Interfaces cerveau-ordinateur : méthodes, applications et perspectives

R. Flamary, A. Rakotomamonjy, M. Sebag

February 5, 2015

Ce chapitre introduit l'apprentissage statistique et son application aux interfaces cerveau-machine. Dans un premier temps, le principe général de l'apprentissage supervisé est présenté et les difficultés de mise en œuvre sont discutées, en particulier les aspects relatifs à la sélection de capteurs et l'apprentissage multi-sujets. Ce chapitre détaille également la validation d'une approche d'apprentissage, incluant les différentes mesures de performance et l'optimisation des hyper-paramètres de l'algorithme considéré.

Le lecteur est invité à expérimenter les algorithmes décrits : une boite à outils Matlab/Octave[1] permet de reproduire les expériences illustrant le chapitre et contient les détails d'implémentation des différentes méthodes.

1 Apprentissage statistique supervisé

L'apprentissage supervisé a pour but de construire une fonction de prédiction, associant une étiquette à tout exemple ; cette fonction de prédiction est construite à partir d'exemples étiquetés formant une base d'apprentissage. La fonction de prédiction est obtenue par optimisation d'un critère, faisant intervenir d'une part le risque empirique (le comportement de la fonction sur les exemples disponibles) et d'autre part un terme de régularisation (garantissant le comportement de la fonction sur de nouveaux exemples).

Cette partie décrit le principes de l'apprentissage statistique supervisé, ainsi que les deux algorithmes principaux de l'état de l'art. Le lecteur pourra se reporter à Lotte *et al.* [1] pour un état de l'art plus détaillé.

1.1 Données d'apprentissage et fonction de prédiction

Le but d'une méthode d'apprentissage statistique supervisé est d'estimer une fonction de prédiction $f(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathcal{Y}$ permettant de prédire une étiquette $y \in \mathcal{Y}$ à partir d'une observation $\mathbf{x} \in \mathbb{R}^d$ [2, 3]. Les coordonnées du vecteur $\mathbf{x}$ sont les caractéristiques descriptives extraites pour l'observation $\mathbf{x}$ donnée (voir chapitres 7, 8 et 10). On distingue classiquement les méthodes de classification (ou discrimination) lorsque l'étiquette y est nominale (par exemple $\mathcal{Y} = \{-1, 1\}$ correspond à un problème de classification binaire) et les méthodes de régression lorsque l'étiquette y est réelle ($\mathcal{Y} = \mathbb{R}$).

En pratique, la fonction $f(\cdot)$ est estimée à partir d'un ensemble de n exemples d'apprentissage $\mathcal{E} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathcal{Y}, i = 1, \ldots, n\}$. La fonction $f(\cdot)$ cherchée doit rendre compte des données disponibles (prédire au mieux les labels y_i des données d'apprentissage $\mathbf{x}_i$) mais aussi et surtout rendre compte des exemples $\{\mathbf{x}_j, y_j\}$ futurs : on dit qu'elle doit *généraliser* les données disponibles.

Nous nous limitons dans ce chapitre à des fonctions de prédiction linéaire, définies par:

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

où $\mathbf{w} \in \mathbb{R}^d$ est un vecteur de poids, w_j étant le poids de la j-ème coordonnée dans la fonction de décision et $b \in \mathbb{R}$ est un biais (constant). Les fonctions de prédiction linéaire sont les plus utilisées dans le domaine des interfaces cerveau-machine en raison de leur facilité d'interprétation (le poids w_j est interprété comme l'impact de la j-ème coordonnée) et leur facilité d'apprentissage [4, 5]. Notons qu'une fonction linéaire (Eq. (1)) retourne une valeur réelle ; dans le cas d'un problème de classification binaire, la classe prédite est obtenue en prenant le signe de la fonction $f(\cdot)$ et non sa valeur.

En fonction du contexte BCI, le problème d'apprentissage supervisé posé est un problème de classification ou de régression. Par exemple, les tâches d'imagerie motrice correspondent en général à un problème de classification (Fig. 1, gauche). Les coordonnées de $\mathbf{x}$ correspondent par exemple aux puissances dans une bande de fréquence après un filtrage spatial de type CSP [6]. De même, une tâche de P300 speller cherche à détecter (classifier) un potentiel évoqué directement dans le signal, où les caractéristiques sont par exemple les différents signaux EEGs filtrés et sous-échantillonnés [7]. La fonction de prédiction ou classifieur partitionne l'espace en zones correspondant chacunes à une classe ; l'hyperplan séparateur est défini par le vecteur $\mathbf{w}$ (normale à l'hyperplan séparateur).

[1] Boite à outils Matlab/Octave: `https://github.com/rflamary/mltool`

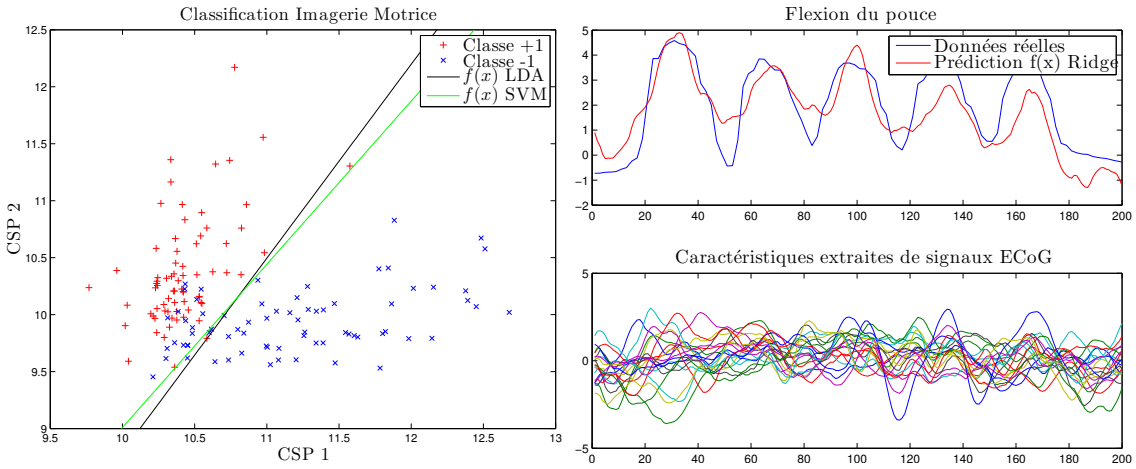

Figure 1: Apprentissage supervisé : Données et fonctions de prédiction. Gauche: Classification, données d'imagerie motrice avec 2 filtres CSP et un classifieur LDA qui partitionne l'espace. Droite: Régression, données ECoG avec une régression ridge.

Coût Classif.	$L(y, f(\mathbf{x}))$
Coût 0-1	$\mathbf{1}_{yf(\mathbf{x})<0}$
Hinge	$\max(0, 1 - yf(\mathbf{x}))$
Hinge^2	$\max(0, 1 - yf(\mathbf{x}))^2$
Logistique	$\log(1 + \exp(-yf(\mathbf{x})))$
Sigmoïde	$(1 - \tanh(yf(\mathbf{x})))/2$

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

Table 1: Fonctions de perte illustrées Fig. 2. Gauche: Classification. Droite: régression.

Au contraire, une tâche de prédiction de mouvement définit en général un problème de régression ; l'objectif est de prédire un réel caractérisant par exemple la position d'un membre à partir des mesures ECog (cf Chapitre 10; Fig. 1, droite).

1.2 Risque empirique et régularisation

Comme dit précédemment, l'objectif de l'apprentissage statistique supervisé est de faire peu d'erreurs sur l'ensemble des données possibles (d'une part les données d'apprentissage et d'autre part les données futures), et plus précisément de minimiser le coût des erreurs commises. On appelle *risque empirique* (ou fonction d'attache aux données) le coût moyen des erreurs commises sur les données d'apprentissage:

$$R_{emp}(f) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) \tag{2}$$

où $L(y, f(\mathbf{x}))$ mesure le coût de remplacer l'étiquette y par la prédiction $f(\mathbf{x})$ pour l'exemple $\mathbf{x}$. La fonction de perte L est également très liée aux mesures de performances (section 3).

Dans le cas de la classification, la fonction de perte la plus connue est la fonction de coût $0 - 1$ (Fig. 2, gauche, illustrée en noir) ; le coût de l'erreur est 1 si la fonction de prédiction est de même signe que la classe attendue y et 0 sinon. Cependant, cette fonction de coût définit un problème d'apprentissage difficile : elle induit l'optimisation d'une fonction non différentiable et non convexe. D'autres fonctions de perte, telles la perte charnière (hinge loss) [8, 9] ou le coût logistique [10] sont donc souvent préférés (Fig. 2, gauche, et Table 1). Une autre alternative est la fonction coût sigmoïde, classiquement utilisée dans le cadre des réseaux de neurones.

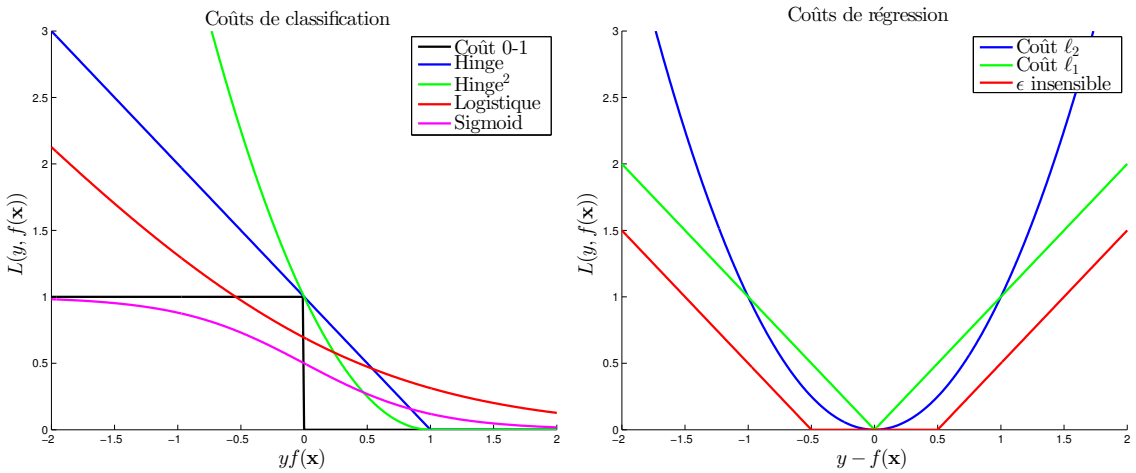

Figure 2: Illustration des différentes fonctions de perte. Gauche: Classification. Droite: Régression

Dans le cadre de la régression, on cherche à prédire une valeur réelle y. L'erreur est classiquement mesurée par la valeur absolue de la différence entre y et $f(\mathbf{x})$ [11] (fonction de perte ℓ_1), ou son carré (fonction de perte ℓ_2). La régression au sens des moindres carrés ou des moindres carrés régularisés (ridge) utilise la fonction de perte ℓ_2, ou MSE (*Mean Square Error*). Une autre fonction de perte est le coût ϵ-insensible qui reste à zéro tant que la valeur absolue de l'erreur est inférieure à ϵ [12]. L'utilisation de la valeur absolue (ℓ_1 ou ϵ-insensible) au lieu du carré (ℓ_2) de l'erreur permet d'être plus robuste par rapport aux points aberrants (l'impact d'un exemple avec une grande erreur est plus limité dans le cas ℓ_1 que ℓ_2).

Notons qu'on ne peut pas en général se contenter de minimiser le risque empirique (Eq. (2)) en raison du phénomène de *sur-apprentissage* : le sur-apprentissage consiste à trouver une fonction $f(\cdot)$ avec une très faible erreur sur les données d'apprentissage, qui fait beaucoup d'erreurs sur les exemples ultérieurs. Ce phénomène est fréquent lorsque les données d'apprentissage comprennent un nombre d'exemples n qui est faible relativement à leur complexité (e.g. le nombre d d'attributs). La parade consiste à limiter à la fois le risque empirique et la complexité de la fonction $f(\cdot)$, en introduisant un terme de régularisation $\Omega(f)$ [8]. Le problème d'optimisation considéré devient alors :

$$\min_f \quad \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda\Omega(f) \tag{3}$$

où $\lambda > 0$, le poids du terme de régularisation, est un paramètre de l'algorithme qui doit être validé (voir section 4). Pour le cas de fonctions $f(\cdot)$ linéaires, une bonne mesure de complexité $\Omega(f)$ est la norme ℓ_2 du gradient $\mathbf{w}$ de $f(\mathbf{x})$, avec $\Omega(f) = \sum_{i=1}^{d} w_i^2$. Cette régularisation est utilisée en pratique dans le cadre des machines à vecteurs supports (SVM) et de la régression ridge (section 1.3). Une autre mesure de complexité est la norme ℓ_1 de $\mathbf{w}$, avec l'avantage de conduire à la sélection des attributs ou des capteurs dans le cadre BCI, au prix d'introduire un terme non-différentiable dans le critère d'apprentissage (section 2).

Selon la nature du critère d'optimisation (Eq. (3)), la solution pourra être obtenue de façon explicite (e.g. par résolution d'un système linéaire pour la régression ridge, ou l'analyse linéaire discriminante), ou par des méthodes d'optimisation de type descente de gradient.

1.3 Méthodes de classification classiques

Cette section présente les deux algorithmes de classification linéaire, analyse discriminante linéaire (LDA) et machines à vecteurs supports (SVM), qui ont été appliqués avec succès aux interfaces cerveau-machine, en particulier dans le cadre du logiciel OpenVibe.

1.3.1 Analyse linéaire discriminante (LDA)

L'analyse linéaire discriminante est une approche Bayesienne, supposant que les exemples positifs (resp. négatifs) suivent une distribution gaussienne sur $\mathbb{R}^d$, donnée par $\mathcal{N}(\boldsymbol{\mu}_+, \boldsymbol{\Sigma})$ (resp. $\mathcal{N}(\boldsymbol{\mu}_-, \boldsymbol{\Sigma})$), où $\boldsymbol{\mu}_+$ et $\boldsymbol{\mu}_-$ sont les centres des gaussiennes et $\boldsymbol{\Sigma}$ la matrice de covariance (supposée identique pour les deux classes).

L'apprentissage de la fonction de décision $f(\cdot)$ procède en identifiant $\boldsymbol{\mu}_+$, $\boldsymbol{\mu}_-$ et $\boldsymbol{\Sigma}$ en maximisant la vraisemblance des données d'apprentissage. La fonction de décision $f(\cdot)$ associée est linéaire avec

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-), \qquad b = -\mathbf{w}^\top(\boldsymbol{\mu}_+ + \boldsymbol{\mu}_-)/2 \tag{4}$$

Les paramètres $\boldsymbol{\Sigma}, \boldsymbol{\mu}_+, \boldsymbol{\mu}_-$ sont estimés empiriquement à partir des données d'apprentissage. Une des limites de cette approche est que selon la dimensionnalité du problème d et le nombre d'exemples d'apprentissage n, la matrice de covariance estimée peut ne pas être inversible. En pratique, on considère l'inverse de la matrice $\tilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma} + \lambda \mathbf{I}_d$ où $\mathbf{I}_d$ est la matrice identité de dimension d. On peut montrer que le fait de considérer l'inverse de $\tilde{\boldsymbol{\Sigma}}$ correspond à une régularisation quadratique, interprétée comme un *a priori* gaussien sur le vecteur $\mathbf{w}$ en supposant une variance $\sigma = 2/\sqrt{\lambda}$. Notons que LDA est un cas particulier de l'analyse discriminante de Fisher (FDA), également très utilisée en BCI [13]. Le lecteur pourra se reporter à [2, Chapitre 4] pour plus de détails sur LDA et son extension quadratique QDA lorsque les covariances de chaque classe sont différentes. Il existe de nombreuses extensions de cette approche ; par exemple le stepwise-LDA permet de sélectionner des caractéristiques à l'aide de tests statistiques [14]. La méthode LDA s'étend aussi au contexte multiclasse.

L'approche LDA, particulièrement utilisée pour les interfaces cerveau-machine [15, 16], est relativement simple à mettre en oeuvre et ne fait intervenir aucun hyper-paramètre (dans sa version non régularisée; dans sa version régularisée, il faut ajuster le paramètre λ). Parmi les 18 jeux de données utilisés lors des dernières compétitions BCI [16, 4, 5], LDA et FDA ont été utilisées dans neuf des méthodes menant aux meilleures performances de classification.

La régression logistique, une méthode de classification également utilisée en BCI [10], fait intervenir un coût logistique (Table 1). Quoique moins utilisée que LDA ou SVM, elle permet également d'obtenir de très bonnes performances en classification linéaire et tout particulièrement en grande dimension.

1.3.2 Séparateurs à Vaste Marge (SVM)

Les séparateurs à vaste marge, ou machines à vecteurs supports (SVM) sont obtenus par minimisation d'un critère de type Eq. (3), où le terme d'attache aux données est le coût charnière (hinge loss, Table 1, Fig.2). Le terme de régularisation le plus courant est la régularisation ℓ_2, qui conduit à maximiser la *marge*, i.e. la distance minimum des points à l'hyperplan séparateur.

Les SVM linéaires sont des classifieurs largement utilisés en BCI [17, 18]; ils ont obtenu des résultats au niveau de l'état de l'art en détection de potentiels évoqués [17, 19] et en imagerie motrice [20]. Spécifiquement, les SVM ont obtenu les meilleures performances pour 6 des 18 jeux de données lors des 3 dernières compétitions BCI, tout particulièrement les plus récentes [16, 4, 5].

Notons que les SVM peuvent être étendus pour apprendre des classifieurs non linéaires au moyen de l'astuce des noyaux (*kernel trick*), remplaçant le produit scalaire par une fonction de similarité et obtenant une frontière de séparation plus complexe. Cependant, malgré certains résultats très encourageants en compétition [19], l'utilisation de classifieur non-linéaire est souvent considéré comme superflue en BCI.

2 Méthodes d'apprentissage spécifiques

Dans le cadre des interfaces cerveau-machine, les méthodes d'apprentissage statistique ont essentiellement pour but d'apprendre une relation entre des signaux EEGs et des états mentaux spécifiques (présence/absence de P300, un mouvement imaginé prédéfini). Cependant, des méthodes spécifiques plus avancées ont également été développées dans la perspective d'amélioration des performances de décodage de ces états mentaux, mais également dans un but de réduction du temps de calibration du système BCI lors de l'usage par un nouvel

utilisateur. Dans les paragraphes suivants, nous présentons des méthodes récentes visant à attaquer ces verrous.

2.1 Sélection de variables et de capteurs

Pour certains paradigmes de BCI tels que les BCI P300 speller ou les BCI imagerie moteur, les connaissances a priori neurophysiologiques permettent de placer les capteurs EEGs de manière quasi-optimale en terme de qualité du signal enregistré. Cependant, il existe des situations, par exemple lorsque l'on vise à developper de nouveaux paradigmes BCI ou lorsque les régions du cortex d'interêt sont endommagées, où il peut être judicieux d'optimiser ces localisations. L'optimisation de la localisation des capteurs peut être également motivée par un objectif d'amélioration des performances en supprimant les capteurs qui n'apportent que des informations marginales. Il existe une littérature très florissante sur les méthodes de sélection de variables et de capteurs. Nous ne proposons ici qu'une revue limitée de ces méthodes et nous engageons le lecteur intéressé à se réferer à des travaux de références pour plus de détails [21]. Dans le cadre des BCI, parmi les différentes méthodes possibles, deux approches de sélection de capteurs ont été étudiées en profondeur.

La première méthode vise à trouver un sous-ensemble de capteurs qui soit le plus performant en terme de reconnaissance des états mentaux, par le biais de technique d'élimination successives de capteurs [22, 23, 7]. Le principe consiste d'abord, à définir un critère de sélection (typiquement la marge d'un classifieur SVM ou une estimation de la performance en généralisation) que l'on appelle ici C_r. On commence la procédure en sélectionnant tous les capteurs et, on en élimine un à chaque itération. Le critère d'élimination est comme suit: on calcule à chaque itération la performance C_r avec tous les capteurs restants, puis la performance C_r^{-j} lorsque l'on élimine le j-ième capteur parmi pour tous les capteurs restants. Le capteur éliminé est celui qui minimise

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

c'est à dire la perte de performance minimum liée à l'élimination du capteur. Le critère d'arret est un nombre prédéfini de capteurs restants ou une décroissance trop brutale de C_r. Au niveau computationnel, ce critère d'élimination peut être très couteux à évaluer, cependant dans certains cas (critère de marge et SVM linéaire), il a une expression analytique facile à calculer [23].

Une autre alternative possible pour la sélection de variables et de capteurs dans les systèmes BCI consiste à mettre en oeuvre ce processus de sélection lors de l'apprentissage de la fonction de décision. Cela est possible en choisissant directement dans l'équation (3) un terme de régularisation qui induit la parcimonie du vecteur $\mathbf{w}$ et qui implicitement impose une sélection de variables ou de capteurs. Une revue des récents travaux sur ce domaine est présentée par Bach et al. [24]. Dans le cadre des interfaces cerveau-machine, les termes de régularisation utilisés sont essentiellement basés sur des normes non-différentiables

- la norme $\ell_1(\mathbf{w}) = \sum_{i=1}^{d} |w_i|$. Cette norme permet de générer une parcimonie non-structurée sur les variables composant le vecteur $\mathbf{w}$. Elle est plus adaptée à la sélection de variables qu'à la sélection de capteur.

- la norme mixte $\ell_{1,p}(\mathbf{w}) = \sum_{j=1}^{|\{G_j\}|} \left(\sum_{i \in G_j} |w_i|^p \right)^{1/p}$. Ici, les ensembles $\{G_j\}$ forment une partition disjointe des indices de 1 à d et la norme mixte agit donc en calculant la norme ℓ_1 du vecteur de dimension $|\{G_j\}|$ dont la j-iéme composante est donnée par la norme ℓ_p du vecteur dont les éléments sont indicés par G_j. Cette norme a tendance à induire une parcimonie groupée sur les indices contenus dans certains G_j. Ainsi, si les ensembles G_j sont construits de sorte qu'ils ne fassent références qu'aux variables liées à un capteur donné, l'utilisation d'un terme de régularisation de ce type induit implicitement un processus de sélection de canaux.

Les normes mixtes ont été utilisées pour la sélection de groupes de canaux pour le BCI P300 Speller [25, 26] et la localisation de sources EEG [27]. Par le biais d'une structure plus complexe de représentation telle que les méthodes à noyaux, Jrad et al. se limite à l'utilisation d'une norme ℓ_1 pour induire une sélection de capteurs dans plusieurs problèmes d'apprentissage BCI [28]. Bien que nous n'ayons discuté ici que de deux types de normes, d'autres termes régularisants peuvent être construits en fonction des connaissances *a priori* disponibles sur le problème et en fonction des modélisations que l'on désire induire [24].

2.2 Apprentissage multi-sujets, transfert d'information

Un des verrous majeur pour la diffusion des interfaces cerveau-machine est leur besoin actuel d'une calibration dédiée à la personne utilisant l'interface. Ainsi, à ce jour, dans la plus plupart des cas, avant une utilisation effective, une séance d'acquisition de signaux EEGs visant à obtenir des données d'apprentissage doit être mise en place. Plusieurs voies de recherche, basées sur des techniques d'apprentissage statistique, sont possibles pour lever ce verrou.

Une des approches possibles pour obtenir des interfaces cerveau-machine necessitant moins de calibration liée à un nouveau sujet est d'utiliser des techniques de transfert d'information en apprentissage ou des techniques d'apprentissage multi-tâche. Ces techniques visent à apprendre conjointement plusieurs fonctions de décisions et à partager l'ensemble des données disponibles. Dans le cadre des interfaces cerveau-machine, cela revient donc à de l'apprentissage multi-sujet. L'objectif dans cette démarche est d'essayer de compenser le faible nombre de données d'apprentissage disponibles pour un classifieur donné par le transfert d'informations issues des données disponibles pour les autres classifieurs. Cette idée a été implémentée avec succès dans plusieurs travaux de recherche. Par exemple, Devlaminck et al. [29] utilise ce principe pour apprendre des filtres spatiaux adaptés à des sujets pour qui très peu de données d'apprentissage sont disponibles. Dans ce contexte, grâce au transfert d'information, de nets gains de performances sont observés en reconnaissance de signaux EEG. Suivant le même principe d'apprentissage multi-tâche, Alamgir et al. [30] propose d'apprendre les classifieurs EEGs pour différents sujets en modélisant leur vecteur $\mathbf{w}$ comme étant la réalisation d'une variable aléatoire suivant une loi normale. Grâce à cette modélisation, le vecteur $\mathbf{w}$ moyen est utilisé pour un nouveau sujet, puis est ensuite mis à jour au fur et à mesure que des données deviennent disponibles. Toujours dans ce contexte multi-tâche, il est possible de sélectionner les capteurs communs à plusieurs sujets à l'aide d'une norme mixte adaptée [26]. Les expériences montrent que, dans un cadre d'apprentissage d'un classifieur pour la reconnaissance de signaux P300, un important gain de performance est obtenu pour les sujets performant faiblement sans transfert d'information.

3 Mesures de performances

Cette section présente quelques mesures de performance utilisées pour évaluer et comparer les qualités de différentes hypothèses ou algorithmes, en distinguant le cas de la classification et celui de la régression. Ces mesures de performance, qui doivent toujours être estimées à partir de données qui n'ont pas servi pour l'apprentissage (voir section 4 [4, 16]), évaluent la capacité de généralisation des hypothèses apprises. Le lecteur pourra se référer à Schlögl et al. [31] pour une présentation plus complète des mesures de performance.

3.1 Mesure de performance pour la classification

Les mesures de performance classiques en classification se fondent sur la matrice de confusion $\mathbf{C}$, donnant pour toute paire de classes (i, j) le nombre $C_{i,j}$ des exemples de la classe i qui ont été attribués à la classe j. Dans le cas d'une classification parfaite, la matrice $\mathbf{C}$ est diagonale.

Le taux de bonne reconnaissance (TBR) est la fraction des exemples bien classés (figurant sur la diagonale de la matrice de confusion), qui correspond à l'erreur $0-1$ (section 1.2) et estime l'erreur de Bayes du classifieur. Ce taux a été utilisé dans la plupart des compétitions BCI pour évaluer les performances des classifieurs en imagerie motrice [4, 16].

Le TBR n'est cependant pas une bonne mesure de performance dans le cas de classes déséquilibrées : en effet, si une classe représente 99% des exemples, le classifieur trivial classant tous les exemples dans cette classe a un TBR de 99%. Une mesure alternative, mieux appropriée, est le coefficient Kappa de Cohen [32] qui a été utilisé lors de compétitions BCI [4, 16], défini par

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

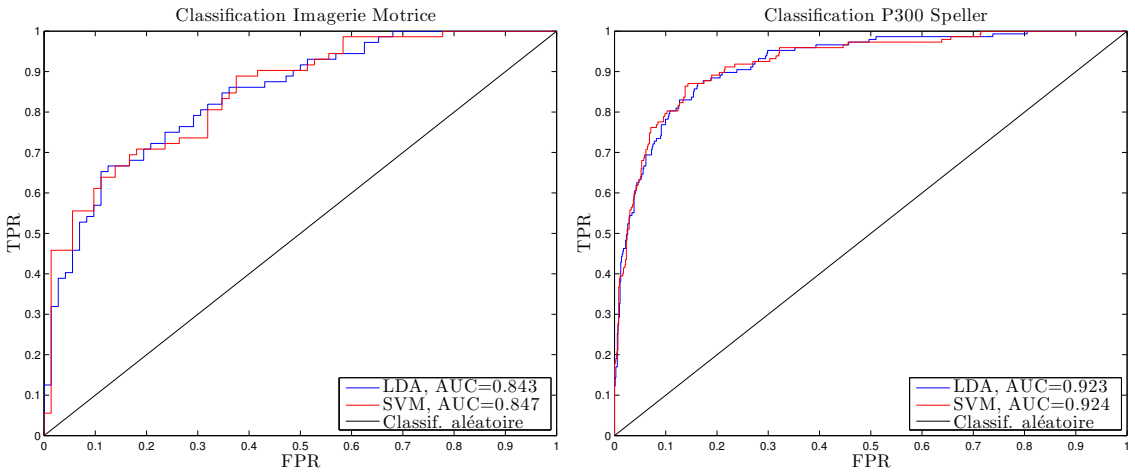

Figure 3: Courbes ROC pour une application Imagerie Motrice (gauche) et P300 Speller (droite). Mesure de performance AUC : Aire sous la courbe ROC. Axe des abscisses, taux de faux positif (FPR); axe des ordonnées, taux de vrais positifs (TPR).

où p_e est la probabilité d'une bonne classification aléatoire. L'information mutuelle est également une mesure de performance souvent utilisée en BCI [33]. Elle est fondée sur la théorie de l'information [4, 31] et permet de mesurer en bits la taux d'information transmis à la machine par l'interface.

Une autre mesure particulièrement adaptée au cas de classes déséquilibrées, est l'aire sous la courbe ROC (AUC), ou score de Mann-Whitney-Wilcoxon, donné par:

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

Concrètement, considérons dans le cas de la classification binaire une hypothèse $f(\cdot)$ à valeurs dans $\mathbb{R}$. Pour tout seuil $\tau \in \mathbb{R}$, on peut définir le classifieur f_τ classant $\mathbf{x}$ dans la classe positive ssi $f(\mathbf{x}) > \tau$, ainsi que le taux de vrais positifs TPR_τ ($P(f(\mathbf{x}) > \tau|y = 1)$) et le taux de faux positifs FPR_τ ($P(f(\mathbf{x}) < \tau|y = 1)$). La courbe monotone des points (FPR_τ, TPR_τ) est appelée courbe ROC (Receiver Operating Characteristics, Fig. 3). S'il existe un seuil tel que le classifieur $f(\cdot)_\tau$ soit parfait, la courbe ROC visite le point (0,1) (0% de faux negatifs et 100% de vrais positifs). Si au contraire, la courbe ROC coincide avec la diagonale, toute amélioration du point de vue du taux de vrais positifs est compensée par une dégradation égale du taux de faux négatifs : en d'autres termes, le classifieur n'apporte pas d'information. Plus généralement, l'aire sous la courbe ROC mesure la qualité de l'hypothèse $f(\cdot)$, indépendamment du déséquilibre des classes (dû au fait que les coordonnées FPR_τ et TPR_τ sont des pourcentages).

Le critère AUC a été utilisé dans la compétition BCI MLSP 2010 [34] ; dans le cadre des P300 speller en particulier, la classe positive compte beaucoup moins d'exemples que la classe négative par construction.

3.2 Performance pour la régression

D'autres mesures de performance doivent être considérées dans le cadre d'une tâche de régression, comme par exemple pour la prédiction de mouvement d'un membre [35]. Comme pour la classification, la mesure de performance est souvent liée au terme de coût des erreurs, minimisé lors de l'apprentissage. Pour la régression au sens des moindres carrés, la mesure de performance est donc l'erreur quadratique moyenne (MSE) ou coût ℓ_2 (Table 1). L'inconvénient de cette mesure est qu'elle n'est pas normalisée (elle dépend de l'amplitude des données y_i), ce qui rend difficile la comparaison de performance pour par exemple une prédiction de flexion de doigt [36] et un mouvement dans l'espace [35].

Une alternative est de prendre comme mesure de performance la corrélation entre la valeur prédite $f(\mathbf{x})$ et la valeur observée y. Cette mesure vaut 1 dans le cas d'une prédiction monotone ($f(\mathbf{x})$ croit avec y) ;

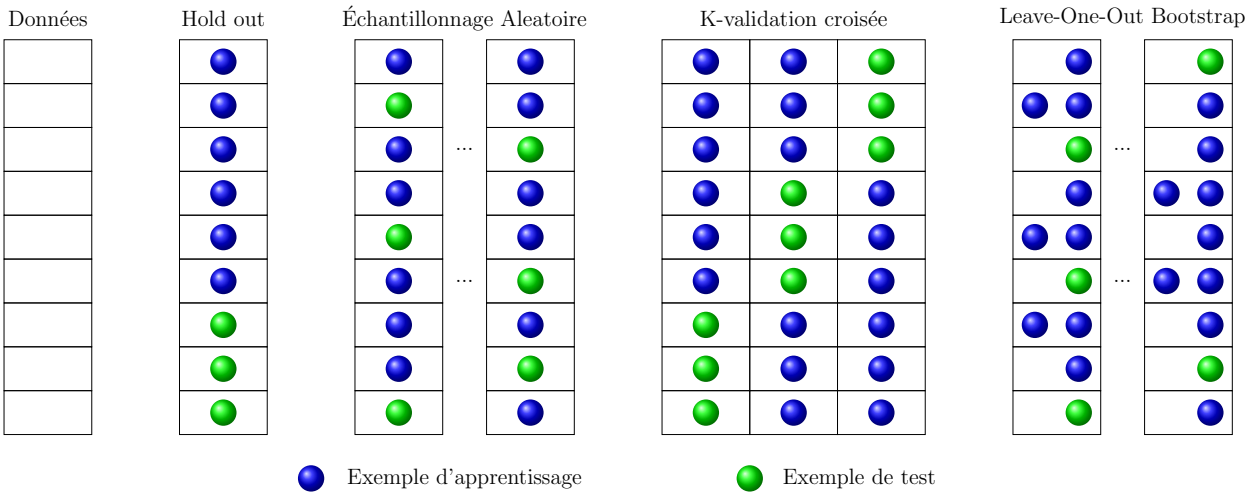

Figure 4: Illustration des différents découpages de données pour l'estimation de l'erreur de généralisation présentés dans le chapitre

une corrélation de 0 correspond à une hypothèse non informative. La corrélation ne prend toutefois pas en compte la valeur réelle de la prédiction, qui importe pour e.g. contrôler la position d'un curseur [37].

4 Validation et sélection de modèle

L'apprentissage statistique s'intéresse à deux aspects de la validation des résultats obtenus sur une application : l'évaluation de la mesure de performance choisie (section 3) et l'optimisation de la mesure de performance par ajustement des hyper-paramètres de l'algorithme. Le lecteur pourra se reporter à [3, 2, 38] pour une présentation approfondie des questions liées à la validation.

4.1 Estimation de la mesure de performance

Ainsi que dit précédemment, l'important n'est pas la performance d'un classifieur $f(\cdot)$ sur les données d'apprentissage : il est toujours possible en augmentant la complexité de $f(\cdot)$ d'obtenir un classifieur qui se comporte parfaitement sur les données connues. L'important est la capacité de généralisation de $f(\cdot)$, c'est à dire ses performances sur les données ultérieures. On cherche à évaluer les performances générales en extrayant des données disponibles une base d'apprentissage, qui sert à optimiser $f(\cdot)$, et une base de test, qui sert à estimer la performance de $f(\cdot)$ dans le cas général. L'hypothèse de travail sous-jacente est que l'ensemble des données disponibles est à la distribution complète, ce que la base d'apprentissage est à l'ensemble des données disponibles (Fig. 4).

Dans le cas où les données disponibles sont abondantes, l'estimation de la mesure de performance ne pose pas de problème difficile : on dispose de suffisamment de données pour apprendre un classifieur $f(\cdot)$, et on dispose de suffisamment de données (différentes) pour estimer la performance de $f(\cdot)$. Le problème se complique lorsque les données ne sont pas abondantes, ce qui est le cas dans le contexte BCI en raison du coût d'acquisition des données. Dans ce cas, il faut trouver un compromis entre réduire la base d'apprentissage (et donc diminuer la qualité de l'hypothèse apprise) et réduire la base de test (et donc diminuer la fiabilité de l'estimation de qualité).

En pratique, on distingue trois approches principales (Fig. 4), selon la façon dont les ensemble d'apprentissage et de test sont construits.

4.1.1 Échantillonnage aléatoire

La procédure dite de *hold-out* partitionne aléatoirement les n exemples entre base d'apprentissage $\mathcal{E}$ et base de test $\mathcal{T}$, apprend une hypothèse $f(\cdot)$ à partir de $\mathcal{E}$ et mesure la performance de l'hypothèse $f(\cdot)$ sur $\mathcal{T}$. Cette performance peut avoir une forte variance (si les exemples de $\mathcal{E}$ sont faciles et les exemples de $\mathcal{T}$ sont difficiles, ou inversement). Pour diminuer cette variance, la méthode dite *d'échantillonnage aléatoire* repète la procédure de hold-out en considérant K tirages indépendants des bases d'apprentissage et de test. Les performances obtenues sur les différentes bases de test sont ensuite moyennées. Les performances moyennes et l'intervalle de confiance associé permettent de comparer deux classifieurs et de déterminer si l'un est meilleur que l'autre avec une significativité donnée, selon un test d'hypothèse classique. Le fait de répéter K fois l'apprentissage sur des bases d'apprentissage indépendantes et de moyenner les performances évaluées sur les bases de test rend la mesure de performance moins sensible aux alea d'échantillonnage – au prix d'une augmentation du coût de calcul d'un facteur K.

4.1.2 K-validation croisée et leave-one-out (LOO)

La validation croisée à K-sous-ensembles (K-fold cross-validation) partitionne les n exemples en K sous-ensembles de taille égale. Chaque hold-out se fait alors en prenant l'un des sous-ensembles en base de test, et les données restantes en base d'apprentissage. Comme précedemment, la performance est estimée par la moyenne des performances sur les K sous-ensembles de test. Par rapport à l'échantillonnage aléatoire, la validation croisée réduit la variance de l'estimation (chaque exemple figure une et une seule fois dans l'ensemble de test). Lorsque l'on dispose de très peu d'exemples, on peut en particulier prendre K égal au nombre n d'exemples : la validation croisée coïncide alors avec la méthode de *leave-one-out* (LOO), où chaque apprentissage considère tous les exemples sauf un, et où on teste le classifieur appris sur l'exemple restant.

Le choix de K pose la question du compromis entre le biais et la variance : lorsque K augmente, le biais de l'estimation diminue (sous des hypothèses faibles [2]) mais la variance augmente, parce que les ensembles d'apprentissage sont très corrélés. La complexité augmente également avec K. Finalement, le choix de K dépend essentiellement du nombre n d'exemples. Si n est grand relativement au nombre d d'attributs de description, une valeur faible de K est considérée comme suffisante ($K = 2 \dots 10$), et la variance peut être réduite en répétant la validation croisée sur 5 partitions différentes des données [**?**].

En BCI la validation croisée est généralement utilisée avec de bonnes performances (voir par exemple Labbé et al. [19] pour l'estimation de la performance de reconnaissance de signaux P300).

4.1.3 Bootstrap

La particularité du *boostrap* est de construire la base d'apprentissage par n tirages uniformes *avec remise* parmi les n exemples disponibles. Certains exemples figurent ainsi plusieurs fois dans la base d'apprentissage, et d'autres sont absents ; en moyenne, la fraction des exemples non sélectionnés est de 37% des données disponibles. L'estimation de performance du classifieur se fait sur toutes les données disponibles ; comme ci-dessus, la performance est moyennée sur plusieurs tirages de la base d'apprentissage. L'estimation est moins optimiste que dans le cas du leave-one-out ; le biais est similaire à celui de la validation croisée avec $K = 2$ [2]. En dépit de l'existence de ce biais, le classement d'hypothèses selon leur performance estimée par bootstrap est généralement considéré comme fiable.

4.2 Optimisation des hyper-paramètres

Les algorithmes d'apprentissage font souvent intervenir des hyper-paramètres, tel le poids du terme de régularisation (Eq. 3, paramètre λ) ou les paramètres du noyau d'un SVM non-linéaire (écart d'un noyau gaussien, degré d'un noyau polynomial). Il est naturel de chercher la valeur des hyper-paramètres conduisant à une performance en généralisation optimale. La procédure recommandée est alors la suivante. On partitionne les données disponibles en trois ensembles : l'ensemble d'apprentissage $\mathcal{E}$, l'ensemble de test $\mathcal{T}$, et l'ensemble de validation $\mathcal{V}$.

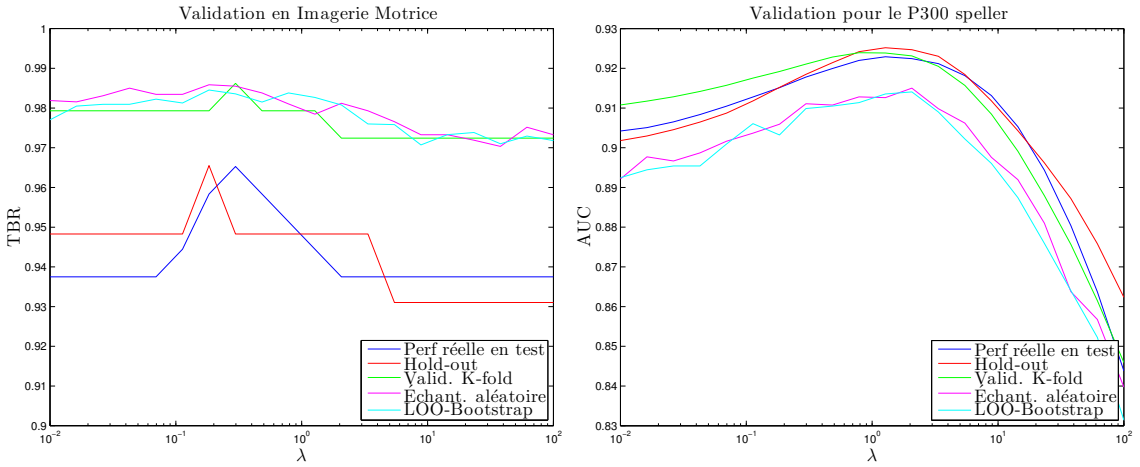

Figure 5: Illustration des différentes méthodes d'estimation de l'erreur de généralisation sur deux jeux de données de classification. Gauche : classification en imagerie motrice avec très peu d'exemples, mesure de performance = taux de bonne reconnaissance (TBR). Droite : détection de potentiel évoqué pour P300 speller, mesure de performacne = aire sous la courbe ROC (AUC).

A partir de $\mathcal{E}$ et d'un vecteur d'hyper-paramètre θ, on mesure la performance sur l'ensemble de test $\mathcal{T}$, notée $\mathcal{F}(\theta)$. La question devient alors de déterminer le vecteur d'hyper-paramètres optimal θ^* :

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

Après avoir obtenu θ^* (voir ci-dessous), il convient de recalculer la mesure de performance. En effet, puisque θ^* a été déterminé en utilisant l'information de $\mathcal{E}$ et $\mathcal{T}$, la mesure de performance $\mathcal{F}(\theta)$ calculée sur $\mathcal{T}$ est optimiste[2]. On garde ainsi la valeur θ^* pour apprendre sur $\mathcal{E} \bigcup \mathcal{T}$, et le classifieur obtenu est évalué sur l'ensemble de validation $\mathcal{V}$, qui n'avait jamais été considéré jusque la.

La détermination de θ^* peut procéder de trois manières. Si l'algorithme fait intervenir un petit nombre d'hyper-paramètres ($\theta \in \mathbb{R}^d$, avec $d = 2, 3$), la méthode usuelle est la recherche exhaustive sur une grille de l'espace des paramètres [39, 7]. Cette grille peut être régulière (e.g. le degré d'un noyau polynomial varie dans 1, 2, ... 10) ou non (e.g. on fait varier le poids de régularisation dans les puissances de 10, $\lambda = 10^i$, pour $i = -3 \ldots 3$). La recherche exhaustive de $\theta*$ est illustrée Fig. 5. L'avantage de cette méthode est sa simplicité ; en contre-partie, son coût augmente exponentiellement avec le nombre d'hyper-paramètres de l'algorithme.

Si la mesure de performance $\mathcal{F}$ (ou une approximation ou une borne de cette mesure) est dérivable, $\theta*$ peut être cherché en utilisant des méthodes de descente de gradient. Cette approche a été utilisée par exemple pour l'optimisation des hyper-parametres d'un SVM [40, 41] ou d'une analyse discriminante linéaire à deux classes [42]. Bien que complexe à mettre en oeuvre, cette méthode a l'avantage d'être moins coûteuse que la précédente lorsque le nombre de paramètres est grand.

Enfin, si $\mathcal{F}$ est connue uniquement sous forme de boîte noire (par un algorithme retournant la valeur de $\mathcal{F}(\theta)$ pour tout θ), on se retourne vers l'optimisation boîte noire stochastique, allant du recuit simulé aux stratégies d'évolution [43]. De manière générale, les algorithmes d'optimisation stochastique font évoluer une distribution sur l'espace de recherche, que l'on biaise graduellement vers les régions de meilleure performance. L'algorithme *Covariance-Matrix-Adaption* (CMA-ES)[3] procède de la sorte en adaptant les paramètres d'une distribution gaussienne, qui converge graduellement vers un optimum de la fonction objectif. Cette méthode est plus coûteuse que les précédentes : elle demande l'évaluation de $\mathcal{F}(\theta)$ pour un nombre assez grand de vecteurs d'hyper-paramètres θ ; or le calcul de $\mathcal{F}(\theta)$ correspond à apprendre et évaluer un classifieur.

[2]En d'autres termes, ce serait tricher que de considérer que $\mathcal{F}(\theta^*)$ représente la vraie performance du classifieur.
[3]https://www.lri.fr/ hansen/cmaes_inmatlab.html

Des méthodes d'optimisation hybride, couplant l'optimisation stochastique de $\mathcal{F}$ et l'apprentissage d'une approximation de $\mathcal{F}$, appelé modèle surrogé (*surrogate model-based optimisation*) sont donc considérées pour l'optimisation des hyper-paramètres d'un algorithme d'apprentissage en général [44, 45], et en particulier dans le cas BCI, pour la sélection des variables [46, 47, 48].

5 Conclusion

Ce chapitre donne une présentation synthétique des principales méthodes d'apprentissage supervisé utilisées dans le domaine des interfaces cerveau-machine, en mettant l'accent sur les difficultés pratiques de leur mise en œuvre. Certains travaux récents et prometteurs ont été suggérés, et le lecteur pourra avec profit les approfondir en fonction de son contexte particulier. Enfin, un aspect essentiel de l'apprentissage concerne la validation des résultats obtenus, visant d'une part le fonctionnement optimal d'un algorithme et d'autre part la comparaison rigoureuse des résultats de plusieurs algorithmes.

References

[1] Fabien Lotte, Marco Congedo, Anatole Lécuyer, Fabrice Lamarche, Bruno Arnaldi, et al. A review of classification algorithms for eeg-based brain–computer interfaces. *Journal of neural engineering*, 4, 2007.

[2] T. Hastie, J. Friedman, and R. Tibshirani. *The elements of statistical learning*. Springer Series in Statistics, 2001.

[3] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons,, 1999.

[4] Benjamin Blankertz, K Muller, Dean J Krusienski, Gerwin Schalk, Jonathan R Wolpaw, Alois Schlogl, Gert Pfurtscheller, Jd R Millan, M Schroder, and Niels Birbaumer. The bci competition iii: Validating alternative approaches to actual bci problems. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 14(2):153–159, 2006.

[5] Michael Tangermann, Klaus-Robert Müller, Ad Aertsen, Niels Birbaumer, Christoph Braun, Clemens Brunner, Robert Leeb, Carsten Mehring, Kai J Miller, Gernot R Müller-Putz, et al. Review of the bci competition iv. *Frontiers in neuroscience*, 6, 2012.

[6] Fabien Lotte and Cuntai Guan. Regularizing common spatial patterns to improve bci designs: unified theory and new algorithms. *Biomedical Engineering, IEEE Transactions on*, 58(2):355–362, 2011.

[7] A. Rakotomamonjy and V. Guigue. BCI competition III: Dataset II - ensemble of SVMs for BCI P300 speller. *IEEE Trans. Biomedical Engineering*, 55(3):1147–1154, 2008.

[8] V. Vapnik. *Statistical learning theory*. Wiley, 1998.

[9] Olivier Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.

[10] Ryota Tomioka, Kazuyuki Aihara, and Klaus-Robert Müller. Logistic regression for single trial eeg classification. *Advances in neural information processing systems*, 19:1377–1384, 2007.

[11] Norman Richard Draper and Harry Smith. Applied regression analysis 2nd ed. 1981.

[12] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[13] Yijun Wang, Zhiguang Zhang, Yong Li, Xiaorong Gao, Shangkai Gao, and Fusheng Yang. Bci competition 2003-data set iv: an algorithm based on cssd and fda for classifying single-trial eeg. *Biomedical Engineering, IEEE Transactions on*, 51(6):1081–1086, 2004.

[14] Dean J Krusienski, Eric W Sellers, Dennis J McFarland, Theresa M Vaughan, and Jonathan R Wolpaw. Toward enhanced p300 speller performance. *Journal of neuroscience methods*, 167(1):15–21, 2008.

[15] Vladimir Bostanov. 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. *Biomedical Engineering, IEEE Transactions on*, 51(6):1057–1061, 2004.

[16] Benjamin Blankertz, K Muller, Gabriel Curio, Theresa M Vaughan, Gerwin Schalk, Jonathan R Wolpaw, Alois Schlogl, Christa Neuper, Gert Pfurtscheller, Thilo Hinterberger, et al. The bci competition 2003: progress and perspectives in detection and discrimination of eeg single trials. *Biomedical Engineering, IEEE Transactions on*, 51(6):1044–1051, 2004.

[17] Alain Rakotomamonjy and Vincent Guigue. Bci competition iii: dataset ii-ensemble of svms for bci p300 speller. *Biomedical Engineering, IEEE Transactions on*, 55(3):1147–1154, 2008.

[18] Matthias Kaper, Peter Meinicke, Ulf Grossekathoefer, Thomas Lingner, and Helge Ritter. Bci competition 2003-data set iib: support vector machines for the p300 speller paradigm. *Biomedical Engineering, IEEE Transactions on*, 51(6):1073–1076, 2004.

[19] Benjamin Labbé, Xilan Tian, and Alain Rakotomamonjy. Mlsp competition, 2010: Description of third place method. In *Machine Learning for Signal Processing (MLSP), 2010 IEEE International Workshop on*, pages 116–117. IEEE, 2010.

[20] Alois Schlögl, Felix Lee, Horst Bischof, and Gert Pfurtscheller. Characterization of four-class motor imagery eeg data for the bci-competition 2005. *Journal of neural engineering*, 2(4):L14, 2005.

[21] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[22] Thomas Navin Lal, Michael Schroder, Thilo Hinterberger, Jason Weston, Martin Bogdan, Niels Birbaumer, and Bernhard Scholkopf. Support vector channel selection in bci. *Biomedical Engineering, IEEE Transactions on*, 51(6):1003–1010, 2004.

[23] Michael Schröder, Thomas Navin Lal, Thilo Hinterberger, Martin Bogdan, N Jeremy Hill, Niels Birbaumer, Wolfgang Rosenstiel, and Bernhard Schölkopf. Robust eeg channel selection across subjects for brain-computer interfaces. *EURASIP Journal on Applied Signal Processing*, 2005:3103–3112, 2005.

[24] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, 2012.

[25] R. Tomioka and K.R. MÃ$\frac{1}{4}$ller. A regularized discriminative framework for EEG analysis with application to brain-computer interface. *NeuroImage*, 49(1):415–432, 2010.

[26] R. Flamary, N. Jrad, R. Phlypo, M. Congedo, and A. Rakotomamonjy. Mixed-norm regularization for brain decoding. *Computational and Mathematical Methods in Medicine*, 2014(1):1–13, 2014.

[27] Daniel Strohmeier, Jens Haueisen, and Alexandre Gramfort. Improved meg/eeg source localization with reweighted mixed-norms. In *Pattern Recognition in Neuroimaging, 2014 International Workshop on*, pages 1–4, Tubingen, Allemagne, Jul 2014. IEEE.

[28] Nisrine Jrad, Marco Congedo, Ronald Phlypo, Sandra Rousseau, Rémi Flamary, Florian Yger, and Alain Rakotomamonjy. sw-svm: sensor weighting support vector machines for eeg-based brain–computer interfaces. *Journal of neural engineering*, 8(5):056004, 2011.

[29] Dieter Devlaminck, Bart Wyns, Moritz Grosse-Wentrup, Georges Otte, and Patrick Santens. Multisubject learning for common spatial patterns in motor-imagery bci. *Computational intelligence and neuroscience*, 2011:8, 2011.

[30] M. Alamgir, M. Grosse-Wentrup, and Y. Altun. Multitask learning for brain-computer interfaces. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 17–24, 2010.

[31] Alois Schlogl, Julien Kronegg, Jane E Huggins, and Steve G Mason. Evaluation criteria for bci research. In *Toward brain-computer interfacing*. 2007.

[32] Jean Carletta. Assessing agreement on classification tasks: the kappa statistic. *Computational linguistics*, 22(2):249–254, 1996.

[33] Tommi Nykopp. Statistical modelling issues for the adaptive brain interface. *Master's thesis, Helsinki University of Technology*, 2001.

[34] Kenneth E Hild, Mikko Kurimo, and Vince D Calhoun. The sixth annual mlsp competition, 2010. In *Machine Learning for Signal Processing (MLSP), 2010 IEEE International Workshop on*, pages 107–111. IEEE, 2010.

[35] T. Pistohl, T. Ball, A. Schulze-Bonhage, A. Aertsen, and C. Mehring. Prediction of arm movement trajectories from ecog-recordings in humans. *Journal of Neuroscience Methods*, 167(1):105–114, January 2008.

[36] Nanying Liang and Laurent Bougrain. Decoding finger flexion from band-specific ecog signals in humans. *Frontiers in neuroscience*, 6, 2012.

[37] Wei Wu, Yun Gao, Elie Bienenstock, John P Donoghue, and Michael J Black. Bayesian population decoding of motor cortical activity using a kalman filter. *Neural computation*, 18(1):80–118, 2006.

[38] Nathalie Japkowicz and Mohak Shah. *Evaluating Learning Algorithms A Classification Perspective*. Cambridge University Press, 2014.

[39] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[40] Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing multiple parameters for support vector machines. *Machine learning*, 46(1-3):131–159, 2002.

[41] R. Flamary, A. Rakotomamonjy, and G. Gasso. Learning constrained task similarities in graph-regularized multi-task learning. In Suykens J. A.K. , Signoretto M., Argyriou A., editor, *Regularization, Optimization, Kernels, and Support Vector Machines*. 2014.

[42] Sathiya Keerthi, Vikas Sindhwani, and Olivier Chapelle. An efficient method for gradient-based adaptation of hyperparameters in svm models. 2007.

[43] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[44] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

[45] Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michèle Sebag. Collaborative hyperparameter tuning. In *Int. Conf. on Machine Learning (ICML)*, volume 28 of *JMLR Proceedings*, pages 199–207. JMLR.org, 2013.

[46] Deon Garrett, David A Peterson, Charles W Anderson, and Michael H Thaut. Comparison of linear, nonlinear, and feature selection methods for eeg signal classification. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 11(2):141–144, 2003.

[47] Michael Schroder, Martin Bogdan, T Hinterberger, and N Birbaumer. Automated eeg feature selection for brain computer interfaces. In *Neural Engineering, 2003. Conference Proceedings. First International IEEE EMBS Conference on*, pages 626–629. IEEE, 2003.

[48] Rebeca Corralejo, Roberto Hornero, and Daniel Alvarez. Feature selection using a genetic algorithm in a motor imagery-based brain computer interface. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pages 7703–7706. IEEE, 2011.