better than DSWMV. So, it seems that in the case when there are no dependencies between the base-level classifiers the dependency extension actually worsens the performance of the resulting ensemble classifier. However, the BAC differences are relatively small. In contrast to that, there is a relatively huge difference of 6% in favor of DSWMV on the doubles and all scenario. This provides evidence that the dependency extension produces, in fact, a better combination rule than WMV if there are dependencies between the base-level classifiers. Therefore, DSWMV and SWMV will both be included in the real data comparison. DSWMV performs significantly worse than three other methods, that take into account dependencies between the base-level classifiers, namely fAB and the two stacking combiners

BC is part of the promising combiners. When there are no dependencies between the classifiers, BC is one of the best combiners. It achieves the shared first place on the base and the constant scenario and the shared second place on the noise scenario. However, as for all the other promising but not winning combiners, the BAC of the resulting ensemble classifier drops significantly on the doubles scenario, leading to a mean BAC of 86.78% on the all scenario. This is not surprising as BC does not take into account dependencies between base-level classifiers.

The two harmonic series combiners, *Harmonic Series Weighted Voting* (HSWV) and *Dependent Harmonic Series Weighted Voting* (DHSWV), are not part of the promising combiners. The resulting ensemble classifiers perform significantly worse than the promising combiners, but still significantly better than the base-line methods SelectBest and ORACLE. HSWV achieves a mean BAC of 82.75% and DHSWV achieves a mean BAC of 75.25% on the all scenario. DHSWV performs better than HSWV on the noise, base, and doubles scenario. HSWV performs better than DHSWV on the constant and all scenario. The reason for that seems to be that DHSWV is disturbed by the constant classifiers. Because of this unclear relationship both methods will nevertheless be

included in the comparison on the real data.

The simple *Majority Voting* (MV) combiner produces a promising mean BAC of 92.17% on the base scenario. For all other scenarios, it is, not surprisingly, heavily disturbed by the noisy and depended base-level classifiers; leading to a mean BAC of 55.84% on the all scenario. The extension to the SMV combiner performs better than MV on all scenarios. With the exception of the doubles and the all scenario, it performed similar to the promising combiners. Hence, only SMV will be included in the comparison on real data sets.

Information Theoretic Combination (ITC) always performs worse or equally bad than *Random Weighted Voting* (RWV). The mean BAC of the ensemble classifier built by ITC on the all scenario is 50%. Because of that, ITC will not be included in the final comparison. ITC chooses bad base-level classifier subsets. They consist of the base-level classifier with the highest BAC and 6 base-level classifiers that perform comparatively bad. The reason for that seems to be that the information theoretic score is dominated by the information theoretic diversity.

4.4. Electroencephalography Data Sets

In this section, I will present the results on *Electroencephalography* (EEG) data sets. In addition to the main questions specified at the beginning of this chapter, I will address what feature extraction and classification methods are employed for classification. Furthermore, to access the potential of my methods, I will compare the classification accuracies of my methods to the accuracies that were achieved by other researchers on similar data sets.

I will start this section by introducing the different studies from which the data sets originate. After that, I will present the results separately for each study. The emphasis during this part is to find out if the proposed set of base-level learners is sufficient for a fair comparison of the methods and if the employment of ensemble classifiers produces more accurate classifiers than the base-line methods *Select the Best* (SelectBest), ORACLE, and *Concatenation* (CONCAT). Following this part, I will compare the different combiners on data sets originating from various studies to find out if there is a superior combiner. After that, I will apply the most promising methods on a data set on that no successful classification has been achieved yet.

4.4.1. Description of the Studies

Attention

The classification task for the data sets originating from the Attention study is to classify if the participant attends to the left or the right half of a computer screen, while looking at a fixation cross. The original results of this study, as well as a more extensive description of the experimental design, can be found in Sander et al. (2012).

The participants of the study originate from three groups, 22 children ($\mu_{\text{age}} = 11.9$, $\sigma_{\text{age}} = 0.52$, range $10 - 13$ years), 12 young adults ($\mu_{\text{age}} = 24.19$, $\sigma_{\text{age}} = 1.57$, range

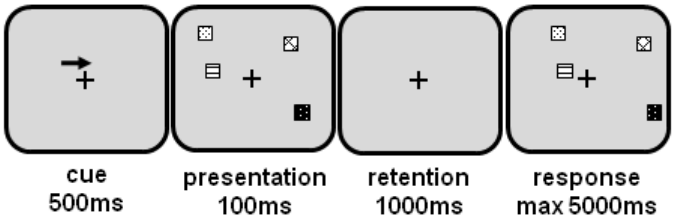


Figure 4.4.1.: Sequence of screens for one trial of the Attention study. Adapted with permission from Sander et al. (2012).

$20-26$ years), and 22 older adults ($\mu_{\text{age}} = 73.3$, $\sigma_{\text{age}} = 1.54$, range $70-75$ years).

During the experiment, the participants were seated comfortably in an electromagnetically and acoustically shielded room. They were shown a screen that displayed a fixation cross and a set of colored squares for 100ms. A cue, which was permanently shown from -500ms until 0ms relative to the presentation of the screen, indicated to which half of the screen the participants should attend. The participants were instructed to only shift their attention but to keep their visual focus on the fixation cross. After a retention interval of 1000ms, they were shown a screen that potentially differed in the half to which they were asked to attend to.

Their task was to respond if the screen differed from the screen they had seen before. The response time was limited to a maximum of 5000ms. Each participant completed 360 trials. Between the trials there was a 1500ms break, in which a fixation cross was shown.

For the comparison, the task of the classifier is to predict if a participant attends to the left or the right half of the screen, based on the EEG signals from 0 to 1000ms relative to the onset of the presentation of the to be memorized screen. Only those trials for that the response of the participants is correct are included in the analysis.

The EEG signals were recorded using 61 Ag/Ag-Cl electrodes. Electrode impedance was below 5kΩ before the recording. The sampling rate was 1000hz. During the recording, a $0.1-250$Hz band-pass filter was applied and electrodes were referenced to the right mastoid electrode, but the left mastoid electrode was also recorded.

For preprocessing the EEG signals were re-referenced to the mathematically linked mastoids and high-pass filtered with 0.5Hz. Trials that included eye movement or excessive muscle activity were removed. On the remaining data independent component analysis was used to project the residual noise sources out of the data (Jung et al., 2000). This was done by visually inspecting the components and rejecting those components that represented noise sources.

Motor Imaginary

The classification task for data sets originating from the Motor Imaginary study is to discriminate between a right and left index finger button press. Data from 36 participants are analyzed. The data sets were recorded by Zander et al. (2011). In their paper a more extensive description of the experimental paradigm can be found.

During each trial, the participants were shown a "L" or an "R" for 700ms followed by a pause of 300ms. The presentation of "L" or "R" indicated that they should press the left (L) or right (R) CTRL-key as quickly and accurately as possible with their left (L) or right index finger (R). Between the trials there was a 1000ms break.

The EEG signals were recorded using 32 Ag/Ag-Cl electrodes. The sampling rate was 1000hz. During the recording, the EEG signals were filtered using a 0.1-1000Hz band-pass filter.

As input for the base-level learners, I extract the EEG signals from -500ms up to 200ms relative to the button press. A previous approach to only extract the EEG signals from -500 to -200ms relative to the button press did not lead to accurate base-level classifiers.

Auditory Oddball

The classification task for data sets originating from the Auditory Oddball study is to classify if a participant listens to a rare or a common tone.

The data sets originate from a pilot study employed at the Max Planck Institute for Human Development. Data sets for six subjects were recorded. The experiment implemented the auditory oddball paradigm (see Squires et al., 1975).

During the experiment, the participants were standing still. The room in that they were standing was neither electromagnetically nor acoustically shielded. The participants were presented high- and low-pitched tones with varying timely gaps. The high-pitched tone was presented in 80% (common) of the cases and the low-pitched tone in the remaining 20% (rare) of the cases. The task of the participants was to count how many times the rare tone occurred. The tones were played for 50ms. The frequency of the common tone was 1000Hz and 800Hz for the rare tone. The gap between two consecutive tones was varied between 1200 and 1500ms.

The EEG signals were recorded using 60 Ag/Ag-Cl electrodes. The sampling rate was 1000hz. During the recording, electrodes were referenced to the right mastoid electrode, but the left mastoid electrode was also recorded. Furthermore, a $0.1 - 250$Hz band-pass filter was applied.

As input for the base-level learners I extract, analogously to Beckmann (2010), the EEG data from 0ms to 512ms relative to the onset of the auditory stimuli.

For preprocessing the EEG signals were re-referenced to the mathematically linked mastoids and high-pass filtered with 0.5Hz.

Memory

The classification task for the data sets originating from the memory study is to classify if the participant is able to memorize an object based on EEG data from the memorization

phase. That is, the classifier should predict if a person will be able to remember something at the time they is trying to memorize it.

For this task a data set for one subject was recorded by me and my colleagues.

The paradigm described by Brehmer et al. (2004) was used. The participant was seated comfortably in an acoustically as well as electromagnetically shielded room. The participant was presented a set of location-word pairs. The task of him was to remember the pairs. The participant was trained to fulfill this task by employing the method of loci (see Bower, 1970).

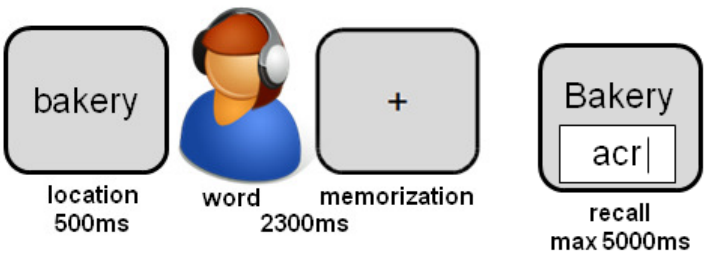
The experiment consisted of 36 blocks. Each block was separated in an encoding and a recall phase. During the encoding phase, location cues were presented visually on a monitor and the to-be-recalled words were presented aurally over headphones. For each location-word pair, first the location cue was shown for 500ms. This was followed by the presentation of the word. After that, followed a break, in which the participant should memorize the location-word pair. Then, the next location cue followed immediately. For each block 16 location-word pairs had to be remembered. In every block each location was part of exactly one pair.

After all 16 location-word pairs had been shown, the participant could start the recall phase at his own will. In the recall phase each location cue was presented for 5000ms. During the presentation of the location, the participant had to type in the first three letters of the memorized word. After successive 6 Blocks, the subjects was allowed to pause for several minutes.

If a word occurred in one block, it was guaranteed not to occur in the following block. A total of 16 locations and 413 highly imaginable words were used as stimuli. The time between the presentation of two successive locations was 2300ms. In prior sessions it was adjusted such that the participant could remember approximately 10 out of 16 pairs.

For the classification one trial consists of the EEG signals from the beginning of the location presentation until the beginning of the next location presentation. The to-be-separated classes are “the person will remember the pair” and “the person will not remember the pair”.

The EEG signals were recorded using 60 Ag/Ag-Cl electrodes. Electrode impedance was below 2kΩ before the recording. The sampling rate was 5000Hz. A $0.1 - 1000$Hz



(a) (b)

Figure 4.4.2.: Sequence of screens for the (a) encoding and the (b) recall.

band-pass filter was applied. During the recording, electrodes were referenced to the right mastoid electrode, but the left mastoid electrode was also recorded.

For preprocessing the EEG data was re-referenced to the mathematically linked mastoids and down-sampled to 500Hz. Trials that included eye movement or excessive muscle activity were removed.

4.4.2. Results

A short notational remark. Most measures employed are per data set measures. I will often report means of that measures. The number between the brackets after the number for the mean denotes the corresponding standard deviation.

Attention

One subject had to be excluded from the analysis. The training of the *Ledoit's Regularized Linear Discriminant Analysis* (LRLDA) classifier that was used for CONCAT needed more main memory than was request-able on the computing cluster (see Section 4.2).

From the 49 base-level classifiers on average 16.64(6.24) achieve an accuracy better than random guessing. The average disagreement measure between the base-level classifiers that performed better than random guessing is 0.436(0.0249). Hence, the proposed set of base-level learners produces a set of diverse and accurate base-level classifiers on the Attention data sets. Thus, an appropriate combination of the base-level classifiers is expected to result in an ensemble classifier that is more accurate than ORACLE.

In fact, the ensemble classifier built by the best combiner achieves a mean BAC of 66% and a mean rank of 7.43, while ORACLE achieves a mean BAC of 61.83% and a mean rank of 4.28. Furthermore, all combiners generate ensemble classifiers that are more accurate than ORACLE. With the exception of *fixed Adaptive Boosting* (fAB) and *Signifigance Majority Voting* (SMV), the rank differences between all combiners and ORACLE are significant.

On top of that, all combiners, with the exception of fAB and SMV, produce more accurate ensemble classifiers than SelectBest (mean BAC 64.28% rank 5.24) and CONCAT (mean BAC 64.25% rank 5.68). The ranks of the top three performing combiners, DSWMV, DHSWV and DHSWV, all proposed in this thesis, are significantly larger than the ranks of SelectBest. Testing the differences between the combiners and CONCAT for significance reveals that only the rank difference between DSWMV and CONCAT is significant.

There were two previous studies that successfully classified spatial attention based an neuroimaging data. Kelly et al. (2005) achieved a mean accuracy of 73%. Because both classes were of equal size this measure is equivalent to the BAC. While their mean BAC is 7% higher than the BAC for my best classification method, a direct comparison seems at least questionable as the subjects that participated in their study could concentrate on the attention task, while in the study from which the data sets I used originate from, the participants also had to concentrate on the memory task. Hence, it is reasonable to assume that the classification task for my data sets is more difficult.

method	*Balanced Accuracy* (BAC)	rank
DSWMV	66	7.43
HSWV	65.89	7.13
DHSWV	65.84	7.05
BC	65.33	6.51
SWMV	65.33	6.64
STLRLDA	65.21	6.41
CONCAT	64.25	5.68
SelectBest	64.28	5.25
SMV	64.06	4.97
fAB	63.91	4.66
ORACLE	61.83	4.28

Table 4.2.: Mean BACs, in percent, and ranks, for all methods, over all data sets originating from the Attention study. The methods are ordered by their mean ranks. The gray rows mark the combiners that are proposed in this thesis.

van Gerven and Jensen (2009) even classified four different directions of covert spatial attention at a reasonable classification rate using Magnetoencephalography (MEG) for signal acquisition. Because they used, instead of EEG, MEG as signal acquisition method a direct comparison seems inappropriate.

Another interesting question is: What base-level classifiers are used for the classification? To address this question I calculate the mean weights over all folds from all subjects as learned by the best method *Dependent Significant Weighted Majority Voting* (DSWMV). For each data set D_i and each fold f a weight vector $w_{i,f}$ is learned, which consists of a weight $w_{i,f}(j)$ for each base-level classifier. There are $n = 55$ data sets and 10 folds per data set. Each entry $w_\mu(j)$ of the mean vector is calculated as follows

$$w_\mu(j) = \frac{1}{10n} \sum_{i=1}^{n} \sum_{f=1}^{10} w_{i,f}(j)$$

Analogous to that, the entries of the standard deviation vector w_σ are calculated as

$$w(j)_\sigma = \sqrt{\frac{1}{10n-1} \sum_{i=1}^{n} \sum_{f=1}^{10} (w(j)_{i,f} - w(j)_\mu)^2}$$

The mean vector w_μ and the standard deviation vector w_σ of all weights are displayed in Figure 4.4.3. The models that correspond to the 10 largest entries in the mean vector w_μ can be seen in Table 4.3. While one needs to be careful when comparing the mean weights of the different base-level learners, especially because the standard deviation is comparatively high, it is interesting that from the ten base-level learners corresponding to the largest weights, nine are based on *Common Spatial Patterns* (CSP) or *Local Means* (LM) features. Especially when taking into account that previous classification of spatial

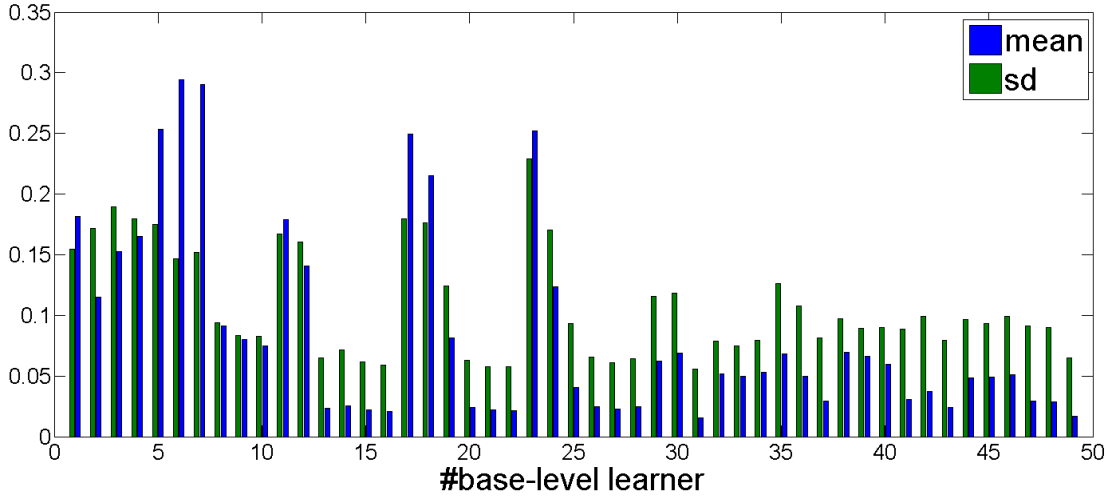


Figure 4.4.3.: Means w_μ and standard deviations w_σ of the weights for each base-level learner as learned by DSWMV over all folds from all data sets from the Attention study. The table that translates #base-level learner to the corresponding base-level learner can be found in Appendix A.1.

base-level classifier	weight
γ + CSP + LRLDA	0.2943
rem + CSP + LRLDA	0.2901
β + CSP + LRLDA	0.2532
θ + LM + SVMOPTC	0.2523
δ +LM SVMOPTC	0.2495
δ + RM + SVMOPTC	0.2149
con + CSP + LRLDA	0.1814
con +LM + SVMOPTC	0.1791
α +CSP + LRLDA	0.1652
θ +CSP + LRLDA	0.1529

Table 4.3.: Ten largest entries of the mean vector w_μ and the corresponding base-level learners.

attention (Kelly et al., 2005, van Gerven and Jensen, 2009) was solely based on α power features, this an interesting finding.

Motor Imaginary

From the 49 base-level classifiers on average 15.78(3.66) achieve an accuracy better than random guessing. The average disagreement measure between the base-level classifiers that performed better than random guessing is 0.4294(0.0201). Hence, the proposed set of base-level learners produces a set of diverse and accurate base-level classifiers on the Motor Imaginary data sets. Thus, an appropriate combination of these base level classifiers should result in an ensemble classifier that is more accurate than ORACLE.

Analogously to the Attention data sets, all combiner produce ensemble classifiers that have higher mean BACs than ORACLE. The most accurate ensemble classifier is created by STLRLDA and achieve a mean BAC of 76.45%, while the classifier selected by ORACLE achieves a mean BAC of 68.88%. With the exception of SMV, the rank differences between the ensemble classifiers and ORACLE are significant.

Also, with the exception of SMV, all combiners build ensemble classifiers that perform better than the ensemble classifier built by SelectBest (mean BAC 72.65%). Excluding *Harmonic Series Weighted Voting* (HSWV) and *Dependent Harmonic Series Weighted Voting* (DHSWV), the rank differences between all ensemble classifiers and SelectBest are significant. However, the high difference between SelectBest and HSWV, and SelectBest and DHSWV, both in mean BAC and rank, suggest that with a larger set of data sets the differences could be found to be significant.

In contrast to the Attention data sets, CONCAT clearly outperforms all other methods on the Motor Imaginary data sets. With 87.91% its mean BAC is 11.46% higher than the mean BAC of the ensemble classifier built by the best combiner STLRLDA. Furthermore, its mean rank is 11, that means that CONCAT is the method that produces the most accurate classifier on every single data set. This very pregnant difference between the Attention and the Motor Imaginary data sets will be further investigated in the following sections.

It is known that motor imaginary can be classified robustly. Indeed, it is one of the main paradigms in *Brain Computer Interface* (BCI) research. There exist classification results for exactly the same data sets as were used for this study. While Zander et al. (2011) do not report exact values, their figure suggest that CONCAT performs better than most of the classification methods they tried, and at a similar level than their best classification method.

Analogously to the Attention data sets, I also want to examine what features are employed for the classification. As CONCAT clearly is the most accurate method, I interpret the classifier built by CONCAT. To do that the weight vectors w learned by LRLDA are examined. Note that CONCAT learns a weight for each feature, contrary to DSWMV that learns a weight for each base-level classifier. The larger the deviation of a weight $w(k)$ from 0, the higher is the contribution of the corresponding feature to the classification score $w^T x + c$ (see Equation 2.2.6).

In contrast to the mean weights presented in Section 4.4.2, for each data set D_i out of the $n = 36$ data set only one weight vector w_i is obtained by applying the learning algorithm of CONCAT to the complete data set. Hence, each entry of the mean vector w_μ is calculated as

method	BAC	rank
CONCAT	87.91	11
STLRLDA	76.45	8.25
fAB	76.18	8.01
DSWMV	75.87	7.33
BC	75.37	6.47
SWMV	75.42	6.42
HSWV	74.28	5.07
DHSWV	74.32	5.03
SelectBest	72.65	3.43
SMV	72.09	2.97
ORACLE	68.88	2.01

Table 4.4.: Mean BACs, in percent, and ranks, for all methods, over all data sets originating from the Motor Imaginary study.The methods are ordered by their mean ranks. The gray rows mark the combiners that are proposed in this thesis.

$$w_\mu(k) = \frac{1}{n}\sum_{i=1}^{n} w_i(k)$$

Analogous to that, each entry of the standard deviation vector w_σ is calculated as

$$w_\sigma(k) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(w_i(k) - w_\mu(k))^2}$$

In Figure 4.4.3 w_μ and w_σ are plotted for all features.

There are eight accumulations of highly deviating weights: $S_1 = \{w_\mu(k) : k \in \{1,\ldots,42\}\}$, $S_2 = \{w_\mu(k) : k \in \{139,\ldots,778\}\}$, $S_3 = \{w_\mu(k) : k \in \{875,\ldots,1514\}\}$, $S_4 = \{w_\mu(k) : k \in \{1611,\ldots,2250\}\}$, $S_5 = \{w_\mu(k) : k \in \{2347,\ldots,2986\}\}$, $S_6 = \{w_\mu(k) : k \in \{3083,\ldots,3722\}\}$, $S_7 = \{w_\mu(k) : k \in \{3819,\ldots,4458\}\}$, and $S_8 = \{w_\mu(k) : k \in \{4555,\ldots,5194\}\}$. These accumulations are interrupted by accumulations of almost zero weights. The first set of highly deviating weights, S_1, corresponds to the CSP features calculated on all frequency bands. The next group S_2 corresponds to the *Spatio Temporal Features* (STF), LM, *Regional Means* (RM), and *Global Mean* (GM), calculated on the *con* (1-45 Hz) frequency band. The following peaks $S_3,\ldots,S_8$ correspond to the STF calculated on the δ, θ, α, β, γ and *rem* (70+ Hz) frequency bands in that order. For all features based on the *Permutation Entropy* (PE) the LRLDA algorithm consistently learned very low weights. Thus, their influence on the classification score is negligible.

For weights with a large mean the variance is also relatively high. For weights with a small mean the variance is also relatively small. This implies that, the same types of features have been employed for every subject.

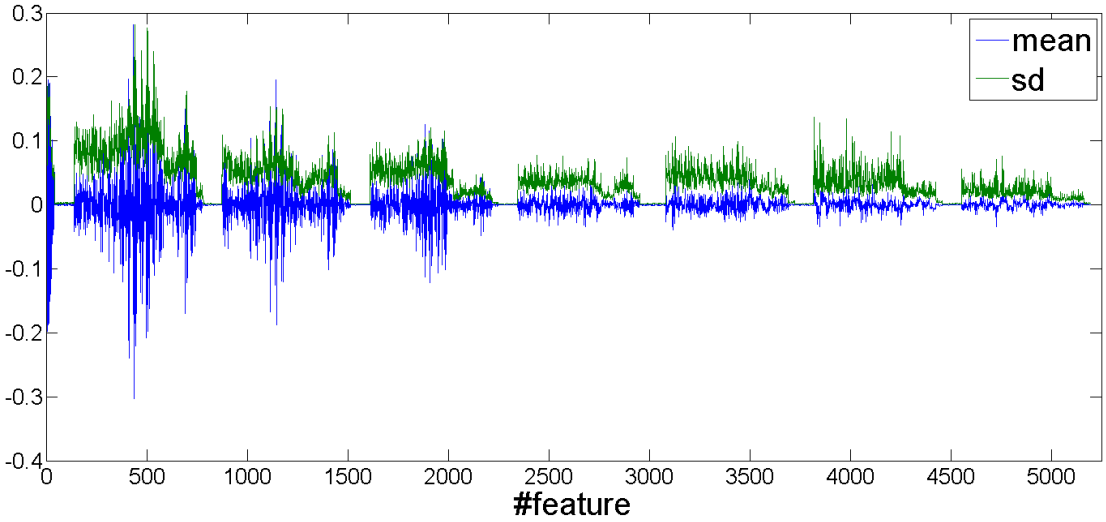


Figure 4.4.4.: Means w_μ and standard deviations w_σ of the weights for each feature as learned by CONCAT over all data sets from the Motor Imaginary study.

Auditory Oddball

From the 49 base-level classifiers on average 6.67(1.5055) achieve an accuracy better than random guessing. The average disagreement measure between the base-level classifiers that perform better than random guessing is 0.1831(0.0201). This means that only very few base-level learners produce accurate classifiers. Furthermore, these classifiers are not very diverse. Hence, a combination of base-level classifiers may be able to improve the performance, but independently of the employed combiner drastic improvements are not to be expected. The reason for this lacking diversity seems to be that for most subjects only the classifiers based on STF were more accurate than random guessing.

Considering the previous examination it is not surprising that no combiner is able to significantly improve the performance. Indeed, from the combiners, only *Bayes Combination* (BC) (mean BAC 70.21) performs a little better than ORACLE (mean BAC 69.71). Also, no improvement over SelectBest (mean BAC 69.28) is observable.

What is very interesting is that the CONCAT combination is still able to boost the accuracy significantly. CONCAT produces a mean BAC of 74.86% and an average rank of 9.83 in comparison to a mean BAC of 69.71% and a mean rank of 8.17 for ORACLE.

The classification of the *Event Related Potentials* (ERP) elicited by a rare target stimuli is one of the most popular approaches for building BCIs. Thus, it is not surprising that a successful classification is possible. The same data sets are used for the analysis as employed by Beckmann (2010). His best method achieved a mean BAC of 83%. He estimated the BAC using the holdout method. Hence, it is questionable if a direct comparison of the results is appropriate. However, it seems like his specialized method performs even better than CONCAT.

Analogously to the previous studies, I also want to examine what features have been

method	BAC	rank
CONCAT	74.86	9.83
BC	70.21	9
ORACLE	69.71	8.17
STLRLDA	69.49	8.33
SelectBest	69.28	7.67
fAB	68.98	7.83
SMV	64.2	4.33
SWMV	63.35	4.33
HSWV	61.43	3.08
DSWMV	58.49	1.66
DHSWV	53.59	1.7

Table 4.5.: Mean BACs, in percent, and ranks for all methods over all data sets originating from the Auditory Oddball study. The methods are ordered by their mean ranks. The gray rows mark the combiners that are proposed in this thesis.

used for the classification. As CONCAT clearly was the best classification method, I interpret it. I calculate the mean vector w_μ and the standard deviation vector w_σ analogously to the approach that was used for the Motor Imaginary study. The entries of the mean and the standard deviation vector are displayed in Figure 4.4.5. As the weights are basically the same as for the Motor Imaginary data sets, please refer to the interpretation presented there.

Intermediate Summary and Open Questions

For the Motor Imaginary and the Attention study, the proposed set of base-level learners was clearly sufficient and produced accurate and diverse base-level classifiers. For the Auditory Oddball data sets that was not the case. It is unclear whether this is an elementary property of the data sets or if the set of base-level learners was not sufficient.

It was shown that if the base-level classifiers are diverse and accurate, the combination of base-level classifiers clearly outperforms ORACLE. Furthermore, a fusion of the base-level classifier decisions led to more accurate ensemble classifiers than the selection of the best base-level classifier by SelectBest.

However, while the combination of base-level classifiers performed better than CONCAT on the Attention data sets, on the Motor Imaginary data sets CONCAT clearly outperformed the ensemble classifiers. When taking into account that the Attention data sets consist of $r = 5194$ features and on average $N = 608.17(19.95)$ trials, so the mean number of features per trial is $\frac{r}{N} \approx 8.54$, this is a surprising result. The original *Linear Discriminant Analysis* (LDA) algorithm fails if $\frac{r}{N} > 1$ because the estimated covariance matrix is non-invertible. Of course, the estimator of the covariance matrix used for the training of LRLDA was built such that it produces a reasonable estimate if $\frac{r}{N} > 1$, but I

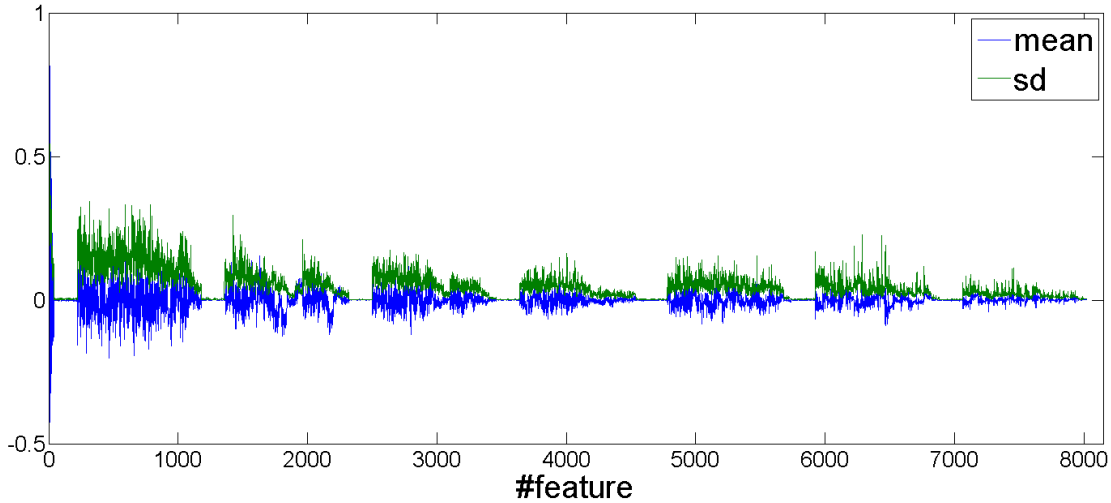


Figure 4.4.5.: Means w_μ and standard deviations w_σ of the weights for each feature as learned by CONCAT over all data sets from the Auditory Oddball study.

did not expect it to work that well if the relationship is as extreme. Also, this is inconsistent with the findings of Dornhege et al. (2004) and Boostani et al. (2007). They both found that CONCAT performs worse than ORACLE. However, none of them employed the very advanced LRLDA classification method, specifically tailored for the situation when the number of trials is small compared to the number of features.

As result of searching for differences between the Attention and the Motor Imaginary data sets, I found that the two groups of data sets mainly differ in the number of features per trial $\frac{r}{N}$. While the mean number of features per trial is 8.54 for the Motor Imaginary data sets, it is $12222/260.89 = 46.85$ for the Attention data sets. This may very well be the reason why the ensemble classifiers perform better than CONCAT on the Attention data sets.

To confirm this relationship, I rerun the analysis for the Motor Imaginary data sets and modify the set base-level learners such that the total number of features increases to 72394, resulting in an average feature per trial ratio of $72394/608.17 = 119.0358$. This is done by including three new base-level learners to the set. As features the raw amplitude EEG signals from the con (1-45 Hz), α, and β band are extracted separately.

It is not possible to employ LRLDA as classification method for CONCAT on the modified set of base-level learners. Recall that the learning algorithm of LRLDA estimates the per-class covariance matrix of the features. Hence, if the number of features is $r = 72394$, it will have to estimate $\frac{r(r+1)}{2} \approx 2.6205 \cdot 10^9$ values. Taking into account that Matlab allocates 8 Byte main memory for every entry, this results in a memory consumption of $2.0964 \cdot 10^{10}$Byte $= 20.964$GB. The maximum request-able amount of main memory on the computing cluster is smaller than 17GB. Hence, it is impossible to execute the learning algorithm of the LRLDA on the available hardware. Therefore, I used *Support Vector Machine with Optimization of the C hyper-parameter* (SVMOPTC)

as classification method for CONCAT.

The second open question is, which out of the combiners leads to the most accurate ensemble classifier. This question will be answered in Section 4.4.2.

Modified Motor Imaginary

Three subjects had to be excluded from the analysis because the time needed for calculation of the results for each fold exceeded the maximum available computing time of two days. The reason for that was that the SVMOPTC learning algorithm took several hours for the high dimensional feature vectors based on the raw EEG amplitude data. To get a result for these subjects, I employ *Support Vector Machine* (SVM) as classification method instead. The C parameter is chosen according to the standard routine of Fieldtrip if no C hyper-parameter is specified. By reading Appendix A.2, it can be confirmed that the results of these subjects do not vary substantially from the results presented here.

As expected, CONCAT completely fails on the very high dimensional features. With a mean BAC of 60.32% and a mean rank of 1.03 it performs significantly worse than every combiner. Further analysis reveals that it performs worse than every combiner on all data sets. The mean BAC achieved by the best combiner STLRLDA is 77.05. This is tremendously lower than the mean BAC achieved by CONCAT on the original set of base-level learners (87.91%).

Furthermore, all combiners perform better than ORACLE (mean BAC 68.83%). With the exception of SMV, the differences between them and ORACLE are significant.

method	BAC	rank
STLRLDA	77.05	9.79
fAB	76.28	8.70
DSWMV	76.10	8.64
BC	75.26	7.30
SWMV	75.09	6.67
HSWV	74.83	6.45
DHSWV	74.71	6.42
SelectBest	73.04	4.47
SMV	71.87	3.64
ORACLE	68.83	2.89
CONCAT	60.32	1.03

Table 4.6.: Mean BACs, in percent, and ranks for all methods over all data sets originating from the Modified Motor Imaginary study. The methods are ordered by their mean ranks. The gray rows mark the combiners that are proposed in this thesis.

Comparison of the Combiners

The comparison of the combination methods is based on the results of the Attention data sets and on the results of the set of modified base-level learners on the Motor Imaginary data sets, introduced in the previous section. For the original set of base-level learners on the Motor Imaginary data sets and the Auditory Oddball data sets, it is apparent that CONCAT performs best.

The best combiner DSWMV yields a mean BAC of 69.78%. With the exception of SMV, all combiners perform better than SelectBest (mean BAC 67.56%). The rank differences are significant for all combiners but fAB.

Overall the combiners perform very similar. With the exception of SMV, the mean BAC varies only between 68.55% for fAB and 69.78% for DSWMV. The rank differences are higher but also not very big. Comparing all combiners, excluding SelectBest, against each other reveals the following picture: While DSWMV performs best, the differences between it and STLRLDA, HSWV and BC are not significant. However, when looking at the data, especially at the ranks, it seems like DSWMV and STLRLDA are the best combiners.

The extension DSWMV of *Signifigance Weighted Majority Voting* (SWMV) performs significantly better than SWMV (mean BAC 68.99%, $p < 0.01$). DSWMV, also, performs significantly better than fAB ($p < 0.01$), indicating that the strategy that was chosen to correct the weights for dependencies (see Equation 3.3.1), which led to DSWMV, is superior to the strategy fAB uses. The extension DHSWV performs worse than the original HSWV algorithm ($p = 0.8790$).

	BAC	rank
DSWMV	69.78	6.30
STLRLDA	69.65	6.13
HSWV	69.06	5.33
BC	69.20	5.29
DHSWV	69.21	5.27
SWMV	68.99	5.17
fAB	68.55	4.69
SelectBest	67.56	3.67
SMV	66.99	3.16

Table 4.7.: Mean BACs, in percent, and ranks for all combiners over all data sets originating from the Modified Motor Imaginary study and the Attention study. The methods are ordered by their mean ranks. The gray rows mark the combiners that are proposed in this thesis.

Memory

Since the classification of a successful memorization has not yet been achieved, the purpose of this data set is not to compare the different methods but rather to use the most powerful methods to try the successful classification.

From the 49 base-level classifiers 8 achieve a BAC better than random guessing. The disagreement measure between those base-level classifiers is 0.3.

The number of features is 15582 and the number of trials 557. Hence, the number of features per trial is 27.97. Because this value lies between 8.54 and 46.85, it is unclear if CONCAT or one of the combiners should be chosen. But because of the large numbers of features, as classification method for CONCAT SVMOPTC has to be used. Because of that, I propose that the ensemble classifiers will be the superior methods and choose to include only them in the significance test against random guessing.

In Table 4.8 you can see the mean accuracies for the five most promising combiners, as identified in the previous section. The combiner out of these combiners that yields to the highest BAC is BC (58.04%). With the exception of STLRLDA and DHSWV, the BACs achieved by these combiners are significantly better than the expected BAC by

combiner	BAC
BC	58.04
HSWV	57.3
DSWMV	55.31
STLRLDA	53.52
DHSWV	53.34

Table 4.8.: BACs, in percent, for the most promising combiners on the memory data set. The gray rows mark the combiners that are proposed in this thesis.

base-level learner	BAC
rem + CSP + LRLDA	59.81
β + CSP + LRLDA	58.43
con + CSP + LRLDA	56.92
θ+ CSP + LRLDA	55.91
γ + CSP + LRLDA	55.73
δ+ CSP + LRLDA	55.12
α + CSP + LRLDA	54.33
θ+ LM + SVMOPTC	54.25

Table 4.9.: The ten most accurate base-level learners for the memory data set ordered by their corresponding BAC.

random guessing. The BAC of CONCAT is also estimated. With 48.40 it is in the area of random guessing.

Since the interpretation of an ensemble classifier built by BC is not straightforward, I report the base-level classifiers that performed better than chance instead. The BACs for the ten more accurate base-level learners can be seen in Table 4.9. All nine base-level learners that are based on CSP features are included in the set of the ten base-level learners that produce the most accurate base-level classifiers.

With 59.81% ORACLE performs better than any ensemble classifier. It is important to note that, in contrast to the other data sets, only one data set is available from the memory study. Thus, when picking the best single classifier after having evaluated the accuracies the statistical advantage of ensemble classifiers vanishes (see Section 2.3.3). Furthermore, comparing the best base-level classifier against the best combiners is a biased comparison. There are 49 base-level classifiers and only five combiners. Hence, there is a statistical advantage for the base-level classifiers. A fair comparison is the comparison against SelectBest. SelectBest produces an ensemble classifier with a BAC of 56.57%.

4.5. Summary

With the exception of the Auditory Oddball data sets, the proposed set of base-level learners produced accurate and diverse base-level classifiers. It was, thus, suited for a fair comparison of the several methods.

The combination of base-level classifiers based on different feature and classification methods produced significantly more accurate classifiers than ORACLE. Also, the true combination of base-level classifiers produced ensemble classifiers that had higher *Balanced Accuracy*s (BACs) than the ensemble classifiers created by *Select the Best* (SelectBest).

The comparison against *Concatenation* (CONCAT) revealed that CONCAT produces more accurate classifiers when the number of features per trial is relatively low and that the ensemble classifiers generate more accurate classifiers when the number of features

per trial is relatively high. When the number of features per trials was smaller than 8.54, the classifier built by CONCAT was more accurate than the classifier induced by any combiner. When the number of features per trial was larger than 30, the ensemble classifiers were more accurate than CONCAT.

Out of the combiners DSWMV, which was proposed in this thesis, produced the most accurate ensemble classifiers on the *Electroencephalography* (EEG) data sets. The BAC differences between it and the remaining combiners was, with the exception of *Stacking with Ledoit's Regularized Linear Discriminant Analysis* (STLRLDA), *Harmonic Series Weighted Voting* (HSWV), and *Bayes Combination* (BC) significant .

On the simulation data sets STLRLDA induced the most accurate classifier, while *Dependent Significant Weighted Majority Voting* (DSWMV) followed on the 3rd rank. It is interesting that *fixed Adaptive Boosting* (fAB), which was the second best combiner in the simulation study, was the second worst combiner on the EEG data sets. In the other direction HSWV performed relatively bad on the simulation data sets but achieved the shared first rank on the real data sets.

For all presented data sets, one of the compared classification methods was able to infer a separating model. Furthermore, the best combiners could be employed to successfully classify if a person memorizes something based on the EEG signals during the encoding phase. This is the first proof of concept for this classification task.

5. Summary, Conclusion and Outlook

5.1. Summary and Conclusion

The main hypothesis of this thesis was (see Chapter 3) that the combination of the different feature extraction (see Section 2.2.3) and classification methods (see Section 2.2.4) that are employed for the classification of *Electroencephalography* (EEG) signals leads to a more accurate classifier than the best classifier based on only one combination of feature extraction and classifier method, estimated by the ORACLE classifier. ORACLE returned the classifier that achieved the best mean accuracy after having evaluated a set of classifiers on all data sets from one *Pattern Recognition* (PR) task. A further proposition was that this results in a classification method that achieves good classification accuracies on a variety of EEG data sets resulting in a very powerful EEG single-trial analysis tool. Furthermore, it was proposed that the combination of the classification and feature extraction methods through a *Multiple Classifier System* (MCS) (see Section 2.3) leads to a more accurate classifier than the employment of a single classification method on the concatenation of the outputs of all feature extraction methods. The latter approach was called *Concatenation* (CONCAT) throughout this thesis. The last hypothesis was that a combination of the classifiers leads to a more accurate ensemble classifier than the selection of the best classifier, by *Select the Best* (SelectBest) (see Section 2.3.2.7).

To examine this hypotheses, the aforementioned methods were compared on a number of data sets originating from four different EEG studies. To examine if an ensemble classifier is superior, a set of base-level classifiers was defined. The set was chosen to consist of base-level classifiers that extract different characteristics of the EEG signals (see Section 3.4). Multiple different combiners were employed to generate multiple ensemble classifiers based on the defined set of base-level classifiers. In addition to well known combiners, new combiners were introduced, implemented, and evaluated (see Section 3.3). Because the combiners used the same set of base-level classifiers to build the ensemble classifier, no further analysis was required to determine which combiner results in the most accurate ensemble classifier.

The combination of base-level classifiers, for example, by *Stacking with Ledoit's Regularized Linear Discriminant Analysis* (STLRLDA) and *Dependent Significant Weighted Majority Voting* (DSWMV), boosted the *Balanced Accuracy* (BAC) compared to ORACLE by up to 7.57% (see Chapter 4). Furthermore, the combination of the base-level classifiers by DSWMV led to an increase of the mean BAC of 2.22% compared to the selection of the most accurate base-level classifier by SelectBest. When comparing the MCSs against CONCAT, the picture is not as clear. The MCSs had an clear advantage when the number of features per trial was larger than 30. Contrary to that, CONCAT performed substantially better than any MCS if the number of features per trial was

smaller than 9.

The results suggest (see Section 4.4.2) that, out of the employed combiners (see Section 3.3), *Dependent Significant Weighted Majority Voting* (DSWMV) and *Stacking with Ledoit's Regularized Linear Discriminant Analysis* (STLRLDA), from which DSWMV was proposed in this thesis, are the best combiners for heterogeneous classifiers. It is worth noting that, DSWMV produced significantly more accurate ensemble classifiers than the two famous and powerful combiners *Adaptive Boosting* (AB) and *Weighted Majority Voting* (WMV).

In combination with the the proposed set of base-level learners, the best combiners and CONCAT can be used as a very powerful single-trial analysis tool. It was shown that these methods are able to infer a separating model (classifier) for a variety of different EEG data sets. Also, it was shown how to interpret these models (see Chapter 4). The *Pattern Recognition Systems* (PRSs) presented in this thesis are the first PRSs that have been shown to be able to infer separating models on more than one type of EEG data sets.

The classification task for the memory data set was to classify if the participant will remember a location-word pair based on the EEG signals during the encoding (see Section 4.4.1). The best combiners were employed to create ensemble classifiers that achieve a BAC of 58.04% on the memory data set, which is significantly better than the BAC expected by random guessing (see Section 4.4.2). This represents the first proof of concept that it is possible to classify if somebody will remember something at the time they is trying to memorize it.

Overall, the results imply that the general direction in EEG classification research should be changed from "finding the best single classification method" to "finding the best combination of classification methods".

5.2. Outlook

Another very popular approach to deal with high dimensional feature vectors is to perform feature selection before the data set is fed to the classification method. It would be interesting to compare the performance of a PRS that employs feature selection, e.g., Boostani et al. (2007), against the MCS built by the best combiners on the data sets with a number of features per trial of greater than 30. The employment of a MCS may also be a better strategy when features are combined that originate from the same feature extraction method but are extracted on different time intervals. This procedure is very often employed as feature extraction method for *Brain Computer Interfaces* (BCIs). In this case, for each interval a separate base-level classifier could be induced. Indeed, the employment of MCS could be superior in any case where the feature space is large compared to the numbers of trials.

My results suggest that the regularization performed by MCSs is beneficial compared to the regularization performed by *Ledoit's Regularized Linear Discriminant Analysis* (LRLDA) if the number of features per trial is above 30. A further investigation when and under what circumstances which regularization is appropriate could be informative.

Another interesting question is if it is possible to further increase the accuracy by including more base-level learners. There are various popular feature extraction methods that have not been employed in this thesis, such as Autoregressive models (Dornhege et al., 2004), Power Spectral Densities, Adaptive Autoregressive Parameters (Lotte et al., 2007, and references within), Common Sparse Spectral Spatial Pattern (Dornhege et al., 2006), and Regularized Common Spatial Patterns (Lotte and Guan, 2011).

What might also have great potential is the combination of the a-posteriori likelihoods, as used by Dornhege et al. (2004) (see Section 3.1.1), with the DSWMV combiner. In that way not only the accuracies of the base-level classifier are considered, but also the a-posteriori likelihood for each class. Another very interesting approach is the employment of a trainable combiner, as introduced by Sun (2007) (see Section 3.1.3). It should by examined if the resulting ensemble classifiers gets more accurate, when the normalized mutual information is used as estimate for the dependency between two base-level classifiers, instead of the mutual information. Motivated by the fact that DSWMV was the best combiner, it should definitely be examined if popular methods that automatically generate the base-level classifiers, such as AB, Bagging and Random Subspace, can be improved by employing DSWMV as combiner.

The finding that it is possible to classify if a person will remember something at the time the person is trying to memorize it has definitely to be further investigated. If the accuracy of such a classifier could be increased, a cheap and mobile EEG system could become a revolutionary tool for the study and practice of learning. For example, a device could be worn by students to alert them when they have successfully memorized an equation.

It is unclear if the proposed PRSs can be used as BCIs because the real-time capabilities were not tested. Thus, it is worthwhile to examine if the proposed PRSs can be used as BCIs and provide a higher information transfer rate than the existing BCIs.

6. List of Abbreviations

AB *Adaptive Boosting*, see Section 2.3.2.3

BAC *Balanced Accuracy*, see Definition 13

BC *Bayes Combination*, see Section 2.3.2.4

BCI *Brain Computer Interface*, see Section 2.2.2

CONCAT *Concatenation*, see chapter 3

CSP *Common Spatial Patterns*, see Section 2.2.3

DHSWV *Dependent Harmonic Series Weighted Voting*, see Section 3.3.3

DSWMV *Dependent Significant Weighted Majority Voting*, see Section 3.3.2

DWMV *Dependent Weighted Majority Voting*, see Section 3.3.2

EEG *Electroencephalography*, see Section 2.2.1

ERP *Event Related Potentials*, see Section 2.2.1

fAB *fixed Adaptive Boosting*, see Section 2.3.2.3

GM *Global Mean*, see Section 3.4

HSWV *Harmonic Series Weighted Voting*, see Section 3.3.3

ITC *Information Theoretic Combination*, see Section 2.3.2.6

k-NN *k-Nearest Neighbor*, see Section 2.2.4

LDA *Linear Discriminant Analysis*, see Section 2.2.4

LM *Local Means*, see Section 3.4

LRLDA *Ledoit's Regularized Linear Discriminant Analysis*, see Section 2.2.4

MCS *Multiple Classifier System*, see Section 2.3

MV *Majority Voting*, see Section 2.3.2.1

PE *Permutation Entropy*, see Section 2.2.3

PR *Pattern Recognition*, see Section 2.1

PRS *Pattern Recognition System*, see Section 2.1

RLDA *Regularized Linear Discriminant Analysis*, see Section 2.2.4

RM *Regional Means*, see Section 3.4

RWV *Random Weighted Voting*, see Section 3.3.4

SelectBest *Select the Best*, see Section 2.3.2.7

SSP *Shaffer's Static Procedure*, see Section 2.1.4.2

STF *Spatio Temporal Features*, see Section 2.2.3

STLDA *Stacking with Linear Discriminant Analysis*, see Section 3.3.5

STLRLDA *Stacking with Ledoit's Regularized Linear Discriminant Analysis*, see Section 3.3.5

SVMOPTC *Support Vector Machine with Optimization of the C hyper-parameter*, see Section 3.4

SMV *Signifigance Majority Voting*, see Section 3.3.1

SVM *Support Vector Machine*, see Section 2.2.4

SWMV *Signifigance Weighted Majority Voting*, see Section 3.3.1

WMV *Weighted Majority Voting*, see Section 2.3.2.2

WV *Weighted Voting*, see Section 2.3.2.2

Bibliography

Autonomos Labs. Braindriver - a mind controlled car, feb 2011. URL `http://autonomos.inf.fu-berlin.de/subprojects/braindriver`. "[Accessed Feb. 6, 2012]".

C. Bandt and B. Pompe. Permutation entropy: A natural complexity measure for time series. *Physical Review Letters*, 88:174102, apr 2002. doi: 10.1103/PhysRevLett.88.174102.

M. Beckmann. Klassifikation von elektroenzephalographischen daten in oddball- und gedächtnisdaten. Master's thesis, Free University Berlin, Berlin, 2010.

B. Blankertz, K.-R. Muller, D. Krusienski, G. Schalk, J. Wolpaw, A. Schlogl, G. Pfurtscheller, J. Millan, M. Schroder, and N. Birbaumer. The bci competition iii: validating alternative approaches to actual bci problems. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 14(2):153–159, jun 2006. doi: 10.1109/TNSRE.2006.875642.

B. Blankertz, G. Dornhege, M. Krauledat, V. Kunzmann, F. Losch, G. Curio, and K.-R. Müller. The berlin brain-computer interface: Machine learning-based detection of user specific brain states. In G. Dornhege, J. d. R. Millán, T. Hinterberger, D. J. McFarland, and K.-R. Müller, editors, *Toward Brain-Computer Interfacing*, chapter 5, pages 85–102. The MIT Press, London, 2007.

B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, and K.-R. Muller. Optimizing spatial filters for robust eeg single-trial analysis. *Signal Processing Magazine, IEEE*, 25(1):41–56, 2008. doi: 10.1109/MSP.2008.4408441.

B. Blankertz, M. Tangermann, C. Vidaurre, T. Dickhaus, C. Sannelli, F. Popescu, S. Fazli, M. Danóczy, G. Curio, and K.-R. Müller. Detecting mental states by machine learning techniques: The berlin brain-computer interface. In B. Graimann, G. Pfurtscheller, and B. Allison, editors, *Brain-Computer Interfaces*, The Frontiers Collection, pages 113–135. Springer, Berlin Heidelberg, 2010a. doi: 10.1007/978-3-642-02091-9_7.

B. Blankertz, M. Tangermann, C. Vidaurre, S. Fazli, C. Sannelli, S. Haufe, C. Maeder, L. E. Ramsey, I. Sturm, G. Curio, and K. R. Mueller. The berlin brain-computer interface: Non-medical uses of bci technology. *Frontiers in Neuroscience*, 4(00198), 2010b. doi: 10.3389/fnins.2010.00198.

B. Blankertz, S. Lemm, M. Treder, S. Haufe, and K.-R. Müller. Single-trial analysis and classification of erp components - a tutorial. *NeuroImage*, 56(2):814 – 825, 2011. doi: 10.1016/j.neuroimage.2010.06.048.

R. Boostani and M. H. Moradi. A new approach in the bci research based on fractal dimension as feature and adaboost as classifier. *Journal of Neural Engineering*, 1(4): 212–217, 2004. doi: 10.1088/1741-2560/1/4/004.

R. Boostani, B. Graimann, M. Moradi, and G. Pfurtscheller. A comparison approach toward finding the best feature and classifier in cue-based bci. *Medical and Biological Engineering and Computing*, 45:403–412, 2007. doi: 10.1007/s11517-007-0169-y.

G. H. Bower. Analysis of a mnemonic device: Modern psychology uncovers the powerful components of an ancient system for improving memory. *American Scientist*, 58(5): 496–510, 1970. URL `http://www.jstor.org/stable/27829239`.

Brain Products, 2012. URL `http://www.brainproducts.com/downloads.php?kid=4`. "[Accessed Apr. 13, 2012]".

A. M. Brandmaier. *Permutation Distribution Clustering and Structural Equation Model Trees*. Dissertation, Saarland University, Saarbrücken, 2012. URL `http://scidok.sulb.uni-saarland.de/volltexte/2012/4545`.

Y. Brehmer, V. Müller, T. von Oertzen, and U. Lindenberger. Episodic memory in childhood and old age: The role of cortical coherence. In A. Mecklinger, H. Zimmer, and U. Lindenberger, editors, *Bound in memory. Insights from behavioral and neuropsychological studies*, Berichte aus der Psychologie, pages 69–91. Shaker Verlag, Aachen, 2004.

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996. doi: 10.1007/BF00058655.

K. Brodersen, C. S. Ong, K. Stephan, and J. Buhmann. The balanced accuracy and its posterior distribution. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 3121–3124, aug 2010. doi: 10.1109/ICPR.2010.764.

R. Bryll, R. Gutierrez-Osuna, and F. Quek. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognition*, 36(6): 1291–1302, 2003. doi: 10.1016/S0031-3203(02)00121-8.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, sep 1995. doi: 10.1023/A:1022627411411.

A. Delorme and S. Makeig. Eeglab: an open source toolbox for analysis of single-trial eeg dynamics including independent component analysis. *Journal of neuroscience methods*, 134(1):9–21, 2004. doi: 10.1016/j.jneumeth.2003.10.009.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006. URL `http://jmlr.csail.mit.edu/papers/volume7/demsar06a/demsar06a.pdf`.

T. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer, Berlin Heidelberg, 2000. doi: 10.1007/3-540-45014-9_1.

G. Dornhege, B. Blankertz, G. Curio, and K.-R. Muller. Boosting bit rates in noninvasive eeg single-trial classifications by feature combination and multiclass paradigms. *Biomedical Engineering, IEEE Transactions on*, 51(6):993–1002, jun 2004. doi: 10.1109/TBME.2004.827088.

G. Dornhege, B. Blankertz, M. Krauledat, F. Losch, G. Curio, and K.-R. Müller. Optimizing spatio-temporal filters for improving brain-computer interfacing. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 315–322. MIT Press, Cambridge, 2006.

R. O. Duda, P. E. Hart, and D. G. Storck. *Pattern Classification*. Wiley-Blackwell, New York, 2 edition, nov 2000.

M. Fatourechi, R. K. Ward, and G. E. Birch. A self-paced brain-computer interface system with a low false positive rate. *Journal of Neural Engineering*, 5(1):9, 2008. doi: 10.1088/1741-2560/5/1/002.

S. Fazli, F. Popescu, M. Danóczy, B. Blankertz, K.-R. Müller, and C. Grozea. Subject-independent mental state classification in single trials. *Neural Networks*, 22(9-23): 1305–1312, 2009. doi: 10.1016/j.neunet.2009.06.003.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. doi: 10.1006/jcss.1997.1504.

J. H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989. URL `http://www.jstor.org/stable/2289860`.

K. Fukunaga. *Introduction to statistical pattern recognition*. Academic Press, San Diego, 2 edition, 1990.

S. García and F. Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008. URL `http://jmlr.csail.mit.edu/papers/volume9/garcia08a/garcia08a.pdf`.

L. Hansen and P. Salamon. Neural network ensembles. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(10):993 –1001, oct 1990. doi: 10.1109/34.58871.

C. S. Herrmann, M. Grigutsch, and N. A. Busch. Eeg oscillations and wavelet analysis. In T. C. Handy, editor, *Event-related Potentials : A Methods Handbook*. The MIT Press, Cambridge, oct 2004.

T. Hinterberger, F. Nijboer, A. Kübler, T. Matuz, A. Furdea, U. Mochty, M. Jordan, T. N. Lal, N. J. Hill, J. Mellinger, M. Bensch, M. Tangermann, G. Widman, C. E. Eigler, W. Rosenstiel, B. Schölkopf, and N. Birbaumer. Brain-computer interfaces for communication in paralysis: A clinical experimental approach. In G. Dornhege, J. d. R. Millán, T. Hinterberger, D. J. McFarland, and K.-R. Müller, editors, *Toward Brain-Computer Interfacing*, chapter 3, pages 43–64. The MIT Press, London, 2007.

O. Jensen, A. Bahramisharif, R. Oostenveld, S. Klanke, A. Hadjipapas, Y. O. Okazaki, and M. A. J. van Gerven. Using brain-computer interfaces and brain-state dependent stimulation as tools in cognitive neuroscience. *Frontiers in Psychology*, 2(00100), 2011. doi: 10.3389/fpsyg.2011.00100.

T.-P. Jung, S. Makeig, C. Humphries, T.-W. Lee, M. J. McKeown, V. Iragui, and T. J. Sejnowski. Removing electroencephalographic artifacts by blind source separation. *Psychophysiology*, 37(2):163–178, 2000. doi: 10.1111/1469-8986.3720163.

S. Kelly, E. Lalor, R. Reilly, and J. Foxe. Independent brain computer interface control using visual spatial attention-dependent modulations of parieto-occipital alpha. In *Neural Engineering, 2005. Conference Proceedings. 2nd International IEEE EMBS Conference on*, pages 667–670, mar 2005. doi: 10.1109/CNE.2005.1419713.

R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on Artificial intelligence*, volume 2, pages 1137–1143. Morgan Kaufmann Publishers Inc., 1995. URL `http://robotics.stanford.edu/~ronnyk/accEst.pdf`.

A. Kübler, F. Nijboer, and N. Birbaumer. Brain-computer interfaces for communication and motor control - perspectives on clinical application. In G. Dornhege, J. d. R. Millán, T. Hinterberger, D. J. McFarland, and K.-R. Müller, editors, *Toward Brain-Computer Interfacing*, chapter 22, pages 373–392. The MIT Press, London, 2007.

L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Blackwell, Hoboken, 2004.

O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411, 2004. doi: 10.1016/S0047-259X(03)00096-4.

F. Lotte and C. Guan. Regularizing common spatial patterns to improve bci designs: Unified theory and new algorithms. *Biomedical Engineering, IEEE Transactions on*, 58(2):355–362, feb 2011. doi: 10.1109/TBME.2010.2082539.

F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, and B. Arnaldi. A review of classification algorithms for eeg-based brain-computer interfaces. *Journal of Neural Engineering*, 4 (2):R1–R13, 2007. doi: 10.1088/1741-2560/4/2/R01.

MathWorks. Matlab, 2012. URL `http://www.mathworks.com/products/matlab/`. "[Accessed Mar. 5, 2012]".

J. Meynet and J.-P. Thiran. Information theoretic combination of pattern classifiers. *Pattern Recognition*, 43(10):3412–3421, 2010. doi: 10.1016/j.patcog.2010.04.013.

A. Ng. Support vector machines. Lecture, 2011. URL `http://cs229.stanford.edu/notes/cs229-notes3.pdf`.

N. Nicolaou and J. Georgiou. Permutation entropy: A new feature for brain-computer interfaces. In *Biomedical Circuits and Systems Conference (BioCAS), 2010 IEEE*, pages 49–52, nov 2010. doi: 10.1109/BIOCAS.2010.5709568.

R. Oostenveld, P. Fries, E. Maris, and J.-M. Schoffelen. Fieldtrip: Open source software for advanced analysis of meg, eeg, and invasive electrophysiological data. *Computational Intelligence and Neuroscience*, 2011, 2011. doi: 10.1155/2011/156869.

L. Parra, C. Alvino, A. Tang, B. Pearlmutter, N. Yeung, A. Osman, and P. Sajda. Linear spatial integration for single-trial detection in encephalography. *NeuroImage*, 17(1): 223–230, 2002. doi: 10.1006/nimg.2002.1212.

G. Pfurtscheller and C. Neuper. Motor imagery and direct brain-computer communication. *Proceedings of the IEEE*, 89(7):1123–1134, jul 2001. doi: 10.1109/5.939829.

G. Pfurtscheller, G. R. Müller-Putz, A. Schlögl, B. Graimann, R. Scherer, R. Leeb, C. Brunner, C. Keinrath, G. Townsend, C. Vidaurre, M. Naeem, F. Y. Lee, S. Wriesnegger, D. Zimmermann, E. Höfler, and C. Neuper. Graz-brain-computer interface: State of research. In G. Dornhege, J. d. R. Millán, T. Hinterberger, D. J. McFarland, and K.-R. Müller, editors, *Toward Brain-Computer Interfacing*, chapter 4, pages 65–84. The MIT Press, London, 2007.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc., San Mateo, 1 edition, dec 1992.

A. Rakotomamonjy and V. Guigue. Bci competition iii: Dataset ii- ensemble of svms for bci p300 speller. *Biomedical Engineering, IEEE Transactions on*, 55(3):1147–1154, mar 2008. doi: 10.1109/TBME.2008.915728.

H. Ramoser, J. Muller-Gerking, and G. Pfurtscheller. Optimal spatial filtering of single trial eeg during imagined hand movement. *Rehabilitation Engineering, IEEE Transactions on*, 8(4):441–446, dec 2000. doi: 10.1109/86.895946.

R. Rojas. *Neural Networks: A Systematic Introduction*. Springer, Berlin, 1996.

R. Rojas. Adaboost and the super bowl of classifiers a tutorial introduction to adaptive boosting, 2009. URL `http://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/documents/tutorials/adaboost4.pdf`.

S. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1:317–328, 1997. doi: 10.1023/A:1009752403260.

M. C. Sander, M. Werkle-Bergner, and U. Lindenberger. Amplitude modulations and inter-trial phase stability of alpha-oscillations differentially reflect working memory constraints across the lifespan. *NeuroImage*, 59(1):646–654, 2012. doi: 10.1016/j.neuroimage.2011.06.092.

F. Schilling. Entwurf eines verteilten systems für die klassifikation multivariater zeitreihen im psychophysiologischen kontext. Diplomarbeit, Freie Universität Berlin, Berlin, feb 2012.

A. Schlogl and C. Brunner. Biosig: A free and open source software library for bci research. *Computer*, 41(10):44–50, oct 2008. doi: 10.1109/MC.2008.407.

E. W. Sellers, D. J. Krusienski, D. J. McFarland, and J. R. Wolpaw. Noninvasive brain-computer interfaces at the wadsworth center. In G. Dornhege, J. d. R. Millán, T. Hinterberger, D. J. McFarland, and K.-R. Müller, editors, *Toward Brain-Computer Interfacing*, chapter 2, pages 31–42. The MIT Press, London, 2007.

J. P. Shaffer. Modified sequentially rejective multiple test procedures. *Journal of the American Statistical Association*, 81(395):826–831, 1986. URL `http://www.jstor.org/stable/2289016`.

L. Shapley and B. Grofman. Optimizing group judgmental accuracy in the presence of interdependencies. *Public Choice*, 43:329–343, 1984. doi: 10.1007/BF00118940.

B. A. Shenoi. *Introduction to Digital Signal Processing and Filter Design*. Wiley-Blackwell, Hoboken, 1 edition, nov 2005.

N. K. Squires, K. C. Squires, and S. A. Hillyard. Two varieties of long-latency positive waves evoked by unpredictable auditory stimuli in man. *Electroencephalography and Clinical Neurophysiology*, 38(4):387–401, 1975. doi: 10.1016/0013-4694(75)90263-1.

S. Sun. An improved random subspace method and its application to eeg signal classification. In M. Haindl, J. Kittler, and F. Roli, editors, *Multiple Classifier Systems*, volume 4472 of *Lecture Notes in Computer Science*, pages 103–112. Springer, Berlin Heidelberg, 2007. doi: 10.1007/978-3-540-72523-7_11.

S. Sun, C. Zhang, and D. Zhang. An experimental evaluation of ensemble methods for eeg signal classification. *Pattern Recognition Letters*, 28(15):2157–2163, 2007. doi: 10.1016/j.patrec.2007.06.018.

M. W. Tangermann, M. Krauledat, K. Grzeska, M. Sagebaum, B. Blankertz, C. Vidaurre, and K.-R. Müller. Playing pinball with non-invasive bci. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1641–1648. 2009.

D. M. Titterington, G. D. Murray, L. S. Murray, D. J. Spiegelhalter, A. M. Skene, J. D. F. Habbema, and G. J. Gelpke. Comparison of discrimination techniques applied

to a complex data set of head injured patients. *Journal of the Royal Statistical Society. Series A (General)*, 144(2):145–175, 1981. URL `http://www.jstor.org/stable/2981918`.

M. van Gerven and O. Jensen. Attention modulations of posterior alpha as a control signal for two-dimensional brain-computer interfaces. *Journal of Neuroscience Methods*, 179 (1):78–84, 2009. doi: 10.1016/j.jneumeth.2009.01.016.

M. van Gerven, C. Hesse, O. Jensen, and T. Heskes. Interpreting single trial data using groupwise regularisation. *NeuroImage*, 46(3):665–676, 2009. doi: 10.1016/j.neuroimage.2009.02.041.

T. von Oertzen. Linear discriminant analysis. Lecture, 2011. URL `https://collab.itc.virginia.edu/access/content/group/0df0ea94-0905-4978-93ba-6d504f23f2ee/slides/MachineLearning%20Classifier%20V.pdf`.

T. O. Zander and C. Kothe. Towards passive brain-computer interfaces: applying brain-computer interface technology to human-machine systems in general. *Journal of Neural Engineering*, 8(2):025005, 2011. doi: 10.1088/1741-2560/8/2/025005.

T. O. Zander, K. Ihme, M. Gärtner, and M. Rötting. A public data hub for benchmarking common brain-computer interface algorithms. *Journal of Neural Engineering*, 8(2): 025021, 2011. doi: 10.1088/1741-2560/8/2/025021.

A. Result Section Appendices

A.1. Complete List of Base-Level Learners

number	base-level learner
1	con + CSP + LRLDA
2	δ + CSP + LRLDA
3	θ +CSP + LRLDA
4	α +CSP + LRLDA
5	β +CSP + LRLDA
6	γ +CSP + LRLDA
7	rem +CSP + LRLDA
8	con + PE3 + SVMOPTC
9	con + PE4 + SVMOPTC
10	con + PE5 + SVMOPTC
11	con + LM + SVMOPTC
12	con + RM + SVMOPTC
13	con + GM + SVMOPTC
14	δ + PE3 + SVMOPTC
15	δ + PE4 + SVMOPTC
16	δ + PE5 + SVMOPTC
17	δ + LM + SVMOPTC
18	δ + RM + SVMOPTC
19	δ + GM + SVMOPTC
20	θ + PE3 + SVMOPTC
21	θ + PE4 + SVMOPTC
22	θ + PE5 + SVMOPTC
23	θ + LM + SVMOPTC
24	θ + RM + SVMOPTC
25	θ + GM + SVMOPTC

number	base-level learner
26	α + PE3 + SVMOPTC
27	α + PE4 + SVMOPTC
28	α + PE5 + SVMOPTC
29	α + LM + SVMOPTC
30	α + RM + SVMOPTC
31	α + GM + SVMOPTC
32	β + PE3 + SVMOPTC
33	β + PE4 + SVMOPTC
34	β + PE5 + SVMOPTC
35	β + LM + SVMOPTC
36	β + RM + SVMOPTC
37	β + GM + SVMOPTC
38	γ + PE3 + SVMOPTC
39	γ + PE4 + SVMOPTC
40	γ + PE5 + SVMOPTC
41	γ + LM + SVMOPTC
42	γ + RM + SVMOPTC
43	γ + GM + SVMOPTC
44	rem + PE3 + SVMOPTC
45	rem + PE4 + SVMOPTC
46	rem + PE5 + SVMOPTC
47	rem + LM + SVMOPTC
48	rem + RM + SVMOPTC
49	rem + GM + SVMOPTC

A.2. Results: Left out Subjects From the Modified Motor Imaginary Data Sets

method	mean	rank
STLRLDA	71.72	9.33
DSWMV	70.91	8.83
AB	71.53	8
BC	70.87	8
SWMV	70.54	6.83
DHSWV	69.64	5.67
ORACLE	69.14	5.33
SelectBest	68.30	5
MV	69.64	5
HSWV	69.09	3
CONCAT	55.35	1