HAL open science

Statistical Learning for BCIs

Rémi Flamary, Alain Rakotomamonjy, Michèle Sebag

To cite this version:

Rémi Flamary, Alain Rakotomamonjy, Michèle Sebag. Statistical Learning for BCIs. Brain–Computer Interfaces 1: Foundations and Methods, John Wiley & Sons, Inc., 2016, 10.1002/9781119144977.ch9 . hal-02343066

HAL Id: hal-02343066

https://hal.science/hal-02343066v1

Submitted on 15 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

Statistical Learning for BCIs

Rémi FLAMARY, Alain RAKOTOMAMONJY and Michèle SEBAG

This chapter introduces statistical learning and its applications to brain–computer interfaces. We begin by presenting the general principles of supervised learning and discussing the difficulties raised by its implementation, with a particular focus on aspects related to selecting sensors and multisubject learning. This chapter also describes in detail how a learning approach may be validated, including various metrics of performance and optimization of the hyperparameters of the considered algorithms.

We invite the reader to experiment with the algorithms described here: the illustrative experiments included in this chapter may be reproduced using a Matlab/Octave toolbox[1], which contains the implementation details of the various different methods.

9.1. Supervised statistical learning

The goal of supervised learning is to construct a predictor function that assigns a label to any given example; this predictor function is constructed from labeled examples that provide a basis for this training process. The predictor function is obtained by optimizing a certain criterion, which includes a term for the empirical risk (the behavior of the function on the given examples) and a regularization term (which guarantees the behavior of the function on new examples).

Matlab/Octave toolbox: https://github.com/rflamary/mltool.

This section will describe the principles of supervised statistical learning, as well as the two most important algorithms in the current state of the art. The reader should refer to Lotte *et al.* [LOT 07] for a more detailed account of the state of the art.

9.1.1. *Training data and the predictor function*

The objective of supervised statistical learning is to estimate a predictor function $f(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathcal{Y}$ which, given an observation $\mathbf{x} \in \mathbb{R}^d$, predicts a label $y \in \mathcal{Y}$ [HAS 01, DUD 99]. The coordinates of the vector $\mathbf{x}$ are the descriptive features extracted from the observation $\mathbf{x}$ (see Chapters 7, 8 and 10). Traditionally, we distinguish between methods of classification (or discrimination), for which the label y is nominal (for example $\mathcal{Y} = \{-1, 1\}$ is a problem of binary classification) and methods of regression, for which the label y is real ($\mathcal{Y} = \mathbb{R}$).

In practice, the function $f(\cdot)$ is estimated from a set of n training examples $\mathcal{E} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathcal{Y}, i = 1, \ldots, n\}$. The target function $f(\cdot)$ must perform well on the given data (predict well the labels y_i of the training data $\mathbf{x}_i$) but must also, more importantly, perform well on future examples $\{\mathbf{x}_j, y_j\}$: we say that the function must *generalize* these data.

In this chapter, we will restrict attention to linear predictor functions, defined by:

$$f(\mathbf{x}) = \sum_{j=1}^{d} w_j x_j + b = \mathbf{x}^\top \mathbf{w} + b \qquad [9.1]$$

where $\mathbf{w} \in \mathbb{R}^d$ is a vector of weights, w_j is the weight of the jth coordinate of the decision and function and $b \in \mathbb{R}$ is the bias (constant). Linear predictor functions are the most commonly employed category of functions for brain–computer interfaces due to the fact that they are easy to interpret (the weight w_j may be thought of as the impact of the jth coordinate) and they are straightforward to train [BLA 06, TAN 12]. Note that linear functions (equation [9.1]) return real values; in the case of problems of binary classification, the predicted class is obtained by taking the sign of the function $f(\cdot)$ and not its value.

Depending on the BCI context, the supervised learning problem is either a problem of classification or regression. For example, tasks of motor imagery are generally problems of classification (Figure 9.1, left). The coordinates of $\mathbf{x}$, for example, correspond to the values of the power in a frequency band after CSP-type spatial filtering [LOT 11]. Similarly, a P300 speller task aims to detect (classify) event-related potentials directly in the signal. In this case, the features are, for example, given by the various different filtered and subsampled EEG signals [RAK 08]. The predictor function, or classifier, partitions the space into regions each of which corresponds to one class; the separating hyperplane is defined by the vector $\mathbf{w}$ (normal to the separating hyperplane). In contrast, tasks of movement prediction generally define regression problems; the objective could, for example, be to predict a real value that characterizes the position of a limb from the ECoG measurements (see Figure 9.1, right).

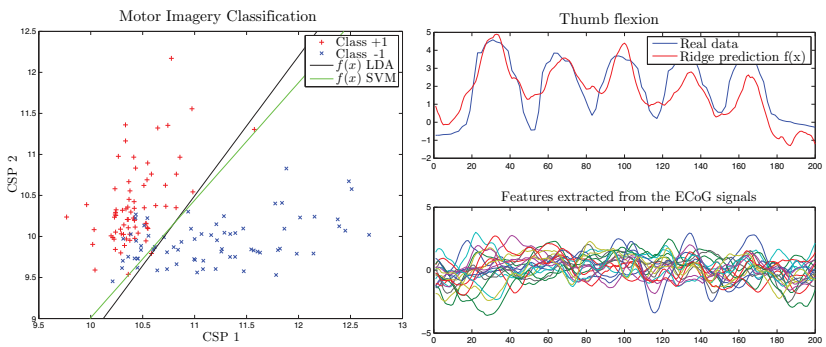

Figure 9.1. *Supervised learning: Data and prediction functions. Left: classification, motor imagery data with two CSP filters and a LDA classifier that partitions the space. Right: Regression, ECoG data with ridge regression. For a color version of this figure, see www.iste.co.uk/clerc/interfaces1.zip*

9.1.2. *Empirical risk and regularization*

As mentioned above, the objective of supervised statistical learning is to make as few mistakes as possible over the full set of possible data (consisting of both the training data and future data), or to be more precise, to minimize the

cost of all errors. The *empirical risk* is the average cost of the errors committed on the training data:

$$R_{\text{emp}}(f) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) \qquad [9.2]$$

where $L(y, f(\mathbf{x}))$ is the cost of replacing the label y with the prediction $f(\mathbf{x})$ for $\mathbf{x}$. The loss function L also plays an important role in performance metrics (section 9.3).

For classification, the best-known loss function is the $0-1$ cost function (Figure 9.2, left, shown in black); the cost of the error is 1 if the predictor function has the same sign as the expected class y, and 0 otherwise. However, the training problem defined by this cost function is difficult to solve: it requires the optimization of a non-differentiable, non-convex function. Other loss functions, such as the hinge loss [VAP 98, CHA 07] or the logistic cost [TOM 07] are therefore often preferred (Figure 9.2, left, and Table 9.1). Another alternative is the sigmoid cost function, traditionally used for neural networks.

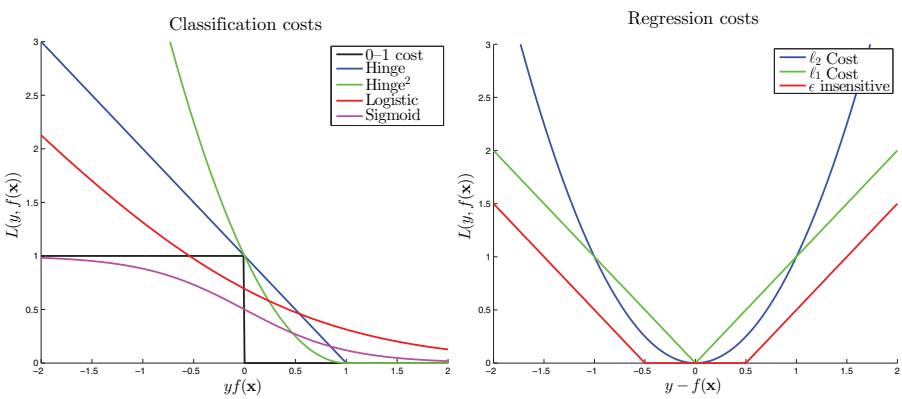

Figure 9.2. *Illustrations of a selection of different loss functions. Left: Classification. Right: Regression. For a color version of this figure, see www.iste.co.uk/clerc/interfaces1.zip*

For regression, the objective is to predict a real value y. The error is traditionally measured by the absolute value of the difference between y and

$f(\mathbf{x})$ [DRA 81] (ℓ_1 loss function) or its square (ℓ_2 loss function). Least squares or regularized least squares (ridge) regression uses the ℓ_2 loss function or mean square error. Another possible loss function is the ϵ-insensitive cost, which is equal to zero whenever the absolute value of the error is less than ϵ [SMO 04]. Using the absolute value (ℓ_1 or ϵ-insensitive) instead of the square (ℓ_2) of the error makes the process more robust against outliers (the impact of a term with a large error is smaller with ℓ_1 than with ℓ_2).

Classification cost	$L(y, f(\mathbf{x}))$
0-1 cost	$\mathbf{1}_{yf(\mathbf{x})<0}$
Hinge	$\max(0, 1 - yf(\mathbf{x}))$
Hinge^2	$\max(0, 1 - yf(\mathbf{x}))^2$
Logistic	$\log(1 + \exp(-yf(\mathbf{x})))$
Sigmoid	$(1 - \tanh(yf(\mathbf{x})))/2$

Reg. cost	$L(y, f(\mathbf{x}))$
ℓ_2 cost	$(y - f(\mathbf{x}))^2$
ℓ_1 cost	$\lvert y - f(\mathbf{x})\rvert$
ϵ-insensitive cost	$\max(0, \lvert y - f(\mathbf{x})\rvert - \epsilon)$

Table 9.1. *Illustrations of loss functions (Figure 9.2); Left: Classification; Right: Regression*

Note that in general minimizing the empirical risk is not sufficient (equation [9.2]) due to the phenomenon of *overfitting*: overfitting is when the function $f(\cdot)$ makes very few errors on the training data but produces high levels of error on subsequent data. This phenomenon occurs frequently when the number of examples n in the training data is small compared to the complexity of the examples (e.g. the number d of attributes). To avoid overfitting, we must limit both the empirical risk and the complexity of the function $f(\cdot)$ by introducing a regularization term $\Omega(f)$ [VAP 98]. The optimization problem therefore becomes:

$$\min_f \quad \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda\Omega(f) \qquad [9.3]$$

where $\lambda > 0$, the weight of the regularization term, is a parameter of the algorithm that requires validation (see section 9.4). When $f(\cdot)$ is linear, the ℓ_2 norm of the gradient $\mathbf{w}$ of $f(\mathbf{x})$ provides a good measure $\Omega(f)$ of the complexity, with $\Omega(f) = \sum_{i=1}^{d} w_i^2$. This type of regularization is used in practice for support vector machines (SVM) and ridge regression (section 9.1.3). Another possible measure of complexity is the ℓ_1 norm of $\mathbf{w}$, which has the advantage of implicitly selecting features, or sensors in the

context of BCIs, at the cost of introducing a non-differentiable term into the training criterion (section 9.2).

Depending on the nature of the optimization criterion (equation [9.3]), the solution can be found explicitly (e.g. by solving a linear system for ridge regression or linear discriminant analysis [LDA]), or by optimization methods such as the gradient descent method.

9.1.3. *Classical methods of classification*

This section will present two algorithms of linear classification, LDA and SVMs. These algorithms have been successfully applied to brain–computer interfaces, in particular in connection with the software package OpenVibe.

9.1.3.1. *Linear discriminant analysis*

LDA is a Bayesian approach that assumes that the positive (and negative) datapoints follow normal distributions in $\mathbb{R}^d$ given by $\mathcal{N}(\boldsymbol{\mu}_+, \boldsymbol{\Sigma})$ and $\mathcal{N}(\boldsymbol{\mu}_-, \boldsymbol{\Sigma})$), where $\boldsymbol{\mu}_+$ and $\boldsymbol{\mu}_-$ are the means of the normal distributions and $\boldsymbol{\Sigma}$ is the covariance matrix (which is assumed to be identical for both classes).

The decision function $f(\cdot)$ is trained by identifying $\boldsymbol{\mu}_+$, $\boldsymbol{\mu}_-$ and $\boldsymbol{\Sigma}$ and maximizing the likelihood of the training data. $f(\cdot)$ is linear with

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-), \qquad b = -\mathbf{w}^\top(\boldsymbol{\mu}_+ + \boldsymbol{\mu}_-)/2 \qquad [9.4]$$

The parameters $\boldsymbol{\Sigma}, \boldsymbol{\mu}_+, \boldsymbol{\mu}_-$ are estimated empirically using the training data. One of the limits of this approach is that depending on the dimension d of the problem and the number n of training datapoints, the covariance matrix may not be invertible. In practice, we take the inverse of the matrix $\tilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma} + \lambda \mathbf{I}_d$ where $\mathbf{I}_d$ is the identity matrix of dimension d. It can be shown that taking the inverse of $\tilde{\boldsymbol{\Sigma}}$ amounts to performing a quadratic regularization, which may be interpreted as assuming an *a priori* Gaussian distribution for the vector $\mathbf{w}$ with an assumed variance of $\sigma = 2/\sqrt{\lambda}$. Note that LDA is a special case of Fisher discriminant analysis (FDA), which is also widely used for BCIs [WAN 04]. The reader can refer to [HAS 01, Chapter 4] for more details on LDA and its quadratic extension Quadratic Discriminant Analysis

(QDA) when the covariances of each class are not equal. There exist many other extensions of this approach; for example, stepwise LDA allows features to be chosen using statistical tests [KRU 08]. The LDA method may also be extended to work for multiclass classification.

The LDA approach, which is widely used for brain–computer interfaces [BOS 04, BLA 04], is relatively simple to implement and does not involve any hyperparameters (in its non-regularized version; in the regularized version, the λ parameter is added). Among the 18 datasets used in the last BCI competitions [BLA 04, BLA 06, TAN 12], LDA and FDA were used in nine of the methods with the best performance in the classification category.

Logistic regression, a classification method that is also used for BCIs [TOM 07], introduces a logistic cost function (Table 9.1). Although this technique is less widely used than LDA or SVM, it performs excellently for linear classifications, particularly at higher dimensions.

9.1.3.2. *Support Vector Machines (SVM)*

Support Vector Machines or SVMs are obtained by minimizing a criterion of type equation [9.3], where the empirical risk term is the hinge loss (Table 9.1 and Figure 9.2). The most common regularization term is the ℓ_2 regularization term, which effectively maximizes the *margin*, i.e. the minimum distance between the points and the separating hyperplane.

Linear SVMs are a widely used category of classifiers for BCIs [RAK 08, KAP 04]; they have produced state-of-the-art results in detecting event-related potentials [RAK 08, LAB 10] and motor imagery [SCH 05a]. Specifically, SVMs were the best performers in six of 18 of the datasets of the three latest BCI competitions, in particular for the most recent datasets [BLA 04, BLA 06, TAN 12].

Note that SVMs may be extended to train nonlinear classifiers using the *kernel trick*, which replaces the scalar product with a similarity function and thus generates a more complex separating boundary. However, despite some very encouraging results in competitions [LAB 10], using nonlinear classifiers is often viewed as unnecessary for BCIs.

9.2. Specific training methods

In the context of brain–machine interfaces, the objective of methods of statistical learning is essentially to train the function describing the relationship between EEG signals and specific mental states (presence/absence of P300, predefined intentions of movement). However, more advanced, specific methods have also been developed to improve the performance of the process of decoding these mental states, but also to reduce the time required to calibrate the BCI system for new users. In the next few sections, we will present a selection of recent methods that address these key obstacles.

9.2.1. *Selection of variables and sensors*

For certain BCI paradigms such as BCI P300 spellers or BCI motor imagery, *a priori* neurophysiological knowledge allows us to place EEG sensors so that the quality of the recorded signal is nearly optimal. Nevertheless, in certain contexts, such as during the development of new BCI paradigms, or when the regions of interest in the cortex are damaged, it may be desirable to explicitly optimize the positioning of the sensors. This optimization might also be motivated by desire to improve performance by removing sensors that only provide marginal information. The literature on the topic of variable and sensor selection is very rich. We will only attempt a limited review of these methods; we encourage interested readers to refer to the references for more detail [GUY 03]. For BCIs, among the various different available methods, two particular strategies of sensor selection have been studied in most depth.

The first strategy finds the best-performing subset of sensors for recognizing mental states by successively eliminating sensors [LAL 04, SCH 05b, RAK 08]. The principle of the technique is to define a selection criterion (typically the margin of a SVM classifier or an estimate of generalized performance), denoted as C_r. The procedure begins by selecting all sensors and then eliminating one sensor per iteration. The elimination criterion is the following: at each iteration, the performance C_r is calculated for all remaining sensors, followed by the performance C_r^{-j} where the jth

sensor has been omitted. The sensor chosen for elimination is the sensor that minimizes:

$$|C_r - C_r^{-j}|$$

which is the minimum performance loss from eliminating a sensor. The stopping criterion is a predefined number of sensors, or too great a decrease in C_r. In computational terms, this elimination criterion may be very expensive to evaluate, however, in certain cases (margin criterion and linear SVM), it has an analytical expression that is easy to calculate [SCH 05b].

Another possible way to select variables and sensors in BCI systems is the technique of applying this selection process while the decision function is being trained. This can be achieved by choosing directly in equation [9.3] a regularization term that induces sparsity in the vector $\mathbf{w}$ and that implicitly selects variables or sensors. Bach et al. present a review of recent work in this area [BAC 12]. In the context of brain–machine interfaces, the most commonly used regularization terms are based on non-differentiable norms as follows:

– the norm $\ell_1(\mathbf{w}) = \sum_{i=1}^{d} |w_i|$. This norm generates unstructured sparsity in the variables comprising the vector $\mathbf{w}$. It is better suited for the selection of variables than for the selection of sensors;

– the mixed norm $\ell_{1,p}(\mathbf{w}) = \sum_{j=1}^{|\{G_j\}|} \left(\sum_{i \in G_j} |w_i|^p \right)^{1/p}$. Here, the sets $\{G_j\}$ form a disjointed partition of the indices 1 to d, so that the mixed norm is obtained by calculating the ℓ_1 norm of the vector of dimension $|\{G_j\}|$ whose jth component is given by the ℓ_p norm of the vector of elements indexed by G_j. This norm tends to induce grouped sparsity in the indices contained in a certain subset of the G_j. Thus, if the sets G_j are constructed so that they only contain references to variables linked to a specific sensor, using a regularization term of this type implicitly induces a process of channel selection.

Mixed norms were used for selecting groups of channels for the P300 speller BCI [TOM 10, FLA 14a] and the localization of EEG sources [STR 14]. Using more complexly structured representations such as kernel methods, Jrad *et al.* performed implicit sensor selection in multiple problems of BCI learning using only the ℓ_1 norm [JRA 11]. Although we have only discussed two types of norm here, other regularization terms may be constructed from *a priori* knowledge of the problem at hand, depending on the types of model that we wish to induce [BAC 12].

9.2.2. *Multisubject learning, information transfer*

One of the major hurdles for the widespread application of brain–machine interfaces is the current need for dedicated calibration specific to the person using the interface. Even today, in most cases, before any such interface can be effectively used, a session of EEG signal acquisition must first be organized in order to obtain training data. There are multiple avenues of research based on techniques of statistical learning that might possibly allow this hurdle to be overcome.

One of the possible approaches for building brain–machine interfaces that require less calibration with new users is to use training techniques based on information transfer, or multitask training techniques. These techniques attempt to train multiple decision functions simultaneously and share the set of available data. Thus, for brain–machine interfaces, this is equivalent to multisubject training. The objective of this approach is to attempt to compensate for the fact that only limited training data are available for a given classifier by transferring information from data that are available to other classifiers. This idea has been successfully implemented in several research projects. For example, Devlaminck *et al*. [DEV 11] used this principle to train spatial filters to adapt to subjects with very little available training data. Based on the same principles of multitask training, Alamgir *et al*. [ALA 10] suggest training EEG classifiers for different subjects by modeling their $\mathbf{w}$ vector as the realization of a random variable from a normal distribution. With this model, the average $\mathbf{w}$ vector is used for new subjects, and is subsequently updated as more data becomes available. Remaining within this multitask context, it is possible to select sensors common to multiple subjects using an appropriate mixed norm [FLA 14a]. Experiments show that when training a classifier to recognize P300 signals, significant performance gains are achieved for subjects that otherwise perform poorly without information transfer.

9.3. Performance metrics

This section presents a selection of the performance metrics used to evaluate and compare the advantages of different hypotheses or algorithms, considering the cases of classification and regression separately. These performance metrics, which must always be estimated from data distinct from

the training data (see section 9.4 [BLA 06, BLA 04]), test the generalization capacity of the trained hypotheses. Readers can refer to Schlögl et al. [SCH 07] for a more complete treatment of the topic of performance metrics.

9.3.1. *Classification performance metrics*

The traditional performance metrics used for classification problems are based on the confusion matrix $\mathbf{C}$, which for each pair of classes (i, j) records the number $C_{i,j}$ of instances of class i that have been assigned to class j. If the classification is perfect, the matrix $\mathbf{C}$ is diagonal. The accuracy (ACC) is the fraction of properly classified instances (which are on the diagonal of the confusion matrix), which is equal to the $0 - 1$ error (section 9.1.2) and estimates the Bayes error rate of the classifier. This rate was used in most BCI competitions to evaluate the performance of the motor imagery classifiers [BLA 06, BLA 04].

However, the accuracy is not a good measure of performance in the case of strongly unequal class sizes: indeed, if one class contains 99% of the datapoints, the trivial classifier that groups all points into the same class has an accuracy of 99%. An alternate, more appropriate metric is Cohen's Kappa coefficient [CAR 96], which was also used in BCI competitions [BLA 06, BLA 04], defined by

$$\kappa = \frac{ACC - p_e}{1 - p_e}, \quad \text{with} \quad p_e = \frac{1}{N^2} \sum_i (\sum_j C_{i,j})(\sum_j C_{j,i})$$

where p_e is the probability that a random classification is correct. The mutual information is another performance metric that is often used for BCIs [NYK 01]. It is based on information theory [BLA 06, SCH 07] and measures in bits the rate of information transmitted to the machine by the interface.

Another metric that is particularly well-suited for classes of unequal sizes is the area under the receiver operating characteristics (ROC) curve (AUC), or the Mann–Whitney–Wilcoxon score, given by:

$$AUC(f(\cdot)) = Pr(f(\mathbf{x}) > f(\mathbf{x}')|y > y')$$

Consider a binary classification hypothesis $f(\cdot)$ that takes values in $\mathbb{R}$. For any threshold $\tau \in \mathbb{R}$, we can define the classifier f_τ that classes $\mathbf{x}$ in the

positive class iff $f(\mathbf{x}) > \tau$, as well as the true positive rate TPR_τ ($P(f(\mathbf{x}) > \tau | y = 1)$) and the false positive rate FPR_τ ($P(f(\mathbf{x}) < \tau | y = 1)$). The monotone curve of the points (FPR_τ, TPR_τ) is called the ROC curve (Figure 9.3). If there exists a threshold such that the classifier $f(\cdot)_\tau$ is perfect, the ROC curve passes through (0,1) (0% false negatives and 100% true positives). If instead the ROC curve coincides with the diagonal, any improvement in the true positive rate is offset by an equal decrease in the false positive rate: in other words, the classifier provides no information. More generally, the area under the ROC curve measures the quality of the hypothesis $f(\cdot)$, regardless of whether the classes are of similar size (since the coordinates FPR_τ and TPR_τ are percentages).

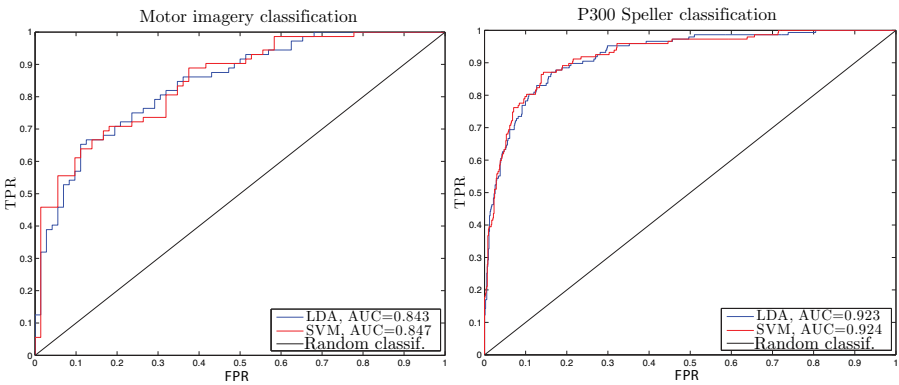

Figure 9.3. *ROC curves for an application in motor imagery (left) and P300 speller (right). Performance metric AUC: area under the ROC curve. Horizontal axis, false positive rate (FPR); vertical axis, true positive rate (TPR). For a color version of the figure, see www.iste.co.uk/clerc/interfaces1.zip*

The AUC criterion was used in the BCI MLSP 2010 competition [HIL 10]; for the P300 speller in particular, since by construction, the positive class contains far fewer instances than the negative class.

9.3.2. *Regression performance metrics*

For regression tasks, such as predicting the motion of limbs, other performance metrics need to be considered [PIS 08]. Similarly to

classification, the performance metrics are often linked to the cost term of the error, which is minimized during training. Thus, for least squares regression, the performance metric is the mean square error or ℓ_2 cost (Table 9.1). The disadvantage of this metric is that it is not normalized (it depends on the amplitude of the data y_i), which makes comparing performance difficult for tasks such as predicting finger flexion [LIA 12] and motion through space [PIS 08].

One possible alternative is to take the correlation between the predicted value $f(\mathbf{x})$ and the observed value y as a performance metric. This metric is equal to 1 in the case of monotone prediction ($f(\mathbf{x})$ increases with y); if the hypothesis provides zero information, the correlation is equal to 0. However, correlation does not take into account the true value of the prediction, which is important for tasks such as controlling the position of a cursor [WU 06].

9.4. Validation and model selection

Validating the results obtained in a given application serves two purposes in statistical learning: evaluating the chosen performance metric (section 9.3) and optimizing the hyperparameters of the algorithm. Readers can refer to [DUD 99, HAS 01, JAP 14] for a more in-depth treatment of the topic of validation.

9.4.1. *Estimation of the performance metric*

As mentioned earlier, the performance of a classifier $f(\cdot)$ on the training data is ultimately not what we are interested in; by increasing the complexity of $f(\cdot)$, it is always possible to construct a classifier that performs perfectly on the known data. The end-goal is the capacity of generalization of $f(\cdot)$, that is to say its performance on future data. We can attempt to evaluate the general performance of $f(\cdot)$ by partitioning the available data into training data, which is used to optimize $f(\cdot)$, and test data, which is used to estimate the performance of $f(\cdot)$ in general. In doing so, we assume that the test data extracted from the available dataset is a good representation of the future data that the classifier will need to predict.

If there is plenty of available data, estimating the performance metric is not particularly difficult: there are sufficient data to train a classifier $f(\cdot)$, and

there are sufficient (other) data for estimating the performance of $f(\cdot)$. But this task becomes more complex when there are limited available data, which is the case for BCIs due to the cost of data acquisition. In this context, we must find the right balance between reducing the quantity of training data (and therefore reducing the quality of the training hypothesis) and reducing the quantity of test data (and therefore reducing the reliability of the quality estimate).

In practice, there are three main approaches to this problem (Figure 9.4). The preferred approach depends on the way that the training data and the test data are constructed.

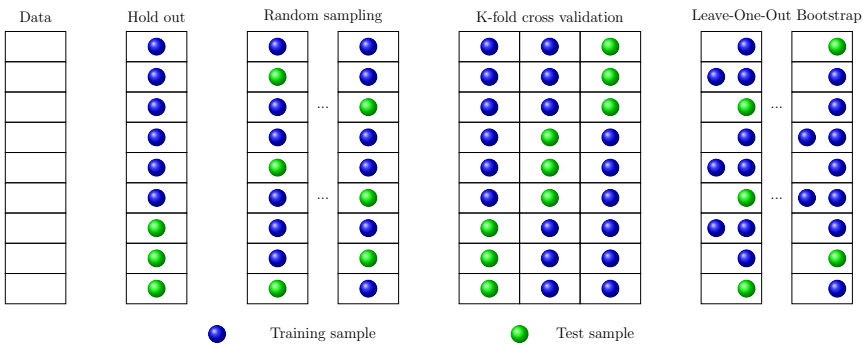

Figure 9.4. *Illustration of the different data partitions presented in this chapter used to estimate the generalization error*

9.4.1.1. *Random sampling*

The process known as the *hold-out* method randomly partitions the n datapoints into training data $\mathcal{E}$ and test data $\mathcal{T}$, uses $\mathcal{E}$ to train a hypothesis $f(\cdot)$ and then measures the performance of the hypothesis $f(\cdot)$ on $\mathcal{T}$. This performance might potentially have a large variance (if the datapoints in $\mathcal{E}$ are easy to classify, and those in $\mathcal{T}$ are difficult, or vice versa). To reduce the variance, the method of *random sampling* repeats the hold-out method by considering K independent samplings for the training data and the test data. The performances obtained on each of the different sets of test data are then averaged. The average performances and their confidence intervals for two different classifiers may be used as a basis for comparison to determine whether one is superior to the other to a certain degree of significance using a standard hypothesis test. Since training is repeated K times on independent

sets of training data, and the performances on the test data are averaged, this process makes the performance metric more robust against the randomness inherent in sampling at the cost of increasing the number of calculations by a factor of K.

9.4.1.2. *K-cross-validation and leave-one-out*

K-fold cross-validation partitions the n datapoints into K subsets of equal size. A series of hold-out processes is then performed by taking each of the subsets as the testing dataset, and the remaining data as the training dataset. As above, the performance is estimated by the average of the performances on the K testing subsets. Compared to random sampling, cross-validation reduces the variance of the estimate (each datapoint is included exactly once in the testing dataset). When the number of available datapoints is very low, we may in particular choose K equal to the number n of points: cross-validation is then equivalent to the *leave-one-out* (LOO) method, which performs training on all datapoints except one, saving the final point to test the trained classifier.

The choice of K involves establishing a compromise between the bias and the variance: as K increases, the bias of the estimate decreases (given some weak assumptions [HAS 01]) but the variance increases, because the various training datasets are strongly correlated. The complexity also increases with K. Finally, the choice of K fundamentally depends on the number n of datapoints. If n is large relative to the number d of descriptive attributes, low values of K may be considered sufficient ($K = 2 \ldots 10$), and the variance may be reduced by repeating cross-validation on five different partitions of the data.

For BCIs, cross-validation is normally used when the performance is good (see, for example, Labbé *et al.* [LAB 10] on estimating the performance of processes for recognizing P300 signals).

9.4.1.3. *Bootstrapping*

Bootstrapping constructs the training data using n uniform samples *with replacement* from the n available datapoints. Certain points can therefore be present multiple times in the training data, and others may be omitted; on average, the fraction of non-selected points is 37% of the available data. Classifier performance estimation is performed on the whole of the available dataset; as above, the performance is averaged over multiple different samples

for the training data. This method of estimation is less optimistic than LOO; the bias is similar to the bias of cross-validation with $K = 2$ [HAS 01]. Despite this bias, performance rankings of hypotheses using bootstrapping are usually considered reliable.

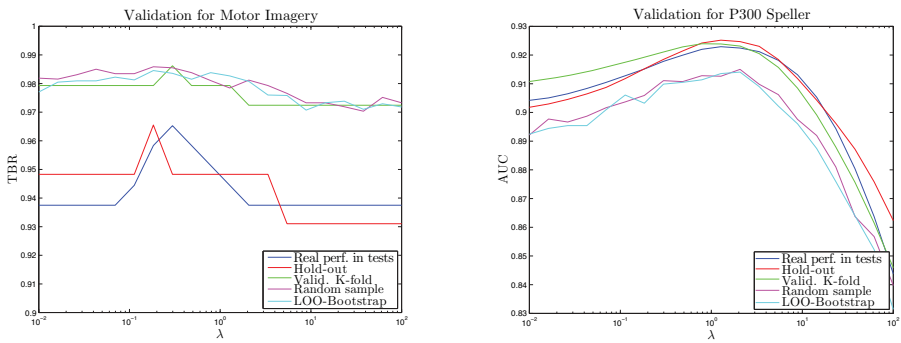

Figure 9.5. *Illustration of the different methods of estimating the generalization error on two different classification datasets. Left: Motor imagery classification with very few datapoints, performance metric = accuracy (ACC). Right: Detection of triggered potential for P300 speller, performance metric = area under the ROC curve (AUC). For a color version of the figure, see www.iste.co.uk/clerc/interfaces1.zip*

9.4.2. *Optimization of hyperparameters*

Training algorithms often involve hyperparameters, such as the weight of the regularization term (equation [9.3], λ parameter), or the kernel parameters of a nonlinear SVM (standard deviation of a Gaussian kernel, degree of a polynomial kernel). Naturally, it is desirable to find the values of hyperparameters that produce the optimal performance in generalization. The recommended procedure for doing so is as follows. The available data are partitioned into three sets: the training set $\mathcal{E}$, the test set $\mathcal{T}$ and the validation set $\mathcal{V}$.

Starting with $\mathcal{E}$ and a vector θ of hyperparameters, the performance is measured on the whole of the test set $\mathcal{T}$, denoted as $\mathcal{F}(\theta)$. The question may then be rephrased as finding the optimal vector of hyperparameters θ^*:

$$\theta^* = \arg\max\{\mathcal{F}(\theta)\}$$

After finding θ^* (see below), it is useful to recalculate the value of the performance metric. Indeed, since θ^* was determined using the information present in $\mathcal{E}$ and $\mathcal{T}$, the performance metric $\mathcal{F}(\theta)$ is optimistic on $\mathcal{T}$[2]. So, θ^* is used for training on $\mathcal{E} \bigcup \mathcal{T}$, and the classifier thus obtained is evaluated on the validation set $\mathcal{V}$, which has not yet been considered.

θ^* may be determined in one of three ways. If the algorithm involves a low number of hyperparameters ($\theta \in \mathbb{R}^d$, with $d = 2.3$), the usual method is an exhaustive search over a grid in the parameter space [CHA 11, RAK 08]. This grid may be regular (e.g. the degree of a polynomial taken within 1, 2,...,10) or not (e.g. varying the regularization weight by powers of 10, $\lambda = 10^i$, for $i = -3 \ldots 3$) . An example of exhaustive search for $\theta*$ is shown in Figure 9.5. The advantage of this method is its simplicity; however, the cost increases exponentially with the number of hyperparameters involved in the algorithm.

If the performance metric $\mathcal{F}$ (or an approximation or a bound for this metric) is differentiable, $\theta*$ may be found using gradient descent methods. This approach was used, for example, to optimize the hyperparameters of a SVM [CHA 02, FLA 14b], and for linear discriminant analysis with two classes [KEE 06]. Although complex to implement, this method has the advantage of being less costly than the previous method when the number of parameters is large.

Finally, if $\mathcal{F}$ is known only as a black box (an algorithm that returns the value of $\mathcal{F}(\theta)$ for each θ), we can use stochastic methods or black box optimization, ranging from simulated annealing to evolution-based strategies [HAN 01]. In general, stochastic optimization algorithms apply a dynamic distribution to the search space, which is gradually biased toward the regions of best performance. The *covariance–matrix–adaption* algorithm[3] achieves this by adapting the parameters of a Gaussian distribution so that it gradually converges toward an optimum of the target function. This method is more costly than the previous methods: it requires $\mathcal{F}(\theta)$ to be evaluated for a relatively large number of hyperparameter vectors θ, whereby calculating one single $\mathcal{F}(\theta)$ requires training and evaluating a classifier. Hybrid methods of optimization combining stochastic optimization of $\mathcal{F}$ and the training of an approximation of $\mathcal{F}$, called a surrogate model (*surrogate model-based*

2 It would be cheating to assume that $\mathcal{F}(\theta^*)$ represents the true performance of the classifier.

3 https://www.lri.fr/ hansen/cmaes_inmatlab.html.

optimization), are therefore used for optimizing the hyperparameters of general learning algorithms [BER 12, BAR 13], and, in the particular context of BCIs, for the selection of variables [GAR 03, SCH 03, COR 11].

9.5. Conclusions

This chapter gives a short presentation of the principal methods of supervised learning used with brain–computer interfaces, with a focus on the practical challenges posed by their implementation. References to some recent and promising work were included, so that readers may explore these topics in more depth according to their individual context. Finally, the validation of obtained results is an essential aspect of machine learning, both for optimizing the implementation of an algorithm and as a basis for a rigorous comparison of different algorithms.

9.6. Bibliography

[ALA 10] ALAMGIR M., GROSSE-WENTRUP M., ALTUN Y., "Multitask learning for brain–computer interfaces", *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pp. 17–24, 2010.

[BAC 12] BACH F., JENATTON R., MAIRAL J. *et al.*, "Optimization with sparsity-inducing penalties", *Foundations and Trends® in Machine Learning*, vol. 4, no. 1, pp. 1–106, 2012.

[BAR 13] BARDENET R., BRENDEL M., KÉGL B. *et al.*, "Collaborative hyperparameter tuning", *International Conference on Machine Learning (ICML)*, vol. 28, pp. 199–207, 2013.

[BEN 10] BÉNAR C.B., CHAUVIÈRE L., BARTOLOMEI F. *et al.*, "Pitfalls of high-pass filtering for detecting epileptic oscillations: a technical note on "false" ripples", *Clinical Neurophysiology*, vol. 121, pp. 301–310, 2010.

[BER 12] BERGSTRA J., BENGIO Y., "Random search for hyper-parameter optimization", *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[BLA 04] BLANKERTZ B., MÜLLER K., CURIO G. *et al.*, "The BCI competition 2003: progress and perspectives in detection and discrimination of EEG single trials", *IEEE Transactions on Biomedical Engineering,* vol. 51, no. 6, pp. 1044–1051, 2004.

[BLA 06] BLANKERTZ B., MÜLLER K., KRUSIENSKI D.J. *et al.*, "The BCI competition III: validating alternative approaches to actual BCI problems", *IEEE Transactions on Neural Systems and Rehabilitation Engineering* vol. 14, no. 2, pp. 153–159, 2006.

[BOS 04] BOSTANOV V., "BCI competition 2003-data sets Ib and IIb: feature extraction from event-related brain potentials with the continuous wavelet transform and the t-value scalogram", *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 1057–1061, 2004.

[CAR 96] CARLETTA J., "Assessing agreement on classification tasks: the kappa statistic", *Computational linguistics*, vol. 22, no. 2, pp. 249–254, 1996.

[CHA 02] CHAPELLE O., VAPNIK V., BOUSQUET O. *et al.*, "Choosing multiple parameters for support vector machines", *Machine Learning*, vol. 46, nos. 1–3, pp. 131–159, 2002.

[CHA 07] CHAPELLE O., "Training a support vector machine in the primal", *Neural Computation*, vol. 19, no. 5, pp. 1155–1178, 2007.

[CHA 11] CHANG C.-C., LIN C.-J., "LIBSVM: a library for support vector machines", *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[COR 11] CORRALEJO R., HORNERO R., ALVAREZ D., "Feature selection using a genetic algorithm in a motor imagery-based brain computer interface", *Annual International Conference of the IEEE on Engineering in Medicine and Biology Society (EMBC),* pp. 7703–7706, 2011.

[DEV 11] DEVLAMINCK D., WYNS B., GROSSE-WENTRUP M. *et al.*, "Multisubject learning for common spatial patterns in motor-imagery BCI", *Computational Intelligence and Neuroscience*, vol. 2011, p. 8, 2011.

[DIE 98] DIETTERICH T., "Approximate statistical tests for comparing supervised classification learning algorithms", *Neural Computation*, vol. 10, no. 1, pp. 1895–1923, 1998.

[DRA 81] DRAPER N., SMITH H., *Applied Regression Analysis*, John Wiley, New York, 1981.

[DUD 99] DUDA R.O., HART P.E., STORK D.G., *Pattern Classification*, John Wiley & Sons, Hoboken, NJ, 1999.

[FLA 14a] FLAMARY R., JRAD N., PHLYPO R. *et al.*, "Mixed-norm regularization for brain decoding", *Computational and Mathematical Methods in Medicine*, Hindhawi Publishing, Cairo, Egypt, vol. 2014, no. 1, pp. 1–13, 2014.

[FLA 14b] FLAMARY R., RAKOTOMAMONJY A., GASSO G., "Learning constrained task similarities in graph-regularized multi-task learning", in SUYKENS J.A.K. , SIGNORETTO M., ARGYRIOU A. (eds), *Regularization, Optimization, Kernels, and Support Vector Machines*, Chapman and Hall/CRC, 2014.

[GAR 03] GARRETT D., PETERSON D.A., ANDERSON C.W. *et al.*, "Comparison of linear, nonlinear, and feature selection methods for EEG signal classification", *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 11, no. 2, pp. 141–144, 2003.

[GUY 03] GUYON I., ELISSEEFF A., "An introduction to variable and feature selection", *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[HAN 01] HANSEN N., OSTERMEIER A., "Completely derandomized self-adaptation in evolution strategies", *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[HAS 01] HASTIE T., FRIEDMAN J., TIBSHIRANI R., *The Elements of Statistical Learning*, Springer, Berlin, 2001.

[HIL 10] HILD K.E., KURIMO M., CALHOUN V.D., "The sixth annual MLSP competition", *IEEE International Workshop on Machine Learning for Signal Processing (MLSP),* pp. 107–111, 2010.

[JAP 14] JAPKOWICZ N., SHAH M., *Evaluating Learning Algorithms A Classification Perspective*, Cambridge University Press, Cambridge, 2014.

[JRA 11] JRAD N., CONGEDO M., PHLYPO R. *et al.*, "sw-SVM: sensor weighting support vector machines for EEG-based brain–computer interfaces", *Journal of Neural Engineering*, vol. 8, no. 5, p. 056004, 2011.

[KAP 04] KAPER M., MEINICKE P., GROSSEKATHOEFER U. *et al.*, "BCI competition 2003-data set IIb: support vector machines for the P300 speller paradigm", *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 1073–1076, 2004.

[KEE 06] KEERTHI S.S., SINDHWANI V., CHAPELLE O., "An efficient method for gradient-based adaptation of hyperparameters in SVM models", *Advances in Neural Information Processing Systems*, vol. 19, pp. 673–680, 2006.

[KRU 08] KRUSIENSKI D.J., SELLERS E.W., MCFARLAND D.J. *et al.*, "Toward enhanced P300 speller performance", *Journal of Neuroscience Methods*, vol. 167, no. 1, pp. 15–21, 2008.

[LAB 10] LABBÉ B., TIAN X., RAKOTOMAMONJY A., "MLSP competition, description of third place method,", *IEEE International Workshop on Machine Learning for Signal Processing (MLSP),* pp. 116–117, 2010.

[LAL 04] LAL T.N., SCHRODER M., HINTERBERGER T. *et al.*, "Support vector channel selection in BCI", *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 1003–1010, 2004.

[LIA 12] LIANG N., BOUGRAIN L., "Decoding finger flexion from band-specific ECoG signals in humans", *Frontiers in Neuroscience*, vol. 6, no. 91, pp. 1–6, 2012.

[LOT 07] LOTTE F., CONGEDO M., LÉCUYER A. *et al.*, "A review of classification algorithms for EEG-based brain–computer interfaces", *Journal of Neural Engineering*, vol. 4, 2007.

[LOT 11] LOTTE F., GUAN C., "Regularizing common spatial patterns to improve BCI designs: unified theory and new algorithms", *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 2, pp. 355–362, 2011.

[NYK 01] NYKOPP T., Statistical modeling issues for the adaptive brain interface, Master's Thesis, Helsinki University of Technology, 2001.

[PIS 08] PISTOHL T., BALL T., SCHULZE-BONHAGE A. *et al.*, "Prediction of arm movement trajectories from ECoG-recordings in humans", *Journal of Neuroscience Methods*, vol. 167, no. 1, pp. 105–114, 2008.

[RAK 08] RAKOTOMAMONJY A., GUIGUE V., "BCI competition III: dataset II - ensemble of SVMs for BCI P300 speller", *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 3, pp. 1147–1154, 2008.

[SCH 03] SCHRODER M., BOGDAN M., HINTERBERGER T. *et al.*, "Automated EEG feature selection for brain computer interfaces", *First International IEEE EMBS Conference on Neural Engineering*, IEEE, Cancun, Mexico, pp. 626–629, 2003.

[SCH 05a] SCHLÖGL A., LEE F. *et al.*, "Characterization of four-class motor imagery EEG data for the BCI-competition", *Journal of Neural Engineering*, vol. 2, no. 4, p. L14, 2005.

[SCH 05b] SCHRÖDER M., LAL T.N., HINTERBERGER T. *et al.*, "Robust EEG channel selection across subjects for brain-computer interfaces", *EURASIP Journal on Applied Signal Processing*, vol. 2005, pp. 3103–3112, 2005.

[SCH 07] SCHLÖGL A., KRONEGG J., HUGGINS J.E. *et al.*, "Evaluation criteria for BCI research", *Toward Brain–computer Interfacing*, The MIT Press, Cambridge, Massachusetts, USA, 2007.

[SMO 04] SMOLA A.J., SCHÖLKOPF B., "A tutorial on support vector regression", *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[STR 14] STROHMEIER D., HAUEISEN J., GRAMFORT A., "Improved MEG/EEG source localization with reweighted mixed-norms", *International Workshop on Pattern Recognition in Neuroimaging*, IEEE, Tubingen, Allemagne, pp. 1–4, 2014.

[TAN 12] TANGERMANN M., MÜLLER K.-R., AERTSEN A. *et al.*, "Review of the BCI competition IV", *Frontiers in Neuroscience*, vol. 6, 2012.

[TOM 07] TOMIOKA R., AIHARA K., MÜLLER K.-R., "Logistic regression for single trial EEG classification", *Advances in Neural Information Processing Systems*, vol. 19, pp. 1377–1384, 2007.

[TOM 10] TOMIOKA R., MÜLLER K., "A regularized discriminative framework for EEG analysis with application to brain–computer interface", *NeuroImage*, vol. 49, no. 1, pp. 415–432, 2010.

[VAP 98] VAPNIK V., *Statistical Learning Theory*, Wiley, Hoboken, NJ, 1998.

[WAN 04] WANG Y., ZHANG Z., LI Y. *et al.*, "BCI competition 2003-data set IV: an algorithm based on CSSD and FDA for classifying single-trial EEG", *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 1081–1086, 2004.

[WU 06] WU W., GAO Y., BIENENSTOCK E., *et al.*, "Bayesian population decoding of motor cortical activity using a Kalman filter", *Neural Computation*, vol. 18, no. 1, pp. 80–118, 2006.