MACHINE LEARNING TECHNIQUES FOR BRAIN-COMPUTER INTERFACES

K.-R. Müller[1,2]*, M. Krauledat*[1,2]*, G. Dornhege*[1]*, G. Curio*[3]*, B. Blankertz*[1]

[1] Fraunhofer FIRST.IDA, Kekuléstr. 7, 12 489 Berlin, Germany
[2] University of Potsdam, August-Bebel-Str. 89, 14 482 Potsdam, Germany
[3] Dept. of Neurology, Campus Benjamin Franklin, Charité University Medicine Berlin,
Hindenburgdamm 30, 12 203 Berlin, Germany

ABSTRACT

This review discusses machine learning methods and their application to Brain-Computer Interfacing. A particular focus is placed on feature selection. We also point out common flaws when validating machine learning methods in the context of BCI. Finally we provide a brief overview on the Berlin-Brain Computer Interface (BBCI).

1. INTRODUCTION

Brain-Computer Interfacing is an interesting, active and highly interdisciplinary research topic ([3, 4, 5, 6]) at the interface between medicine, psychology, neurology, rehabilitation engineering, man-machine interaction, machine learning and signal processing. A BCI could, e.g., allow a paralyzed patient to convey her/his intentions to a computer application. From the perspective of man-machine interaction research, the communication channel from a healthy human's brain to a computer has not yet been subject to intensive exploration, however it has potential, e.g., to speed up reaction times, cf. [7] or to supply a better understanding of a human operator's mental states.

Classical BCI technology has been mainly relying on the adaptability of the human brain to biofeedback, i.e., a subject learns the mental states required to be understood by the machines, an endeavour that can take months until it reliably works [8, 9].

The Berlin Brain-Computer Interface (BBCI) pursues another objective in this respect, i.e., to impose the main load of the learning task on the 'learning machine', which also holds the potential of adapting to specific tasks and changing environments given that suitable machine learning (e.g. [2]) and adaptive signal processing (e.g. [10]) algorithms are used. Short training times, however, imply the challenge that only few data samples are available for learning to characterize the individual brain states to be distinguished. In particular when dealing with few samples of data (trials of the training session) in a high-dimensional feature space (multi-channel EEG, typically several features per channel), overfitting needs to be avoided. It is in this high dimensional – small sample statistics scenario where modern machine learning can prove its strength.

The present review introduces basic concepts of machine learning, which includes a discussion of common linear classification and the idea of classifying in kernel feature spaces. Classification is linked to the interesting task of robust feature selection. Finally, we briefly describe our BBCI activities where some of the discussed machine learning ideas come to an application and conclude. Note that we do not attempt a full treatment of all available literature, rather, we present a somewhat biased point of view illustrating the main ideas by drawing mainly from the work of the authors and providing – to the best of our knowledge – reference to related work for further reading. We hope that it nevertheless will be useful for the reader.

2. LEARNING TO CLASSIFY – SOME THEORETICAL BACKGROUND

Let us start with a general notion of the learning problems that we consider in this paper. The task of classification is to find a rule, which, based on external observations, assigns an object to one of several classes. In the simplest case there are only two different classes. One possible formalization of this task is to estimate a function $f : \mathbb{R}^N \to \{-1,+1\}$ from a given function class F, using input-output training data pairs generated i.i.d. according to an unknown probability distribution $P(\mathbf{x},y)$

$$(\mathbf{x}_1,y_1),\dots,(\mathbf{x}_K,y_K) \in \mathbb{R}^N \times \{-1,+1\}$$

such that f will correctly classify unseen examples $(\mathbf{x},y)$. An example is assigned to the class $+1$ if $f(\mathbf{x}) \geq 0$ and to the class -1 otherwise. The test examples are assumed to be generated from the same probability distribution $P(\mathbf{x},y)$

The studies were partly supported by the *Bundesministerium für Bildung und Forschung* (BMBF), FKZ 01IBB02A and FKZ 01IBB02B, by the *Deutsche Forschungsgemeinschaft* (DFG), FOR 375/B1 and the PASCAL Network of Excellence, EU # 506778. This review is based on [1, 2].

as the training data. The best function f that one can obtain is the one minimizing the expected error (risk)

$$R[f] = \int l(f(\mathbf{x}),y)dP(\mathbf{x},y), \quad (1)$$

where l denotes a suitably chosen loss function, e.g., $l(\hat{y},y) = 0$ for $\hat{y} = y$ and $l(\hat{y},y) = 1$ otherwise (the so-called 0/1-loss). The same framework can be applied for regression problems, where $y \in \mathbb{R}$. Here, the most common loss function is the *squared loss*: $l(f(\mathbf{x}),y) = (f(\mathbf{x}) - y)^2$; see [11] for a discussion of other loss functions.

Unfortunately the risk cannot be minimized directly, since the underlying probability distribution $P(\mathbf{x},y)$ is unknown. Therefore, we have to try to estimate a function that is *close* to the optimal one based on the available information, i.e. the training sample and properties of the function class F the solution f is chosen from. To this end, we need what is called an induction principle. A particular simple one consists in approximating the minimum of the risk (1) by the minimum of the *empirical risk*

$$R_{emp}[f] = \frac{1}{K}\sum_{k=1}^{K} l(f(\mathbf{x}_k),y_k). \quad (2)$$

It is possible to give conditions on the learning machine which ensure that asymptotically (as $K \rightarrow \infty$), the empirical risk will converge towards the expected risk. However, for small sample sizes large deviations are possible and *overfitting* might occur (see Figure 1). Then a small generalization error cannot be obtained by simply minimizing the training error (2). One way to avoid the overfitting dilemma is to *restrict* the complexity of the function class F that one chooses the function f from [12]. The intuition, which will be formalized in the following is that a "simple" (e.g. linear) function that explains most of the data is preferable to a complex one (Occam's razor). Typically one introduces a *regularization* term (e.g. [13]) to limit the complexity of the function class F from which the learning machine can choose. This raises the problem of model selection (e.g. [13]), i.e. how to find the optimal complexity of the function.

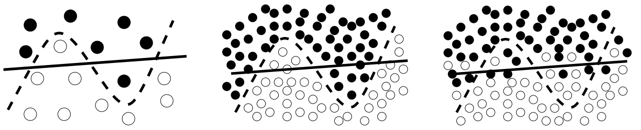

Fig. 1: Illustration of the overfitting dilemma: Given only a small sample (left) either, the solid or the dashed hypothesis might be true, the dashed one being more complex, but also having a smaller training error. Only with a large sample we are able to see which decision reflects the true distribution more closely. If the dashed hypothesis is correct the solid would underfit (middle); if the solid were correct the dashed hypothesis would overfit (right). From [2].

3. LINEAR METHODS FOR CLASSIFICATION AND BEYOND

In BCI research it is very common to use linear classifiers, but although linear classification already uses a very simple model, things *can* still go terribly wrong if the underlying assumptions do not hold, e.g. in the presence of outliers or strong noise which are situations very typically encountered in BCI data analysis. We will discuss these pitfalls and point out ways around them.

Let us first fix the notation and introduce the linear hyperplane classification model upon which we will rely mostly in the following (cf. Fig. 2, see e.g. [14]). In a BCI set-up we measure $k = 1 \ldots K$ samples $\mathbf{x}_k$, where $\mathbf{x}$ are some appropriate feature vectors in n dimensional space. In the training data we have a class label, e.g. $y_k \in \{-1,+1\}$ for each sample point $\mathbf{x}_k$. To obtain a linear hyperplane classifier

$$\mathbf{y} = \operatorname{sign}\left(\mathbf{w}^\top \mathbf{x} + b\right) \quad (3)$$

we need to estimate the normal vector of the hyperplane $\mathbf{w}$ and a threshold b from the training data by some optimization technique [14]. On unseen data $\mathbf{x}$, i.e. in a BCI feedback session we compute the projection of the new data sample onto the direction of the normal $\mathbf{w}$ via Eq.(3), thus determining what class label $\mathbf{y}$ should be given to $\mathbf{x}$ according to our linear model.

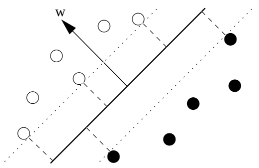

Fig. 2: Linear classifier and margins: A linear classifier is defined by a hyperplane's normal vector $\mathbf{w}$ and an offset b, i.e. the decision boundary is $\{\mathbf{x} \,|\, \mathbf{w}^\top \mathbf{x} + b = 0\}$ (thick line). Each of the two halfspaces defined by this hyperplane corresponds to one class, i.e. $f(\mathbf{x}) = \operatorname{sign}(\mathbf{w}^\top \mathbf{x} + b)$. The *margin* of a linear classifier is the minimal distance of any training point to the hyperplane. In this case it is the distance between the dotted lines and the thick line. From [2].

3.1. Optimal linear classification: large margins versus Fisher's discriminant

Linear methods assume a linear separability of the data. We will see in the following that the optimal separating hyperplane from last section maximizes the *minimal* margin (minmax). In contrast, Fisher's discriminant maximizes the *average* margin, i.e., the margin between the class means.

3.1.1. *Large margin classification*

For linearly separable data there is a vast number of possibilities to determine $(\mathbf{w}, b)$, that all classify correctly on the training set, however that vary in quality on the unseen data (test set). An advantage of the simple hyperplane classifier (in canonical form cf. [12]) is that literature (see e.g. [14, 12]) tells us how to select the *optimal* classifier $\mathbf{w}$ for unseen data: it is the classifier with the largest margin $\rho = 1/\|\mathbf{w}\|_2^2$, i.e. of minimal (euclidean) norm $\|\mathbf{w}\|_2$ [12] (see also Fig. 2). Linear Support Vector Machines (SVMs) realize the large margin by determining the normal vector $\mathbf{w}$ according to

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \ {}^1\!/_2\|\mathbf{w}\|_2^2 + {}^C\!/_K\|\xi\|_1 \quad & \text{subject to} \\ y_k(\mathbf{w}^\top\mathbf{x}_k + b) \geqslant 1 - \xi_k \quad & \text{and} \\ \xi_k \geqslant 0 \quad & \text{for } k = 1, \ldots, K, \end{aligned} \tag{4}$$

where $\|\cdot\|_1$ denotes the ℓ_1-norm: $\|\xi\|_1 = \sum|\xi_k|$. Here the elements of vector ξ are slack variables and parameter C controls the size of the margin vs. the complexity of the separation. While the user has not to care about the slack variables, it is essential to select an approppriate value for the free parameter C for each specific data set. The process of choosing C is called model selection, see e.g. [2]. An issue that is of importance here is discussed in Section 5. One particular strength of SVMs is that they can be turned in nonlinear classifiers in an elegant and effective way, see Section 3.3.

3.1.2. *Fisher's discriminant*

Fisher's discriminant computes the projection $\mathbf{w}$ differently. Under the restrictive assumption that the class distributions are (identically distributed) Gaussians of equal covariance, it can be shown to be Bayes optimal. The separability of the data is measured by two quantities: How far are the projected class means apart (should be large) and how big is the variance of the data in this direction (should be small). This can be achieved by maximizing the so-called Rayleigh coefficient of between and within class variance with respect to $\mathbf{w}$ [15, 16]. The slightly stronger assumptions have been fulfilled in several of our BCI experiments e.g. in [17, 18]. When the optimization to obtain (regularized) Fisher's discriminant is formulated as a mathematical programm, cf. [19, 2], it resembles the SVM:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \ {}^1\!/_2\|\mathbf{w}\|_2^2 + {}^C\!/_K\|\xi\|_2^2 \quad & \text{subject to} \\ y_k(\mathbf{w}^\top\mathbf{x}_k + b) = 1 - \xi_k \quad & \text{for } k = 1, \ldots, K. \end{aligned}$$

3.2. Some remarks about regularization and non-robust classifiers

Linear classifiers are generally more robust than their nonlinear counterparts, since they have only limited flexibility (less free parameters to tune) and are thus less prone to overfitting. Note however that in the presence of strong noise and outliers *even* linear systems can fail. In the cartoon of Fig.3 one can clearly observe that one outlier or strong noise event can change the decision surface drastically, if the influence of single data points on learning is not limited. Although this effect can yield strongly decreased classification results for linear learning machines, it can be even more devastating for nonlinear methods. A more formal way to control one's mistrust in the available training data, is to use regularization (e.g. [13, 20]). Regularization helps to limit (a) the influence of outliers or strong noise (e.g. to avoid Fig.3 middle), (b) the complexity of the classifier (e.g. to avoid Fig.3 right) and (c) the raggedness of the decision surface (e.g. to avoid Fig.3 right). No matter whether linear or nonlinear methods are used, one should *always* regularize, – in particular for BCI data!

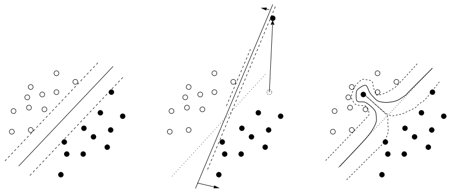

Fig. 3: The problem of finding a maximum margin "hyper-plane" on reliable data (left), data with an outlier (middle) and with a mislabeled pattern (right). The solid line shows the resulting decision line, whereas the dashed line marks the margin area. In the middle and on the right the original decision line is plotted with dots. Illustrated is the noise sensitivity: only one strong noise/outlier pattern can spoil the whole estimation of the decision line. From [21].

3.3. Beyond linear classifiers

Kernel based learning has taken the step from linear to nonlinear classification in a particularly interesting and efficient[1] manner: a linear algorithm is applied in some appropriate (kernel) feature space. Thus, all beneficial properties (e.g. optimality) of linear classification are maintained[2], but at the same time the overall classification is nonlinear in input space, since feature- and input space are nonlinearly related.

Algorithms in feature spaces make use of the following

[1] By virtue of the so-called 'kernel trick' [12].

[2] As we do linear classification in this feature space.

idea: via a nonlinear mapping

$$\begin{aligned}\Phi : \mathbb{R}^N &\rightarrow \mathscr{F}\\ \mathbf{x} &\mapsto \Phi(\mathbf{x})\end{aligned}$$

the data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^N$ is mapped into a potentially much higher dimensional feature space $\mathscr{F}$. For a given learning problem one now considers the same algorithm in $\mathscr{F}$ instead of $\mathbb{R}^N$, i.e. one works with the samples

$$(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_K), y_K) \in \mathscr{F} \times Y.$$

Given this mapped representation a *simple* classification or regression in $\mathscr{F}$ is to be found. This is also implicitly done for (one hidden layer) neural networks, radial basis networks (e.g. [22, 23, 24, 20]) or Boosting algorithms [25] where the input data is mapped to some representation given by the hidden layer, the RBF bumps or the hypotheses space respectively.

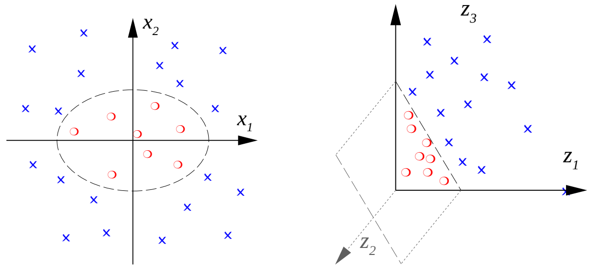

Fig. 4: Two dimensional classification example. Using the second order monomials x_1^2, $\sqrt{2}x_1x_2$ and x_2^2 as features a separation in feature space can be found using a *linear* hyperplane (right). In input space this construction corresponds to a *non-linear* ellipsoidal decision boundary (left). From [2].

The so-called *curse of dimensionality* from statistics says essentially that the difficulty of an estimation problem increases drastically with the dimension N of the space, since – in principle – as a function of N one needs exponentially many patterns to sample the space properly. This well known statement induces some doubts about whether it is a good idea to go to a high dimensional feature space for learning.

However, statistical learning theory tells us that the contrary can be true: learning in $\mathscr{F}$ can be simpler if one uses a low complexity, i.e. *simple* class of decision rules (e.g. linear classifiers). All the variability and richness that one needs to have a powerful function class is then introduced by the mapping Φ. In short: not the dimensionality but the complexity of the function class matters [12]. Intuitively, this idea can be understood from the toy example in Figure 4: in two dimensions a rather complicated *nonlinear* decision surface is necessary to separate the classes, whereas in a feature space of second order monomials (see e.g. [26])

$$\begin{aligned}\Phi : \mathbb{R}^2 &\rightarrow \mathbb{R}^3\\ (x_1, x_2) &\mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2) \qquad (5)\end{aligned}$$

all one needs for separation is a *linear* hyperplane. In this simple toy example, we can easily control both: the statistical complexity (by using a simple linear hyperplane classifier) and the algorithmic complexity of the learning machine, as the feature space is only three dimensional. However, it becomes rather tricky to control the latter for large real world problems. For instance, consider 256 dimensional feature vectors from a BCI experiment as patterns and 5th order monomials as mapping Φ – then one would map to a space that contains all 5$^{\text{th}}$ order products of 256 features, i.e. to a $\binom{5+256-1}{5} \approx 10^{10}$-dimensional space. So, even if one could control the statistical complexity of this function class, one would still run into intractability problems while executing an algorithm in this space.

Fortunately, for certain feature spaces $\mathscr{F}$ and corresponding mappings Φ there is a highly effective trick for computing scalar products in feature spaces using *kernel functions* [27, 28, 29, 12]. Let us come back to the example from Eq. (5). Here, the computation of a scalar product between two feature space vectors, can be readily reformulated in terms of a kernel function k

$$\begin{aligned}\Phi(\mathbf{x})^\top \Phi(\mathbf{y}) &= (x_1^2, \sqrt{2}\, x_1x_2, x_2^2)(y_1^2, \sqrt{2}\, y_1y_2, y_2^2)^\top\\ &= ((x_1, x_2)(y_1, y_2)^\top)^2\\ &= (\mathbf{x}^\top \mathbf{y})^2\\ &=: \mathrm{k}(\mathbf{x}, \mathbf{y}).\end{aligned}$$

This finding generalizes as for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$, and $d \in \mathbb{N}$ the kernel function

$$\mathrm{k}(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\top \mathbf{y})^d$$

computes a scalar product in the space of all products of d vector entries (monomials) of $\mathbf{x}$ and $\mathbf{y}$ [12, 30]. Note furthermore that using a particular SV kernel corresponds to an *implicit* choice of a regularization operator (cf. [33, 34]). Table 1 lists some of the most widely used kernel functions. More sophisticated kernels (e.g. kernels generating splines or Fourier expansions) can be found in [35, 36, 33, 37, 38, 39]. The interesting point about kernel functions is that the scalar product can be *implicitly* computed in $\mathscr{F}$, *without* explicitly using or even knowing the mapping Φ. So, kernels allow to compute scalar products in spaces, where one could otherwise hardly perform any computations. A direct consequence from this finding is [30]: every (linear) algorithm that only uses scalar products can implicitly be executed in $\mathscr{F}$ by using kernels, i.e. one can very elegantly construct a nonlinear version of a linear algorithm.[3] Examples of such

[3] Even algorithms that operate on similarity measures k generating pos-

Table 1: Common kernel functions: Gaussian RBF ($c \in \mathbb{R}$), polynomial ($d \in \mathbb{N}, \theta \in \mathbb{R}$), sigmoidal ($\kappa, \theta \in \mathbb{R}$) and inverse multiquadric ($c \in \mathbb{R}_+$) kernel functions are among the most common ones. While RBF and polynomial are known to fulfill Mercers condition, this is not strictly the case for sigmoidal kernels [31]. Further valid kernels proposed in the context of regularization networks are e.g. multiquadric or spline kernels [13, 32, 33].

Gaussian RBF	$k(\mathbf{x},\mathbf{y}) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{c}\right)$
Polynomial	$(\mathbf{x}^\top \mathbf{y} + \theta)^d$
Sigmoidal	$\tanh(\kappa(\mathbf{x}^\top \mathbf{y}) + \theta)$
inv. multiquadric	$\frac{1}{\sqrt{\|\mathbf{x}-\mathbf{y}\|^2 + c^2}}$

kernel-based learning machines are among others, e.g. Support Vector Machines (SVMs) [12, 2], Kernel Fisher Discriminant (KFD) [40] or Kernel Principal Component Analysis (KPCA) [30].

3.4. Discussion

To summarize: a small error on unseen data cannot be obtained by simply minimizing the training error, on the contrary, this will in general lead to overfitting and non-robust behaviour, even for linear methods (cf. Fig.3). One way to avoid the overfitting dilemma is to *restrict* the complexity of the function class, i.e. a "simple" (e.g. linear) function that explains most of the data is preferable over a complex one that explains all data (Occam's razor). This still leaves the outlier problem which can only be alleviated by an outlier removal step and regularization. Note that whenever a certain linear classifier does not work well, then there are (at least) two potential reasons for this: (a) either the regularization was not done well or non-robust estimators were used and a *properly chosen* linear classifier would have done well. Alternatively it *could* as well be that (b) the problem is intrinsically nonlinear. Then the recommandation is to try a linear classifier in the appropriate kernel-feature space (e.g. Support Vector Machines) and regularize well.

Finally, note that if ideal model selection can be done then the complexity of the learning algorithm is less important. In other words, the model selection process can choose the best method, be it linear or nonlinear. In practice, k-fold cross validation is quite a useful (although not optimal) approximation to such an ideal model selection strategy.

4. FEATURE SELECTION TECHNIQUES

After discussing general ideas on classification, regularization and model selection it is important to stress that the ultimate success of a learning machine relies typically on a proper preprocessing of the data. Here prior knowledge, e.g. about the frequency content of the signal of interest, are taken into account. Furthermore and very important in practice, we can discard non-informative dimensions of the data and thus select the *features of interest for classification* (see e.g. [41]). Straightforward as this may appear, it is in fact a machine learning art of its own, since we must decide on features that don't overfit the training samples but rather generalize to yet unknown test data, even in the presence of noise. It is furthermore a challenge to make feature selection and classification an interleaved and integrated algorithmic process. In this light it becomes clear that, e.g., PCA-based feature selection is in most cases a bad choice, since it only takes the total density of all samples into account, where it should actually consider the class labels, in order not to discard valuable information for the classifier.
In the following we will briefly review some popular feature selection techniques that have also been in use in the BBCI system. Note however that we will not be exhaustive in our exposition, for further references on feature selection in general see e.g. [41, 14] or in the context of BCI see [17, 42].

Suppose for each epoch of recorded brain signals one has a multi-dimensional vector $\mathbf{x}$. Each dimension of that vector is called a *feature*, and the whole vector is called *feature vector*. The samples are coming from a controlled measurement such that the underlying mental state is known, and each sample, resp. feature vector, $\mathbf{x}_k$ ($k = 1, \ldots, K$), has a label y_k. The features may be original, raw features, i.e., potential values at specific times at specific channels as measured by the EEG device, or they may be the result of some preprocessing transformations, like spectral power values in specific frequency bands. The problem of feature selection is the task to find a small subset of all features that suffices to represent the information of the whole feature vector. The objectives of such an enterprise can be manyfold. (1) When feature vectors are very high-dimensional with respect to the number of training examples available, the selection of a suitable smaller subset of features can make the training of the classifier more robust. Note that in some representations the relevant information is spread across all feature dimensions such that a useful selection of features is not possible. In such a case one has to transform the data in a clever way to concentrate the discriminative information. The choice of such a tranformation is often a crucial step in single-trial EEG analysis. The transformation may be based on neurophysiological knowledge about the involved brain functions (frequency filtering, spatial filter-

itive matrices $\mathrm{k}(\mathbf{x}_i, \mathbf{x}_j)_{ij}$ can be interpreted as linear algorithms in some feature space $\mathscr{F}$ [35].

ing, ...), or one may recruit projection techniques from the theory of supervised learning, e.g., a common spatial pattern analysis ([16, 43, 44]), or from unsupervised learning, e.g, independent component analysis ([45, 46, 47]). (2) A neurophysiological assessment of the selected features may lead to a better understanding of the involved brain functions, and—as consequence—to further improvements of the algorithms. (3) One might be interested to reduce the number of features that have to be measured. In BCI research a typical goal is to reduce the number of channels needed to operate the BCI. Note that when more than one feature is derived from each channel, feature selection does not automatically imply a useful channel selection, as the selected features may be spread across many channels. Nevertheless a selection of channels can be gained from feature selection in a straight forward way: define the score of a channel as the norm of the vector of scores of the features belonging to that channel.

Most of the feature selection methods discussed below determine a score (i.e., a real number $\geqslant 0$) for each feature. The selection of features based on this score can be obtained by different strategies. When the objective is to choose the K most informative features, one would choose the features with the top K scores. Such criterium does not take into account what loss one has to tolerate when discarding some features. Another strategy could be to select features in decreasing order of their score until their common score amounts to a specified percentage of the total score of all features.

4.1. Data Used to Illustrate the Feature Selection Methods

To illustrate the feature selection methods we take data from one BCI experiment that will be explained in more detail in Section 6.2. Here it suffices to know that the subject performed mental imagery of left hand and foot movements for 140 epochs of 3.5 seconds length for each class. Brain activity was recorded by 118 EEG channels and spectral power values were calculated at a 2 Hz resolution in the frequency range from 6 to 32 Hz using a Kaiser window. Thus the feature vectors have $13 \cdot 118 = 1534$ dimensions.

4.2. Methods from Classical Statistics or Pattern Recognition

Basic methods from statistics quantify "how significant" the difference between the means of two distributions of independent observations is. Such measures can be applied to each feature separately to get a score of how informative each feature is with respect to discriminating the two distributions.

The methods in this section determine a score for each feature by looking at that feature alone (and its labels). Let $(x_1,y_1),\dots,(x_K,y_K)$ be a sequence of one-dimensional observations (i.e., a single feature) with labels $y_k \in \{+1,-1\}$, define $X^+ := \langle x_k \,|\, y_k = +1\rangle$ and $X^- := \langle x_k \,|\, y_k = -1\rangle$ and let N^+, resp. N^- be the number of samples in the positive, resp. negative class.

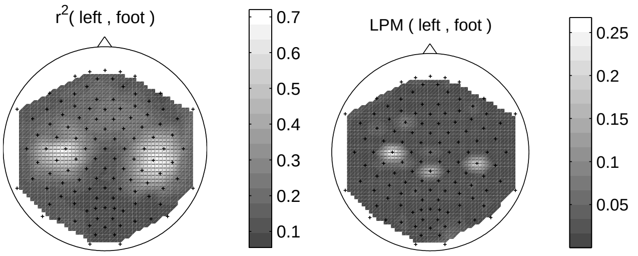

Fig. 5: The left scalp plot depicts the channel selection scores calculated as r^2-coefficients. The scores obtained by t-scaling, r-coefficients and the Fisher criterion look similar. The scalp plot on the right shows the weights as obtained by the sparse LPM classifier, see Section 4.3.2.

4.2.1. *Student's t-Statistics*

As well known from the Student's t-test, one can scale the difference between the estimated means of two one-dimensional distributions, such that it obeys the t-statistics. Defining the estimation of the standard deviation of the difference between the means of X^+ and X^- as $sxd :=$

$$\sqrt{\left(\frac{1}{N^+}+\frac{1}{N^-}\right)\frac{(N^+-1)\,\mathrm{var}(X^+)+(N^--1)\,\mathrm{var}(X^-)}{N^++N^--2}}$$

the t-scaled difference is $X_\mathrm{t} := \frac{\mathrm{mean}(X^+)-\mathrm{mean}(X^-)}{sxd}$. Large absolute values of X_t indicate that the difference between the means of the two distributions is significant. (For the t-test a threshold is calculated, which depends on the desired level of significance α and the degree of freedom $N^+ + N^- - 2$.) The absolute value of X_t serves as score: $\mathrm{score}_\mathrm{t}(X) = \mathrm{abs}(X_t)$. Based on this score features can be selected as described above.

4.2.2. *Bi-serial Correlation Coefficients*

Another statistical way to measure how much information one feature carries about the labels is the bi-serial correlation coefficient r:

$$X_r := \frac{\sqrt{N^+\cdot N^-}}{N^++N^-}\,\frac{\mathrm{mean}(X^-)-\mathrm{mean}(X^+)}{\mathrm{std}(X^+\cup X^-)}$$

or the r^2-coefficient $X_{r^2} := X_r^2$, which reflects how much of the variance in the distribution of all samples is explained by the class affiliation. Figure 5 shows a scalp plot of the channel scores that have been derived from the r^2 feature scores as explained above.

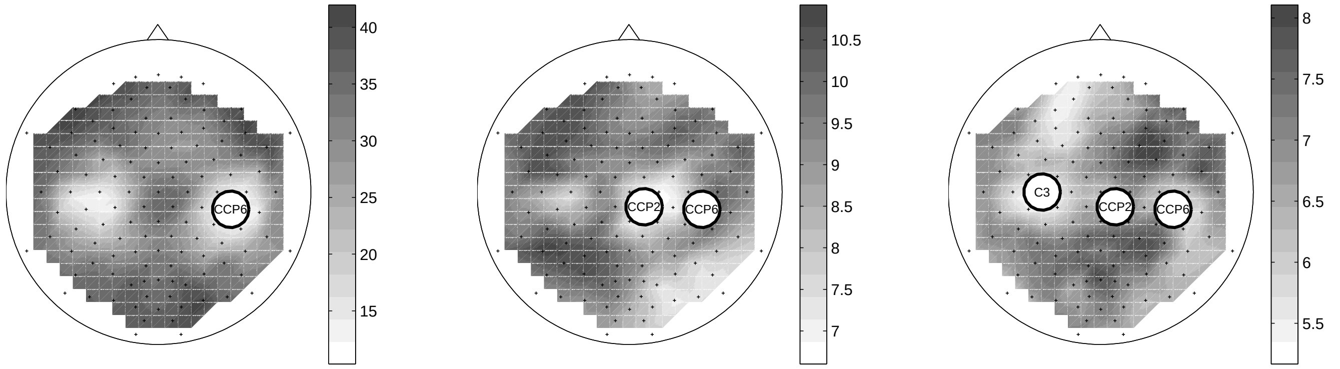

Fig. 6: Series of scalp plots from the incremental channel selection procedure. The first scalp shows the error rates of single channel classification. The channel with minimum validation error, CCP6, is selected. The next scalp shows the error rates of two channel classifications where CCP6 is combined with all other channels. The pair CCP6, CCP2 gives the best result here. The last plot shows that for three channel classification C3 is selected to join the previously chosen channels. Note that the selected channels lie over those cortices expected to contribute to the discrimination of left hand vs. foot imagery. (In C3 ipsi-lateral activation/deactivation differences for left hand motor imagery can be observed.)

4.2.3. *Fisher Criterion*

The Fisher Criterion scores the (squared) distance between the class means in relation to the intra-class variances:

$$\text{score}_{\text{F}}(X) = \frac{(\text{mean}(X^-) - \text{mean}(X^+))^2}{\text{var}(X^-) + \text{var}(X^+)}$$

4.3. Methods based on Machine Learning Techniques

In this section we present two methods for feature selection based on classifiers. One important issue has to be noted when considering classifier based feature scores. In contrast to the methods discussed above, a classifier based method might also score some features very high that only or mainly consists of noise. A classifier might use such a feature to cancel the noise in another channel that contains a mixture of signal and noise. The selected noise channel helps to find the discriminative information although it actually holds no such information by itself. Whether this property is wanted or not, depends on what the feature selection is made for.

4.3.1. *Incremental Selection Based on Cross-Validation*

This straightforward method comes from the theory of machine learning. It is an iterative procedure which can be performed using any classifier. Starting from the empty set, one feature is added in each step based on cross-validation results. Note that this method can directly be used for channel selection without going the detour of averaging feature scores.

Let $\mathscr{C}$ be the set of all channels. Start with the empty set of selected features $\mathscr{S}_0 := \emptyset$. In step j determine $\text{err}(c)$ for each channel $c \in \mathscr{C} - \mathscr{S}_{j-1}$ as the cross-validation error obtained for the feature vectors with those dimensions corresponding to channels $\mathscr{S}_{j-1} \cup \{c\}$. Let c_{best} be the channel with minimum cross-validation error and define $e(j) := \text{err}(c_{\text{best}}) = \min\{\text{err}(c) \,|\, c \in \mathscr{C} - \mathscr{S}_{j-1}\}$ and $\mathscr{S}_j := \mathscr{S}_{j-1} \cup \{c_{\text{best}}\}$.

The choice of the stopping criterion depends on what is intended by the feature selection. If the objective is to obtain a set of K features, one simply stops after step K. If the objective is to find a set of features that gives the best classification, one stops when the sequence $\langle e(j) \,|\, j = 1, \ldots, |\mathscr{C}| \rangle$ starts to increase (or stops to 'significantly' decrease). Figure 6 shows three steps of the incremental feature selection.

4.3.2. *Weighting of Sparse Classifiers*

A special category of classifiers gives rise to a method of feature selection that is considerably faster compared to the incremental procedure above. A linear classifier is called *sparse*, if it favors weighting vectors with a high percentage of elements being negligible. The formulation "high percentage" is not very precise, but typically such classifiers have a parameter, which allows to control the trade-off between sparsity and percentage of misclassification (on the training set). There are two variants of sparse classifiers which are very closely related to linear SVMs, cf. Section 3.1.1. In the classifiers presented here, sparse solutions are obtained by using a linear instead of a quadratic norm, as explained below. Such a strategy leads to feasible optimization problems, but sparsity is not guaranteed. Nevertheless, in practical problems which have sparse solutions, they are typically found.

Using the same notation as above, the *Linear Programming Machine* (LPM) is obtained from the linear SVM by replacing the ℓ_2-norm on the regularizer by the ℓ_1-norm

($\|\mathbf{w}\|_1 = \Sigma|\mathbf{w}_n|$), i.e., the weight vector $\mathbf{w}$ (normal vector of the separating hyperplane), bias b and slack variables ξ are determined by the minimization

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & {}^1\!/_N \, \|\mathbf{w}\|_1 + {}^C\!/_K \, \|\xi\|_1 && \text{subject to} \\ & y_k(\mathbf{w}^\top \mathbf{x}_k + b) \geqslant 1 - \xi_k && \text{and} \\ & \xi_k \geqslant 0 && \text{for } k = 1, \dots, K, \end{aligned}$$

Here the parameter C controls the trade-off between sparsity ($\|\mathbf{w}\|_1$ is small) and margin errors ($\|\xi\|_1$). The classifier is called Linear Programming Machine because the minimization is a constrained linear optimization that can be solved by linear programming technique.

A different approach can be obtained by formulating the regularized Fisher Discriminant as a mathematical programm, cf. Section 3.1.2. This formalization gives the opportunity to consider some interesting variants. Again replacing the ℓ_2-norm on the regularizer by the ℓ_1-norm gives a sparse variant of the Fisher Discriminant:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & {}^1\!/_N \, \|\mathbf{w}\|_1 \; + \; {}^C\!/_K \, \|\xi\|_2^2 && \text{subject to} \\ & y_k(\mathbf{w}^\top \mathbf{x}_k + b) = 1 - \xi_k && \text{for } k = 1, \dots, K \end{aligned}$$

This optimization is a constrained quadratic convex problem, that can be solved, e.g., by the *cplex* optimizer [48]. To have a computationally less demanding classifier, the ℓ_2-norm on the slack variables ξ can also be replaced by the ℓ_1-norm (Linear Sparse Fisher Discriminant, LSFD). In that case the minimum can be found by a constrained linear program. Note that in spite of the little formal difference between this classifier and the LPM ('$=$' vs. '$\geqslant$' and $\xi_k \geqslant 0$ in the constraints) the objective is quite different. While the LPM is—like SVMs—a large margin classifier, the LSFD maximizes the distance between the class means relative to the intra class variance, like the usual Fisher Discriminant.

Having trained one of the linear sparse classifiers results in a sparse weight vector $\mathbf{w}$ that projects feature vectors perpendicular to the separating hyperplane. The absolute value of the weight for each feature can serve as a score for feature selection. Figure 7 depicts the weight vector determined by an LPM as a gray scale coded matrix. The derived channel scores are indicated by the right bar and, arranged as scalp topography, in the right plot of Figure 5.

Note that the classifiers presented in this section can be kernelized. But in the kernelized nonlinear classifiers, sparseness is obtained in feature space and thus cannot be used for the selection of (input) features.

5. CAVEATS IN THE VALIDATION

The objective when evaluating offline classifications is to estimate the future performance of the investigated methods, or in other words the generalization ability. The most objective report of BCI performance are the results of actual feedback sessions. But in the development and enhacement of BCI systems it is essential to make offline investigations. Making BCI feedback experiments is costly and time-consuming. So, when one is exploring new ways for processing or classifying brain signals, one would first like to validate and tune the new methods before integrating them into an online system and pursuing feedback experiments. But there are many ways which lead to an (unintentional) overestimation of the generalization ability. In this section we discuss what has to be noted when analyzing the methods presented in this paper. A much more thorough discussion of the evaluation methods for BCI classifications will be the subject of a forthcoming paper.

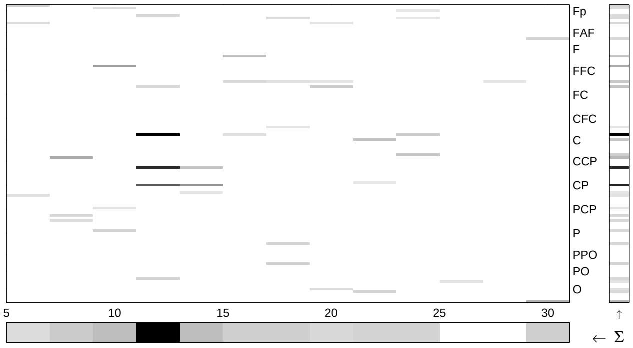

Fig. 7: This figure shows the weight vector of a sparse classifier (absolute values). The bar on the bottom shows the sum across all channels. The focus in the frequency range lies on the α-band (here 11–14 Hz). Note that less than 4% of the features were assigned non-zero weights. The right plot in Figure 5 shows the channel score derived from these feature scores.

5.1. The Fundamental Problem

The essence in estimating the generalization error is to split the available labelled data into training and test set, to determine all free hyperparameters and parameters on the training set and then to evaluate the method on the test data. The test data must not have been used in any way before all parameters have been calculated, all hyperparameters have been selected and all other selections have been made, to ensure that the estimation of the error is unbiased. In a cross-validation or a leave-one-out validation the data set is split in many different ways into training and test set, the procedure as outlined above is performed for each split, and finally the mean of all errors obtained for the test data is taken as estimate for the generalization error. A common error in the evaluation of machine learning techniques is that some preprocessing steps or some parameter selections are performed on the whole data set before the cross-validation. If the preprocessing acts *locally* on each sample, there is no problem, but if the preprocessing of one sample depends somehow on the distribution of all samples, the basic principle that the test set must remain unseen until all free parameters have been fixed, is violated. Whether this violation leads to a severe underestimation of the generalization error cannot be said in general as it depends on many factors, but certainly it cannot be excluded.

When enough data samples are available, the problem can be solved by having a three-fold split of the data into training, test and validation set. In this setting methods with competing parameter settings would all be trained on the training and applied to the validation set. The setting with the best performance on the validation set is chosen and applied to the test set. In a cross-validation one has many of such three-fold splits and the mean error on the test set is taken as estimate of the generalization error.

While this procedure is conceptually sound, it is often not a viable way in BCI context where the number of available labelled samples is very limited compared to the complexity of the data. In such a setting doing the model selection on one fixed split is not robust. One can circumvent this problem, when sufficiently computing resources (computing power or time) are available by doing a *nested* cross-validation. While the outer cross-validation is used to get the estimation of the generalization error, there is an inner cross-validation performed on each training set of the outer validation to do the model selection.

5.2. Evaluating Classifiers with Hyperparameters

Machine learning classifiers have parameters whose values are adapted to given labelled data (training data) by some optimization criterion, like $\mathbf{w}, b, \xi$ in (4). Some classifiers also have some so called hyperparameters, like C in (4). These are parameters which also have to be adapted to the data, but for which no direct optimization criterion exists. Typically hyperparameters control the capacity of the classifier or the raggedness of the separation surface. In the classifiers presented in Section 4.3.2 the hyperparameter C controls the sparsity of the classifier (sparser classifiers have less capacity). To validate the generalization ability of a classifier with hyperparameters one has to perform a nested cross-validation as explained above. On each training set of the outer cross-validation, an inner cross-validation is performed for different values of the hyperparameters. The one with minimum (inner) cross-validation error is selected and evaluated on the test set of the outer cross-validation.

5.3. Evaluating Preprocessing Methods

The fundamental problem that was discussed in Section 5.1 appears when a preprocessing method (as CSP) is applied to the whole data set before the cross-validation. But even a preprocessing which is not label dependent can be problematic when it operates non-locally. To make an unbiased validation non-local processings have to be performed *within* the cross-validation, whereby all parameters have to be estimated from the training data. For example, a correct evaluation of a method that uses ICA as preprocessing has to calculate the projection matrix *with* the cross-validation on each training set. Data of the test set are projected using that matrix. While the bias introduced by applying ICA before the cross-validation can be expected to be marginal, it is critical for the label dependent method CSP.

5.4. Evaluating Feature Selection Methods

It is very tempting to evaluate feature selection methods by running the feature selection on the whole data set and then doing a cross-validation on the data set of reduced features. Unfortunately such a procedure is found in many publications, but it is conceptually wrong and may very well lead to an underestimation of the generalization error. As argumented in Section 5.3 a preprocessing like feature selection has to be performed within the cross-validation. When the method has hyperparameters (like the number of features to extract) the selection of these hyperparameters has to be done by an inner cross-validation, see Section 5.2.

6. THE BERLIN BRAIN-COMPUTER INTERFACE

The Berlin Brain-Computer Interface is driven by the idea to shift the main burden of the learning task from the human subject to the computer under the motto 'let the machines learn'. To this end, the machine learning and feature selection methods presented in the previous sections are applied to EEG data from selected BBCI paradigms: selfpaced [17, 18] and imagined [49, 44, 50] experiments.

6.1. Self-paced Finger Tapping Experiments

In preparation of motor tasks, a negative readiness potential precedes the actual execution. Using multi-channel EEG recordings it has been demonstrated that several brain areas contribute to this negative shift (cf. [51, 52]). In unilateral finger or hand movements the negative shift is mainly focussed on the frontal lobe in the area of the corresponding motor cortex, i.e., contralateral to the performing hand. Based on the laterality of the pre-movement potentials it is possible to discriminate multi-channel EEG recordings of upcoming left from right hand movements. Fig. 8 shows the lateralized readiness potential during a 'self-paced' experiment, as it can be revealed here by averaging over 260 trials in one subject.

In the 'self-paced' experiments, subjects were sitting in a normal chair with fingers resting in the typing position at the computer keyboard. In a deliberate order and on their

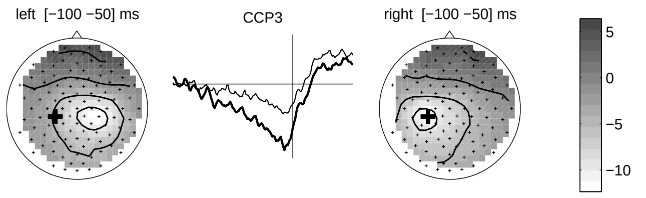

Fig. 8: The scalp plots show the topography of the electrical potentials prior to keypress with the left resp. right index finger. The plot in the middle depicts the event-related potential (ERP) for left (thin line) vs. right (thick line) index finger in the time interval -1000 to -500 ms relative to keypress at electrode position CCP3, which is marked by a bigger cross in the scalp plots. The contralateral negativation (lateralized readiness potential, LRP) is clearly observable. Approx. 260 trials per class have been averaged.

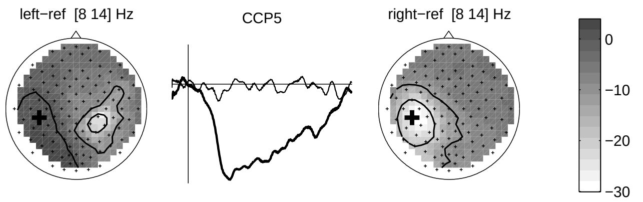

Fig. 9: This scalp plots show the topography of the band power in the frequency band 8–14 Hz relative to a reference period. The plot in the middle shows ERD curves (temporal evolution of band power) at channel CCP5 (mark by a bigger cross in the scalp plots) for left (thin line) and right (thick line) hand motor imagery. The contralateral attenuation of the μ-rhythm during motor imagery is clearly observable. For details on ERD, see [54].

own free will (but instructed to keep a pace of approximately 2 seconds), they were pressing keys with their index and little fingers.

EEG data was recorded with 27 up to 120 electrodes, arranged in the positions of the extended 10-20 system, referenced to nasion and sampled at 1000 Hz. The data were downsampled to 100 Hz for further offline analyses. Surface EMG at both forearms was recorded to determine EMG onset. In addition, horizontal and vertical electrooculograms (EOG) were recorded to check for correlated eye movements.

In [7], it has been demonstrated that when analyzing LRP data offline with the methods detailed in the previous sections, classification accuracies of more than 90% can be reached at 110 ms before the keypress, i.e. a point in time where classification on EMG is still at chance level. These findings suggest that it is possible to use a BCI in time critical applications for an early classification and a rapid response.

Table 2 shows the classification results for one subject when comparing different machine learning methods. Clearly regularization and careful model selection are mandatory which can, e.g., be seen by comparing LDA and RLDA. Of course, regularization is of more importance the higher the dimensionality of features is. The reason of the very bad performance of *k*-NN is that the underlying Euclidean metric is not appropriate for the bad signal-to-noise ratio found in EEG trials. For further details refer to [17, 18]. Note that the accuracy of 90% can be maintained in recent realtime feedback experiments [53]. Here, as no trigger information is not available beforehand, the classification decision is split into one classifier that decides whether a movement is being prepared and a second classifier that decides between left and right movement to come. In fact subjects tell about peculiar experiences in their decision making introspection since the feedback is immediate.

Table 2: Test set error (± std) for classification at 110 ms before keystroke; ›mc‹ refers to the 56 channels over (sensori) motor cortex, ›all‹ refers to all 105 channels. The algorithms in question are Linear Discriminant Analysis (LDA), Regularized Linear Discriminant Analysis (RLDA), Linear Programming Machine (LPM), Support Vector Machine with Gaussian RBF Kernel (SVMrbf) and *k*-Nearest Neighbor (*k*-NN).

channels	LDA	RLDA	LPM	SVMrbf	*k*-NN
all	16.9±1.3	8.4±0.6	7.7±0.6	8.6±0.6	28.4±0.9
mc	9.3±0.6	6.3±0.5	7.4±0.7	6.7±0.7	22.0±0.9

6.2. Motor Imagery Experiments

During imagination of a movement, a lateralized attenuation of the μ- and/or central β-rhythm can be observed localized in the corresponding motor and somatosensory cortex. Besides a usual spectral analysis, this effect can be visualized by plotting event-related desynchronization (ERD) curves [54] which show the temporal evolution of the band-power in a specified frequency band. A typical averaged ERD is shown in Fig. 9.

We performed experiments with 6 healthy subjects performing motor imagery. The subjects were sitting comfortably in a chair with their arms in a relaxed position on an arm rest. Two different sessions of data collection were provided: In both a target “L”, “R” and “F” (for left, right hand and foot movement) is presented for the duration of 3.5 seconds to the subject on a computer screen. In the first session type this is done by visualizing the letter on the middle of the screen. In the second session type the left, right or lower triangle of a moving gray rhomb is colored red. For the whole length of this period, the subjects were instructed to imagine a sensorimotor sensation/movement in left hand, right hand resp. one foot. After stimulus presentation, the screen was blank for 1.5 to 2 seconds. In this manner, 35 trials per class per session were recorded. After 25 trials, there was a short break for relaxation. Four sessions (two of each training type) were performed. EEG data was recorded with 128 electrodes together with EMG from both arms and the involved foot, and EOG as described above.

An offline machine learning analysis of the “imagined”-experiments yields again high classification rates (up to 98.9% with the feature combination algorithm PROB [44, 49]), which predicts the feasibility of this paradigm for online feedback situations (see also [50]). In fact, our recent online experiments have confirmed this prediction by showing high bitrates for several subjects. These subjects were untrained and had to play video games like ’brain pong’, ’basket’ (a spelling task) and ’controlled 1-D cursor movement’ [55]. Depending on the ’game’ scenario the best subjects could achieve information transfer rates of up to 37 Bits/min.

7. CONCLUSION

After a brief review of general linear and non-linear machine learning techniques, this paper discussed variable selection methods and their application to EEG data. These techniques are a salient ingredient of the BBCI online feedback system. Note that although machine learning algorithms were initially always tested offline, it is the mastery of choosing the ’right’ complexity for a learning problem that makes the resulting classifiers generalize and thus renders them useful in real BCI feedback experiments. In particular the paradigm shift away from subject training to individualization and adaptation (’let the machines that learn’) of the signal processing and classification algorithm to the specific brain ’under study’ holds the key to the success of the BBCI. Being able to use (B)BCI for untrained subjects dramatically enhances and broadens the spectrum of practical applications in human-computer interfacing.

Acknowledgments: We thank our co-authors from previous publications for letting us use the figures and joint results [2, 1, 18, 21].

8. REFERENCES

[1] Klaus-Robert Müller, Charles W. Anderson, and Gary E. Birch, “Linear and non-linear methods for brain-computer interfaces,” *IEEE Trans. Neural Sys. Rehab. Eng.*, vol. 11, no. 2, 2003, 165–169.

[2] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, “An introduction to kernel-based learning algorithms,” *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 181–201, 2001.

[3] Jonathan R. Wolpaw, Niels Birbaumer, Dennis J. McFarland, Gert Pfurtscheller, and Theresa M. Vaughan, “Brain-computer interfaces for communication and control,” *Clin. Neurophysiol.*, vol. 113, pp. 767–791, 2002.

[4] J. R. Wolpaw, N. Birbaumer, William J. Heetderks, D. J. McFarland, P. Hunter Peckham, G. Schalk, E. Donchin, Louis A. Quatrano, C. J. Robinson, and T. M. Vaughan, “Brain-computer interface technology: A review of the first international meeting,” *IEEE Trans. Rehab. Eng.*, vol. 8, no. 2, pp. 164–173, 2000.

[5] Andrea Kübler, Boris Kotchoubey, Jochen Kaiser, Jonathan Wolpaw, and Niels Birbaumer, “Brain-computer communication: Unlocking the locked in,” *Psychol. Bull.*, vol. 127, no. 3, pp. 358–375, 2001.

[6] Eleanor A. Curran and Maria J. Stokes, “Learning to control brain activity: A review of the production and control of EEG components for driving brain-computer interface (BCI) systems,” *Brain Cogn.*, vol. 51, pp. 326–336, 2003.

[7] Matthias Krauledat, Guido Dornhege, Benjamin Blankertz, Gabriel Curio, and Klaus-Robert Müller, “The berlin brain-computer interface for rapid response,” in *Proceedings of the 2nd International Brain-Computer Interface and Training Course, Graz 2004*, 2004, accepted.

[8] J. R. Wolpaw, D. J. McFarland, and T. M. Vaughan, “Brain-computer interface research at the Wadsworth Center,” *IEEE Trans. Rehab. Eng.*, vol. 8, no. 2, pp. 222–226, 2000.

[9] N. Birbaumer, N. Ghanayim, T. Hinterberger, I. Iversen, B. Kotchoubey, A. Kübler, J. Perelmouter, E. Taub, and H. Flor, “A spelling device for the paralysed,” *Nature*, vol. 398, pp. 297–298, 1999.

[10] S.S. Haykin, *Adaptive Filter Theory*, Prentice Hall, 1995.

[11] A.J. Smola and B. Schölkopf, “On a kernel–based method for pattern recognition, regression, approximation and operator inversion,” *Algorithmica*, vol. 22, pp. 211–231, 1998.

[12] V.N. Vapnik, *The nature of statistical learning theory*, Springer Verlag, New York, 1995.

[13] T. Poggio and F. Girosi, “Regularization algorithms for learning that are equivalent to multilayer networks,” *Science*, vol. 247, pp. 978–982, 1990.

[14] R.O. Duda, P.E.Hart, and D.G.Stork, *Pattern classification*, John Wiley & Sons, second edition, 2001.

[15] R.A. Fisher, “The use of multiple measurements in taxonomic problems,” *Annals of Eugenics*, vol. 7, pp. 179–188, 1936.

[16] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, San Diego, 2nd edition, 1990.

[17] Benjamin Blankertz, Gabriel Curio, and Klaus-Robert Müller, “Classifying single trial EEG: Towards brain computer interfacing,” in *Advances in Neural Inf. Proc. Systems (NIPS 01)*, T. G. Diettrich, S. Becker, and Z. Ghahramani, Eds., 2002, vol. 14, pp. 157–164.

[18] Benjamin Blankertz, Guido Dornhege, Christin Schäfer, Roman Krepki, Jens Kohlmorgen, Klaus-Robert Müller, Volker Kunzmann, Florian Losch, and Gabriel Curio, “Boosting bit rates and error detection for the classification of fast-paced motor commands based on single-trial EEG analysis,” *IEEE Trans. Neural Sys. Rehab. Eng.*, vol. 11, no. 2, pp. 127–131, 2003.

[19] S. Mika, G. Rätsch, and K.-R. Müller, “A mathematical programming approach to the kernel Fisher algorithm,” in *Advances in Neural Information Processing Systems*, T.K. Leen, T.G. Dietterich, and V. Tresp, Eds. 2001, vol. 13, pp. 591–597, MIT Press.

[20] G. Orr and K.-R. Müller, Eds., *Neural Networks: Tricks of the Trade*, vol. 1524, Springer LNCS, 1998.

[21] G. Rätsch, T. Onoda, and K.-R. Müller, “Soft margins for AdaBoost,” *Machine Learning*, vol. 42, no. 3, pp. 287–320, Mar. 2001, also NeuroCOLT Technical Report NC-TR-1998-021.

[22] J. Moody and C. Darken, “Fast learning in networks of locally-tuned processing units,” *Neural Computation*, vol. 1, no. 2, pp. 281–294, 1989.

[23] S. Haykin, *Neural Networks : A Comprehensive Foundation*, Macmillan, New York, 1994.

[24] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[25] 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*, vol. 55, no. 1, pp. 119–139, 1997.

[26] J. Schürmann, *Pattern Classification: a unified view of statistical and neural approaches*, Wiley, New York, 1996.

[27] M. Aizerman, E. Braverman, and L. Rozonoer, "Theoretical foundations of the potential function method in pattern recognition learning.," *Automation and Remote Control*, vol. 25, pp. 821–837, 1964.

[28] S. Saitoh, *Theory of Reproducing Kernels and its Applications*, Longman Scientific & Technical, Harlow, England, 1988.

[29] B.E. Boser, I.M. Guyon, and V.N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, D. Haussler, Ed., 1992, pp. 144–152.

[30] B. Schölkopf, A.J. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[31] A. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, 2001, Forthcoming.

[32] F. Girosi, M. Jones, and T. Poggio, "Priors, stabilizers and basis functions: From regularization to radial, tensor and additive splines," Tech. Rep. A.I. Memo No. 1430, Massachusetts Institute of Technology, June 1993.

[33] A.J. Smola, B. Schölkopf, and K.-R. Müller, "The connection between regularization operators and support vector kernels," *Neural Networks*, vol. 11, pp. 637–649, 1998.

[34] F. Girosi, "An equivalence between sparse approximation and support vector machines," A.I. Memo No. 1606, MIT, 1997.

[35] B. Schölkopf, *Support vector learning*, Oldenbourg Verlag, Munich, 1997.

[36] V.N. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[37] D. Haussler, "Convolution kernels on discrete structures," Tech. Rep. UCSC-CRL-99-10, UC Santa Cruz, July 1999.

[38] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller, "Engineering Support Vector Machine Kernels That Recognize Translation Initiation Sites," *BioInformatics*, vol. 16, no. 9, pp. 799–807, Sept. 2000.

[39] M. Stitson, A. Gammerman, V.N. Vapnik, V. Vovk, C. Watkins, and J. Weston, "Support vector regression with ANOVA decomposition kernels," Tech. Rep. CSD-97-22, Royal Holloway, University of London, 1997.

[40] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, "Fisher discriminant analysis with kernels," in *Neural Networks for Signal Processing IX*, Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, Eds. 1999, pp. 41–48, IEEE.

[41] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, Mar. 2003.

[42] T.N. Lal, M. Schröder, T. Hinterberger, J. Weston, M. Bogdan, N. Birbaumer, and B. Schölkopf, "Support vector channel selection in bci," *IEEE Transactions on Biomedical Engineering, Special Issue on Brain-Machine Interfaces*, vol. 51, no. 6, pp. 1003–1010, 2004.

[43] H. Ramoser, J. Müller-Gerking, and G. Pfurtscheller, "Optimal spatial filtering of single trial EEG during imagined hand movement," *IEEE Trans. Rehab. Eng.*, vol. 8, no. 4, pp. 441–446, 2000.

[44] Guido Dornhege, Benjamin Blankertz, Gabriel Curio, and Klaus-Robert Müller, "Boosting bit rates in non-invasive EEG single-trial classifications by feature combination and multi-class paradigms," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, pp. 993–1002, 2004.

[45] Frank Meinecke, Andreas Ziehe, Motoaki Kawanabe, and Klaus-Robert Müller, "A Resampling Approach to Estimate the Stability of one- or multidimensional Independent Components," *IEEE Transactions on Biomedical Engineering*, vol. 49, no. 12, pp. 1514–1525, 2002.

[46] K.-R. Müller, R. Vigario, F. Meinecke, and A. Ziehe, "Blind source separation techniques for decomposing event-related brain signals," *International Journal of Bifurcation and Chaos*, vol. 14, no. 2, pp. 773–791, 2004.

[47] G. Wübbeler, A. Ziehe, B.-M. Mackert, K.-R. Müller, L. Trahms, and G. Curio, "Independent component analysis of non-invasively recorded cortical magnetic DC-fields in humans," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 5, pp. 594–599, 2000.

[48] "Ilog solver, ilog cplex 6.5 reference manual," `www.ilog.com`, 1999.

[49] Guido Dornhege, Benjamin Blankertz, Gabriel Curio, and Klaus-Robert Müller, "Increase information transfer rates in BCI by CSP extension to multi-class," in *Advances in Neural Inf. Proc. Systems (NIPS 03)*, 2004, vol. 16, in press.

[50] Matthias Krauledat, Guido Dornhege, Benjamin Blankertz, Florian Losch, Gabriel Curio, and Klaus-Robert Müller, "Improving speed and accuracy of brain-computer interfaces using readiness potential features," in *Proceedings of the 26th Annual International Conference IEEE EMBS on Biomedicine, San Francisco*, 2004, accepted.

[51] R. Q. Cui, D. Huter, W. Lang, and L. Deecke, "Neuroimage of voluntary movement: topography of the Bereitschaftspotential, a 64-channel DC current source density study," *Neuroimage*, vol. 9, no. 1, pp. 124–134, 1999.

[52] W. Lang, O. Zilch, C. Koska, G. Lindinger, and L. Deecke, "Negative cortical DC shifts preceding and accompanying simple and complex sequential movements," *Exp. Brain Res.*, vol. 74, no. 1, pp. 99–104, 1989.

[53] Roman Krepki, Benjamin Blankertz, Gabriel Curio, and Klaus-Robert Müller, "The Berlin Brain-Computer Interface (BBCI): towards a new communication channel for online control in gaming applications," *Journal of Multimedia Tools and Applications*, 2004, accepted.

[54] Gert Pfurtscheller and F. H. Lopes da Silva, "Event-related EEG/MEG synchronization and desynchronization: basic principles," *Clin. Neurophysiol.*, vol. 110, no. 11, pp. 1842–1857, Nov 1999.

[55] Roman Krepki, Benjamin Blankertz, Gabriel Curio, and Klaus-Robert Müller, "The Berlin Brain-Computer Interface (BBCI): towards a new communication channel for online control of multimedia applications and computer games," in *9th International Conference on Distributed Multimedia Systems (DMS'03)*, 2003, pp. 237–244.