A review of classification algorithms for EEG-based brain–computer interfaces

Fabien Lotte, Marco Congedo, Anatole Lécuyer, Fabrice Lamarche, Bruno Arnaldi

TOPICAL REVIEW

A Review of Classification Algorithms for EEG-based Brain-Computer Interfaces

F Lotte[1], M Congedo[2], A Lécuyer[1], F Lamarche[1] and B Arnaldi[1]

[1]IRISA / INRIA Rennes, Campus universitaire de Beaulieu, Avenue du Général Leclerc, 35042 RENNES Cedex, France

[2]France Telecom R&D, Tech/ONE Laboratory, 28 Chemin du vieux Chêne,InoVallée, 38240 Grenoble, France

E-mail: fabien.lotte@irisa.fr

Abstract. In this paper we review classification algorithms used to design Brain-Computer Interface (BCI) systems based on ElectroEncephaloGraphy (EEG). We briefly present the commonly employed algorithms and describe their critical properties. Based on the literature, we compare them in terms of performance and provide guidelines to choose the suitable classification algorithm(s) for a specific BCI.

PACS numbers: 8435, 8780

1. Introduction

A Brain-Computer Interface (BCI) is a communication system that does not require any peripheral muscular activity [1]. Indeed, BCI systems enable a subject to send commands to an electronic device only by means of brain activity [2]. Such interfaces can be considered as being the only way of communication for people affected by a number of motor disabilities [3].

In order to control a BCI, the user must produce different brain activity patterns that will be identified by the system and translated into commands. In most existing BCI, this identification relies on a classification algorithm [4], i.e., an algorithm that aims at automatically estimating the class of data as represented by a feature vector [5]. Due to the rapidly growing interest for EEG-based BCI, a considerable number of published results is related to the investigation and evaluation of classification algorithms. To date, very interesting reviews of BCI have been published [1] [6] but none has been specifically dedicated to the review of classification algorithms used for BCI, their properties and their evaluation. This paper aims at filling this lack. Therefore, one of the main objectives of this paper is to survey the different classification algorithms used in EEG-based BCI research and to identify their critical properties. Another objective is to provide guidelines in order to help the reader with choosing the most appropriate

classification algorithm for a given BCI experiment. This amounts to comparing the algorithms and assessing their performances according to the context.

This paper is organized as follows: Section 2 depicts a BCI as a pattern recognition system and emphasizes the role of classification. Section 3 surveys the classification algorithms used for BCI and finally, Section 4 assesses them and identifies their usability depending on the context.

2. Brain-Computer Interfaces seen as a pattern recognition system

The very aim of BCI is to translate brain activity into a command for a computer. To achieve this goal, either regression [7] or classification [8] algorithms can be used. Using classification algorithms is the most popular approach. These algorithms are used to identify "patterns" of brain activity [4]. In this paper, we consider a BCI system as a pattern recognition system [5] [9] and focus on the classification algorithms used to design them. The performance of a pattern recognition depends on both the features and the classification algorithm employed. These two components are highlighted in this section.

2.1. Feature extraction for BCI

In order to select the most appropriate classifier for a given BCI system, it is essential to clearly understand what features are used, what their properties are and how they are used. This section aims at describing the common BCI features and more particularly their properties as well as the way to use them in order to consider time variations of EEG.

2.1.1. Feature properties

A great variety of features have been attempted to design BCI such as amplitude values of EEG signals [10], Band Powers (BP) [11], Power Spectral Density (PSD) values [12] [13], AutoRegressive (AR) and Adaptive AutoRegressive (AAR) parameters [8] [14], Time-frequency features [15] and inverse model-based features [16] [17] [18]. Concerning the design of a BCI system, some critical properties of these features must be considered:

- **noise and outliers:** BCI features are noisy or contain outliers because EEG signals have a poor signal-to-noise ratio;
- **high dimensionality:** In BCI systems, feature vectors are often of high dimensionality, e.g., [19]. Indeed, several features are generally extracted from several channels and from several time segments before being concatenated into a single feature vector (see next section);
- **time information:** BCI features should contain time information as brain activity patterns are generally related to specific time variations of EEG (see next section);

- **non-stationarity:** BCI features are non-stationary since EEG signals may rapidly vary over time and more especially over sessions;
- **small training sets:** The training sets are relatively small, since the training process is time consuming and demanding for the subjects.

These properties are verified for most features currently used in BCI research. However, it should be noted that it may no longer be true for BCI used in clinical practice. For instance, the training sets obtained for a given patient would not be small anymore as a huge quantity of data would have been acquired during sessions performed over days and months. As the use of BCI in clinical pratice is still very limited [3], this paper deals with classification methods used in BCI research. However, the reader should be aware that problems may be different for BCI used outside the laboratories.

2.1.2. Considering time variations of EEG

Most brain activity patterns used to drive BCI are related to particular time variations of EEG, possibly in specific frequency bands [1]. Therefore, the time course of EEG signals should be taken into account during feature extraction [20]. To use this temporal information, three main approaches have been proposed:

- **concatenation of features from different time segments:** It consists in extracting features from several time segments and concatenating them into a single feature vector [11] [20];
- **combination of classifications at different time segments:** It consists in performing the feature extraction and classification steps on several time segments and then combining the results of the different classifiers [21] [22];
- **dynamic classification:** It consists in extracting features from several time segments to build a temporal sequence of feature vectors. This sequence can be classified using a dynamic classifier [20] [23] (see Section 2.2.1).

The first approach is the most widely used, which explains why feature vectors are often of high dimensionality.

2.2. Classification algorithms

In order to choose the most appropriate classifier for a given set of features, the properties of the available classifiers must be known. This section provides a classifier taxonomy. It also deals with two classification problems especially relevant for BCI research, namely, the curse-of-dimensionality and the Bias-Variance tradeoff.

2.2.1. Classifier taxonomy

Several definitions are commonly used to describe the different kinds of available classifiers:

Generative-discriminative:

Generative (also known as informative) classifiers, e.g., Bayes quadratic, learn the class models. To classify a feature vector, generative classifiers compute the likelihood of each class and choose the most likely. Discriminative ones, e.g., Support Vector Machines, only learn the way of discriminating the classes or the class membership in order to classify a feature vector directly [24] [25];

Static-dynamic:

Static classifiers, e.g., MultiLayer Perceptrons, cannot take into account temporal information during classification as they classify a single feature vector. On the contrary, dynamic classifiers, e.g., Hidden Markov Model, can classify a sequence of feature vectors and thus, catch temporal dynamics [26].

Stable-unstable:

Stable classifiers, e.g., Linear Discriminant Analysis, have a low complexity (or capacity [27]). They are said stable as small variations in the training set does not affect considerably their performance. On the contrary, unstable classifiers, e.g., MultiLayer Perceptron, have a high complexity. As for them, small variations of the training set may lead to important changes in performances [28].

Regularized:

Regularization consists in carefully controlling the complexity of a classifier in order to prevent overtraining. A regularized classifier has good generalization performances and is more robust with respect to outliers [5] [9].

2.2.2. Main classification problems in BCI research

While performing a pattern recognition task, classifiers may be facing several problems related to the features properties such as outliers, overtraining, etc. In the field of BCI, two main problems need to be underlined: the curse-of-dimensionality and the Bias-Variance tradeoff.

The curse-of-dimensionality:

the amount of data needed to properly describe the different classes increases exponentially with the dimensionality of the feature vectors [9] [29]. Actually, if the number of training data is small compared to the size of the feature vectors, the classifier will most probably give poor results. It is recommended to use, at least, five to ten times as many training samples per class as the dimensionality [30] [31]. Unfortunatly this cannot be applied in all BCI systems as generally, the dimensionality is high and the training set small (see section 2.1.1). Therefore this "curse" is a major concern in BCI design.

The Bias-Variance tradeoff:

Formally, classification consists in finding the true label y^* of a feature vector x using a mapping f. This mapping is learnt from a training set T. The best mapping f^* that has generated the labels is, of course, unknown. If we consider the Mean Square Error (MSE), classification errors can be decomposed in three terms [28] [29]:

$$\begin{aligned} & MSE \\ & = E[(y^* - f(x))^2] \\ & = E[(y^* - f^*(x) + f^*(x) - E[f(x)] + E[f(x)] - f(x))^2] \\ & = E[(y^* - f^*(x))^2] + E[(f^*(x) - E[f(x)]^2)] \\ & + E[(E[f(x)] - f(x))^2] \\ & = Noise^2 + Bias(f(x))^2 + Var(f(x)) \end{aligned} \tag{1}$$

These three terms describe three possible sources of classification error:

- *Noise:* represents the noise within the system. This is an irreducible error;
- *Bias:* represents the divergence between the estimated mapping and the best mapping. Therefore, it depends on the method that has been chosen to obtain f (linear, quadratic, ...);
- *Variance:* reflects the sensitivity to the training set T used.

To attain the lowest classification error, both the Bias and the Variance must be low. Unfortunatly, there is a "natural" Bias-Variance tradeoff. Actually, stable classifiers tend to have a high Bias and a low Variance, whereas unstable classifiers have a low Bias and a high Variance. This can explain why simple classifiers sometimes outperform more complex ones. Several techniques, known as stabilization techniques, can be used to reduce the Variance. Among them, we can quote combination of classifiers [28] and regularization (see section 2.2.1).
EEG signals are known to be non-stationary. Training sets coming from different sessions are likely to be relatively different. Thus, a low Variance can be a solution to cope with the variability problem in BCI systems.

3. Survey of classifiers used in BCI research

This section surveys the classification algorithms used to design BCI systems. They are divided into five different categories: linear classifiers, neural networks, nonlinear bayesian classifiers, nearest neighbor classifiers and combinations of classifiers. The most popular are briefly described and their most important properties for BCI applications are highlighted.

3.1. Linear classifiers

Linear classifiers are discriminant algorithms that use linear functions to distinguish classes. They are probably the most popular algorithms for BCI applications. Two main

kinds of linear classifier have been used for BCI design, namely, Linear Discriminant Analysis (LDA) and Support Vector Machine (SVM).

3.1.1. Linear Discriminant Analysis

The aim of LDA (also known as Fisher's LDA) is to use hyperplanes to separate the data representing the different classes [5] [32]. For a two-class problem, the class of a feature vector depends on which side of the hyperplane the vector is (see Figure 1).

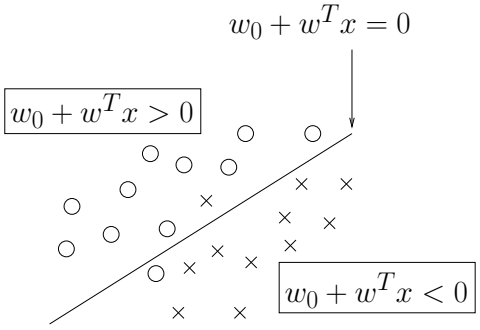

Figure 1. A hyperplane which separates two classes: the "circles" and the "crosses".

LDA assumes normal distribution of the data, with equal covariance matrix for both classes. The separating hyperplane is obtained by seeking the projection that maximize the distance between the two classes means and minimize the interclasse variance [32]. To solve an N-class problem ($N > 2$) several hyperplanes are used. The strategy generally used for multiclass BCI is the "One Versus the Rest" (OVR) strategy which consists in separating each class from all the others.

This technique has a very low computational requirement which makes it suitable for online BCI system. Moreover this classifier is simple to use and generally provides good results. Consequently, LDA has been used with success in a great number of BCI systems such as motor imagery based BCI [33], P300 speller [34], multiclass [35] or asynchronous [36] BCI. The main drawback of LDA is its linearity that can provide poor results on complex nonlinear EEG data [37].

A Regularized Fisher's LDA (RFLDA) has also been used in the field of BCI [38] [39]. This classifier introduces a regularization parameter C that can allow or penalize classification errors on the training set. The resulting classifier can accomodate outliers and obtain better generalization capabilities. As outliers are common in EEG data, this regularized version of LDA may give better results for BCI than the non-regularized version [39] [38]. Surprisingly, RFLDA is much less used than LDA for BCI applications.

3.1.2. Support Vector Machine

An SVM also uses a discriminant hyperplane to identify classes [40] [41]. However, concerning SVM, the selected hyperplane is the one that maximizes the margins, i.e.,

the distance from the nearest training points (see Figure 2). Maximizing the margins is known to increase the generalization capabilites [40] [41]. As RFLDA, an SVM uses a regularization parameter C that enables accomodation to outliers and allows errors on the training set.

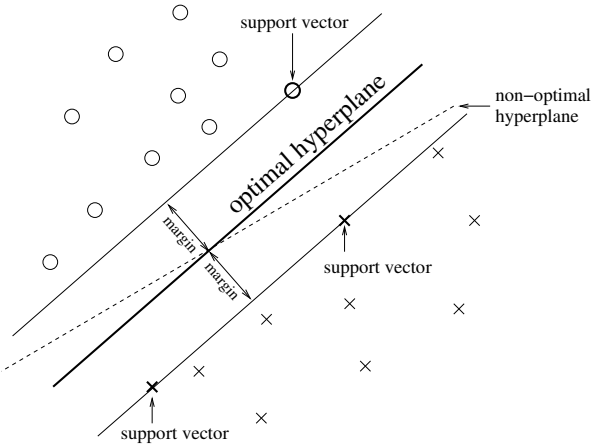

Figure 2. SVM find the optimal hyperplane for generalization.

Such an SVM enables classification using linear decision boundaries, and is known as linear SVM. This classifier has been applied, always with success, to a relatively large number of synchronous BCI problems [38] [35] [19]. However, it is possible to create nonlinear decision boundaries, with only a low increase of the classifier's complexity, by using the "kernel trick". It consists in implicitly mapping the data to another space, generally of much higher dimensionality, using a kernel function $K(x, y)$. The kernel generally used in BCI research is the Gaussian or Radial Basis Function (RBF) kernel:

$$K(x,y) = exp(\frac{-||x-y||^2}{2\sigma^2}) \tag{2}$$

The corresponding SVM is known as Gaussian SVM or RBF SVM [40] [41] RBF SVM have also given very good results for BCI applications [10] [35]. As LDA, SVM has been applied to multiclass BCI problems using the OVR strategy [42].

SVM have several advantages. Actually, thanks to the margin maximization and the regularization term, SVM are known to have good generalization properties [41] [9], to be insensitive to overtraining [9] and to the curse-of-dimensionality [40] [41]. Finally, SVM have a few hyperparameters that need to be defined by hand, namely, the regularization parameter C and the RBF width σ if using kernel 2. These advantages are gained at the expense of a low speed of execution.

3.2. Neural Networks

Neural Networks (NN) are, together with linear classifiers, the category of classifiers mostly used in BCI research (see, e.g., [43] [44]). Let us recall that a NN is an assembly of several artificial neurons which enables to produce nonlinear decision boundaries [45].

This section first describes the most widely used NN for BCI, which is the MultiLayer Perceptron (MLP). Then, it briefly presents other architectures of neural network used for BCI applications.

3.2.1. MultiLayer Perceptron

An MLP is composed of several layers of neurons: an input layer, possibly one or several hidden layers, and an output layer [45]. Each neuron's input is connected with the output of the previous layer's neurons whereas the neurons of the output layer determine the class of the input feature vector.

Neural Networks and thus MLP, are universal approximators, i.e., when composed of enough neurons and layers, they can approximate any continuous function. Added to the fact that they can classify any number of classes, this makes NN very flexible classifiers that can adapt to a great variety of problems. Consequently, MLP, which are the most popular NN used in classification, have been applied to almost all BCI problems such as binary [46] or multiclass [44], synchronous [20] or asynchronous [12] BCI. However, the fact that MLP are universal approximators makes these classifiers sensitive to overtraining, especially with such noisy and non-stationary data as EEG, e.g., [47]. Therefore, careful architecture selection and regularization is required [9].

A MultiLayer Perceptron without hidden layers is known as a perceptron. Interestingly enough, a perceptron is equivalent to LDA and, as such, has been sometimes used for BCI applications [18] [48]

3.2.2. Other Neural Network architectures

Other types of NN architecture are used in the field of BCI. Among them, one deserves a specific attention as it has been specifically created for BCI: the Gaussian classifier [49] [50]. Each unit of this NN is a Gaussian discriminant function representing a class prototype. According to its authors, this NN outperforms MLP on BCI data and can perform efficient rejection of uncertain samples [49]. As a consequence, this classifier has been applied with success to motor imagery [51] and mental task classification [49], particularly during asynchronous experiments [49] [52].

Besides the Gaussian classifier, several other NN have been applied to BCI purposes, in a more marginal way. They are not described here, due to space limitations:

- Learning Vector Quantization (LVQ) Neural Network [53] [54]
- Fuzzy ARTMAP Neural Network [55] [56];
- Dynamic Neural Networks such as the Finite Impulse Response Neural Network (FIRNN) [20], Time-Delay Neural Network (TDNN) or Gamma dynamic Neural Network (GDNN) [57];
- RBF Neural Network [5] [58];
- Bayesian Logistic Regression Neural Network (BLRNN) [8];

- Adaptive Logic Network (ALN) [59];
- Probability estimating Guarded Neural Classifier (PeGNC) [60].

3.3. Nonlinear Bayesian classifiers

This section introduces two Bayesian classifiers used for BCI: Bayes quadratic and Hidden Markov Model (HMM). Although Bayesian Graphical Network (BGN) has been employed for BCI, it is not described here as it is not common and, currently, not fast enough for real-time BCI [61] [62].

All these classifiers produce nonlinear decision boundaries. Furthermore, they are generative, which enables them to perform more efficient rejection of uncertain samples than discriminative classifiers. However, these classifiers are not as widespread as linear classifiers or Neural Networks in BCI applications.

3.3.1. Bayes quadratic

Bayesian classification aims at assigning to a feature vector the class it belongs to with the highest probability [5] [32]. The Bayes rule is used to compute the so-called *a posteriori* probability that a feature vector has of belonging to a given class [32]. Using the MAP (Maximum A Posteriori) rule and these probabilities, the class of this feature vector can be estimated.

Bayes quadratic consists in assuming a different normal distribution of data. This leads to quadratic decision boundaries, which explains the name of the classifier. Even though this classifier is not widely used for BCI, it has been applied with success to motor imagery [22] [51] and mental task classification [63] [64].

3.3.2. Hidden Markov Model

Hidden Markov Models (HMM) are popular dynamic classifiers in the field of speech recognition [26]. An HMM is a kind of probabilistic automaton that can provide the probability of observing a given sequence of feature vectors [26]. Each state of the automaton can modelize the probability of observing a given feature vector. For BCI, these probabilities usually are Gaussian Mixture Models (GMM), e.g., [23].

HMM are perfectly suitable algorithms for the classification of time series [26]. As EEG components used to drive BCI have specific time courses, HMM have been applied to the classification of temporal sequences of BCI features [23] [52] [65] and even to the classification of raw EEG [66]. HMM are not much widespread within the BCI community but these studies revealed that they were promising classifiers for BCI systems.

Another kind of HMM which has been used to design BCI is the Input-Output HMM (IOHMM) [12]. IOHMM is not a generative classifier but a discriminative one.

The main advantage of this classifier is that one IOHMM can discriminate several classes, whereas one HMM per class is needed to achieve the same operation.

3.4. Nearest Neighbor classifiers

The classifiers presented in this section are relatively simple. They consist in assigning a feature vector to a class according to its nearest neighbor(s). This neighbor can be a feature vector from the training set as in the case of k Nearest Neighbors (kNN), or a class prototype as in Mahalanobis distance. They are discriminative nonlinear classifiers.

3.4.1. k Nearest Neighbors

The aim of this technique is to assign to an unseen point the dominant class among its k nearest neighbors within the training set [5]. For BCI, these nearest neighbors are usually obtained using a metric distance, e.g., [38]. With a sufficiently high value of k and enough training samples, kNN can approximate any function which enables it to produce nonlinear decision boundaries.

KNN algorithms are not very popular in the BCI compmunity, probably because they are known to be very sensitive to the curse-of-dimensionality [29], which made them fail in several BCI experiments [42] [38] [39]. However, when used in BCI systems with low-dimensional feature vectors, kNN may prove to be efficient [67].

3.4.2. Mahalanobis distance

Mahalanobis distance based classifiers assume a Gaussian distribution $N(\mu_c, M_c)$ for each prototype of the class c. Then, a feature vector x is assigned to the class that corresponds to the nearest prototype, according to the so-called Mahalanobis distance $d_c(x)$ [52]:

$$d_c(x) = \sqrt{(x-\mu_c)M_c^{-1}(x-\mu_c)^T} \tag{3}$$

This leads to a simple yet robust classifier, which even proved to be suitable for multiclass [42] or asynchronous BCI systems [52]. Despite its good performances, it is still scarcely used in the BCI literature.

3.5. Combinations of classifiers

In most papers related to BCI, the classification is achieved using a single classifier. A recent trend, however, is to use several classifiers, aggregated in different ways. The classifier combination strategies used in BCI applications are the following:

Boosting:

Boosting consists in using several classifiers in cascade, each classifier focusing on

the errors committed by the previous ones [5]. It can build up a powerful classifier out of several weak ones, and it is unlikely to overtrain. Unfortunalty, it is sensible to mislabels [9] which may explain why it was not succesfull in one BCI study [68]. To date, in the field of BCI, boosting has been experimented with MLP [68] and Ordinary Least Square (OLS) [69];

Voting:
While using Voting, several classifiers are being used, each of them assigning the input feature vector to a class. The final class will be that of the majority [9]. Voting is the most popular way of combining classifiers in BCI research, probably because it is simple and efficient. For instance, Voting with LVQ NN [54], MLP [70] or SVM [19] have been attempted;

Stacking:
Stacking consists in using several classifiers, each of them classifying the input feature vector. These classifier are called level-0 classifiers. The output of each of these classifiers is then given as input to a so-called meta-classifier (or level-1 classifier) which makes the final decision [71]. Stacking has been used in BCI research using HMM as level-0 classifiers, and an SVM as meta-classifier [72];

The main advantage of such techniques is that a combination of similar classifiers is very likely to outperform one of the classifiers on its own. Actually, combining classifiers is known to reduce the Variance (see Section 2.2.2) and thus the classification error [29] [28].

3.6. Conclusion

A great variety of classifiers has been tried in BCI research. Their properties are summarized in Table 1. It should be stressed that some famous kinds of classifiers have not been attempted in BCI research. The two most relevant ones are decision trees [9] and the whole category of fuzzy classifiers [73]. Furthermore, different combination schemes of classifiers have been used, but several other efficient and famous ones can be found in the literature such as Bagging or Arcing [28] [9]. Such algorithms could prove useful as they all succeeded in several other pattern recognition problems. As an example, preliminary results using a fuzzy classifier for BCI purposes are promising [74].

4. Guidelines to choose a classifier

This section assesses the use of the algorithms presented in section 3. It aims at providing the readers with guidelines to help them choose a classifier adapted to a given context. The performances of the BCI using the classifiers described above are gathered in tables, in the appendix. Several measures of performance have been proposed in BCI, such as accuracy of classification, Kappa coefficient [42], Mutual Information [75], sensitivity and specificity [76]. The most common one is the accuracy of classification, i.e., the

Table 1. Properties of classifiers used in BCI research

	Linear	Non Linear	Gene-rative	Discri minant	Dynamic	Static	Regu-larized	Stable	Un-stable	High dimension robust
FLDA	X			X		X		X		
RFLDA	X			X		X	X	X		
linear-SVM	X			X		X	X	X		X
RBF-SVM		X		X		X	X	X		X
MLP		X		X		X			X	
BLR NN		X		X		X			X	
ALN NN		X		X		X			X	
TDNN		X		X	X				X	
FIRNN		X		X	X				X	
GDNN		X		X	X				X	
Gaussian NN		X		X		X			X	
LVQ NN		X		X		X			X	
Perceptron	X			X		X		X		
RBF-NN		X		X		X			X	
PeGNC		X		X		X	X		X	
fuzzy ARTMAP NN		X		X			X		X	
HMM		X	X		X				X	
IOHMM		X		X	X				X	
Bayes quadratic		X	X			X			X	
Bayes graphical network		X	X			X			X	
k-NN		X		X		X			X	
Mahalanobis distance		X		X		X			X	

percentage of correctly classified feature vectors. Consequently, this paper only considers this particular measure.

Two different points of view are proposed. The first identifies the best classifier(s) for a given kind of BCI whereas the second identifies the best classifier(s) for a given kind of features.

4.1. Which classifier goes with which BCI ?

Different classifiers were shown to be efficient according to the kind of BCI they were used in. More specifically, different results were observed between synchronous and asynchronous BCI.

4.1.1. The synchronous BCI

The synchronous case is the most widely spread. Three kinds of classification algorithms proved to be particularly efficient in this context, namely, Support Vector Machines, dynamic classifiers and combinations of classifiers. Unfortunatly they have

not been compared with each other yet. Justifications and possible reasons of such an efficiency are given hereafter.

Support Vector Machines: SVM reached the best results in several synchronous experiments, should it be in its linear [38] [42] or nonlinear form [10] [35], in binary [38] [37] [77] or multiclass [35] [42] BCI (see Tables A1, A3, A4 and A5). RFLDA shares several properties with SVM such as being a linear and regularized classifier. Its training algorithm is even very close to the SVM one. Consequently, it also reached very interesting results in some experiments [38] [39].

The first reason for this success may be regularization. Actually, BCI features are often noisy and likely to contain outliers [39]. Regularization may overcome this problem and increase the generalization capabilities of the classifier. As a consequence, regularized classifiers, and more particularly linear SVM, have outperformed unregularized ones of the same kind, i.e., LDA, during several BCI studies [39] [38] [42]. Similarly a nonlinear SVM has outperformed an unregularized nonlinear classifier, namely, an MLP, in another BCI study [35].

The second reason may be the simplicity of SVM. Indeed, the decision rule of SVM is a simple linear function in the kernel space which makes SVM stable and therefore, have a low Variance. Since BCI features are very unstable over time, having a low Variance may also be a key for low classification error in BCI.

The last reason probably is the robustness of SVM with respect to the curse-of-dimensionality (see Section 2.1.1). This has enabled SVM to obtain very good results even with very high dimensional feature vectors and a small training set [42] [19]. However, SVM are not drawback free for BCI as they generally are slower than other classifiers. Luckily, they are fast enough for real-time BCI, e.g., [78].

Dynamic classifiers: Dynamic classifiers almost always outperformed static ones during synchronous BCI experiments [23] [20] [57] (see Table A1). [79] is an exception, but the authors admitted that the chosen HMM architecture may not have been suitable. Dynamic classifiers probably are successfull in BCI because they can catch the relevant temporal variations present in the extracted features. Futhermore, classifying a sequence of low dimensional feature vectors, instead of a very high dimensional one, in a way, solves the curse-of-dimensionality. Finally, using dynamic classifiers in synchronous BCI also solves the problem of finding the optimal instant for classification as the whole time sequence is classified and not just a particular time window [23].

Combination of classifiers: Combining classifiers was shown efficient [19] [72] [69] and almost always outperformed a single one [19] [72] [70] (see Table A3). Similarly, on data set IIb of BCI competition 2003, the best results, i.e., best accuracy and smallest number of repetitions, were obtained with combinations of classifiers,

namely, Boosting of OLS [69] and Voting of SVM [19] (see Table A5). The study in [68] is an exception as a Boosting of MLP was outperformed by a single LDA. This may be explained by the sensitivity of boosting to mislabels [9] and the fact that these mislabels are likely to occur in such noisy and uncertain data as EEG signals. Therefore, combinations such as Voting or Stacking may be prefered for BCI applications.

As seen in Section 3.5, the combination of classifiers helps reducing the Variance component of the classification error which generally makes combinations of classifiers more efficient than their single counterparts [28]. Furthermore, this Variance reflects the sensitivity towards the training set used. In BCI experiments, Variance can be due to time variability [54] [21], session-to-session variability [19] or subject-to-subject variability. Therefore, Variance probably is an important source of error. Combining classifiers may be a way of solving this variability/non-stationarity problem [19], which may explain its success.

4.1.2. The asynchronous BCI

Few asynchronous BCI experiments have been carried out yet, therefore, no optimal classifier can be identified for sure. In this context, it seems that dynamic classifiers do not perform better than static ones [12] [52]. Actually, it is very difficult to identify the beginning of each mental task in asynchronous experiments. Therefore dynamic classifiers cannot use their temporal skills efficiently [12] [52]. Surprinsingly, SVM or combinations of classifiers have not been used in asynchronous BCI yet.

4.1.3. Partial conclusion

Concerning synchronous BCI, SVM seem to be very efficient regardless of the number of classes. This success may be explained by its good properties, namely, regularization, simplicity and immunity to the curse-of-dimensionality. Besides SVM, combination of classifiers and dynamic classifiers also seem to be very efficient and promising for synchronous BCI. Concerning the asynchronous experiments, due to the current lack of published results, no classifier seems better than the other. The only information that can be deduced is that dynamic classifiers seem to loose their superiority in such experiments.

4.2. Which classifier goes with which kind of features ?

To propose another point of view, this section compares classifiers considering their ability to cope with specific problems of BCI features (see Section 2.1.1):

- **noise and outliers:** regularized classifiers, such as SVM, seem appropriate to deal with outliers. Muller *et al* even recommanded to systematically regularize the

classifiers used in BCI systems in order to cope with outliers [39]. It is also argued that discriminative classifiers perform better than generative ones in presence of noise or outliers [12];

- **high dimensionality:** SVM probably are the most appropriate classifier to deal with feature vectors of high dimensionality. If the high dimensionality is due to the use of a large number of time segments, dynamic classifiers can also solve the problem by considering sequence of feature vectors instead of a single vector of very high dimensionality. For instance, SVM [10] [19] and dynamic classifiers such as HMM [66] or TDNN [57] are perfectly able to classify raw EEG. The kNN should not be used in such a case as they are very sensitive to the curse-of-dimensionality. Nevertheless, it is always preferable to have a small number of features [31]. Therefore, it is highly recommended to use dimensionality reduction techniques and/or features selection [39];
- **time information:** For synchronous experiments, dynamic classifiers seem to be the most efficient method to exploit the temporal information contained in features. Similarly, integrating classifiers over time can efficiently utilize the time information [22]. For asynchronous experiments, no clear superiority could be observed (see previous section);
- **non-stationarity:** A combination of classifiers may solve this problem as it reduces the Variance. Stable classifiers such as LDA or SVM can also be used but would probably be outperformed by combinations of LDA or SVM;
- **small training sets:** If the training set is small, simple techniques with few parameters should be used, such as LDA [30].

5. Conclusion

This paper has surveyed classification algorithms used to design Brain-Computer Interfaces (BCI). These algorithms were divided into five categories: linear classifiers, neural networks, nonlinear Bayesian classifiers, nearest neighbor classifiers and combinations of classifiers. The results they obtained, in a BCI context, have been analysed and compared in order to provide the readers with guidelines to choose or design a classifier for a BCI system. In a nutshell, it seems that SVM are particularly efficient for synchronous BCI. This probably is due to their regularization property and their immunity to the curse-of-dimensionality. Furthermore, combinations of classifiers and dynamic classifiers also seem very efficient in synchronous experiments.

This paper focused on reviewing classifiers used in BCI research, i.e., related to published online or offline studies. However, other existing classification techniques, not currently used for BCI purposes, could be explored and may prove to be rewarding. Furthermore, it should be noted that once BCI will be more widely used in clinical practice, new properties will have to be taken into consideration, such as the availability of large data sets or long term variability of EEG signals.

One difficulty encountered in such a study concerns the lack of published objective comparisons between classifiers. Ideally, classifiers should be tested within the same context, i.e., with the same users, using the same feature extraction method and the same protocol. Currently, this is a crucial problem for BCI research. For this reason some researchers have proposed general purpose BCI systems such as the BCI2000 toolkit [80]. This toolkit is a modular framework which makes it possible to easily change the classification, preprocessing or feature extraction modules. With such a system it becomes possible to test several classifiers with the same features and preprocessings. With similar objectives of modularity, the Open-ViBE platform [81] proposes a framework to experiment BCI on various protocols using, for instance, neuro-feedback and Virtual Reality. An extensive use of such platforms could lead to interesting findings in the future.