Classification de Signaux EEG:

une Comparaison entre l'Algorithme SVM et les Réseaux Diffusifs

Jean-Pascal Pfister
Section de Phyisque
Travail de Diplôme dans le
Laboratoire de Calcul Neuromimétique, EPFL
Supervisé par Prof. W. Gerstner

Février 2002

Je puis bien concevoir un homme sans mains, pieds, tête, car ce n'est que l'expérience qui nous apprend que la tête est plus nécessaire que les pieds. Mais je ne puis concevoir l'homme sans pensée. Ce serait une pierre ou une brute.

Blaise Pascal 1623-1662

Résumé

Le but de ce projet est de classifier automatiquement des électro-encéphallogrammes (EEG). La base de données est composée de quatre classes mettant en évidence le transfert hémisphérique. Les données sont des essais uniques et non des moyennes sur des essais, ce qui donne un bruit élevé. Il s'agira de comparer une classification statique produite par l'algorithme *Support Vector Machine* (SVM) et une classification tenant compte de la dynamique obtenue grâce aux réseaux de neurones diffusifs. Cette technique permet d'approximer une distribution de probabilités de chemins continus. Nous utiliserons la méthode de Monte-Carlo ainsi que l'algorithme *Expectation Maximisation* (EM) pour entraîner le réseau. Contrairement à nos attentes, les SVM classifient bien mieux que les réseaux diffusifs.

Table des matières

Chapitre 1

Introduction

Au cours des vingt dernières années, les techniques d'imagerie fonctionnelle du cerveau ont connu des progrès significatifs. En particulier l'imagerie fonctionnelle par résonnance magnétique (fMRI) et recemment la tomographie par emission de positron (PET) ont ouvert de nouvelles possibilités dans l'étude des fonctions cognitives du cerveau.

Ces techniques permettent des résolutions spatiales de l'ordre du demi millimètre, ce qui est remarquable pour des methodes non-invasives. Pourtant, ces techniques d'imagerie souffrent d'un grand défaut: la résolution temporelle est de l'ordre de la seconde, alors que les transferts neuronaux se produisent à des temps de l'ordre de la milliseconde. C'est donc pour son excellente résolution temporelle que l'électro-encéphallogramme (EEG) est encore à l'ordre du jour.

Pour enregistrer des signaux EEG, il suffit de disposer un nombre donné d'électrodes sur le crâne des sujets. Celles-ci perçoivent à chaque instant l'activité électrique à la surface du crâne résultant d'une distribution de charge donnée. Par conséquent, ceci rend difficile la localisation précise de la source de potentiel.

L'intention de ce projet est de classifier des signaux EEG générés par des potentiels évoqués en réponse à un stimulus donné (ERP). Les sujets répondent par la main gauche ou droite à un stimulus visuel gauche ou droit, ce qui nous donne un problème à quatre classes.

Nous appellerons *carte*, l'ensemble des potentiels à un temps donné. La figure **??** en donne une représentation. Les points rouges représentent un potentiel positifs et les bleus un potentiel négatif. Lors d'une expérience, le sujet provoque une suite de cartes différentes que nous nommerons *essais*.

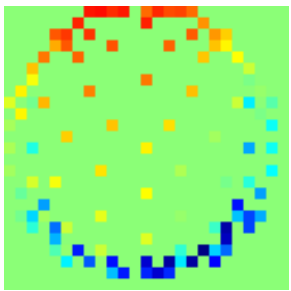

FIG. 1.1 – *Exemple de carte*

Il s'agira donc de reconnaître la classe à laquelle appartient une tâche donnée. Nous remarquerons qu'un tel travail similaire a déjà été effectué à l'aide de l'algorithme SVM [**?**], mais celui-ci concernait la classification de tâches mentales moyennées sur différents essais.

Dans ce projet, nous voulons classifier chaque essai individuellement, ce qui est nettement plus difficile en raison de la quantité de bruit dans les données. L'amplitude du signal et du bruit sont estimés respectivement à $10\mu V$ et $200\mu V$, ce qui donne un rapport signal sur bruit de l'ordre de 0.05!

Il est encore à mentionner que la classification de signaux EEG est également utilisée dans un contexte différent de celui des potentiels évoqués. Dans son article, Millan [**?**] montre qu'il est possible de classifier des EEG résultants de tâches mentales conscientes, ce qui permet entre autres de contruire une nouvelle interface entre l'homme et la machine.

Chapitre 2

Réseaux diffusifs

2.1 Motivations

Le principal intérêt de ce travail est d'explorer une nouvelle technique de classification des EEG. Celle-ci prend en compte la structure spatio-temporelle des données EEG contrairement à l'algorithme SVM qui considère les données de façon statique. Nous voulons donc voir si ces réseaux de neurones diffusifs sont applicables et efficaces pour la catégorisation de signaux EEG.

Toute cette approche concernant les réseaux diffusifs est entièrement inspirée de l'article de Movellan [?]. Ces réseaux permettent d'approximer une distribution de probabilité de chemins continus dans $\mathbb{R}^N$ afin de déterminer la probabilité qu'un chemin donné soit généré par cette distribution. Pour ce faire, il faut déterminer les paramètres caractérisant la dynamique des données. Nous formulons ainsi l'hypothèse que la dynamique des réseaux diffusifs est capable de reproduire celle des signaux EEG ou, plus précisément, nous espérons élaborer une distribution de probabilité de chemins qui reproduise celle des EEG.

En tenant compte de l'aspect dynamique de cette modélisation, nous espérons pouvoir surmonter le problème de la dilatation temporelle. En effet, deux personnes étant soumises aux mêmes conditions peuvent réagir avec une vitesse différente, ce qui a pour effet la dilatation d'un signal par rapport à un autre.

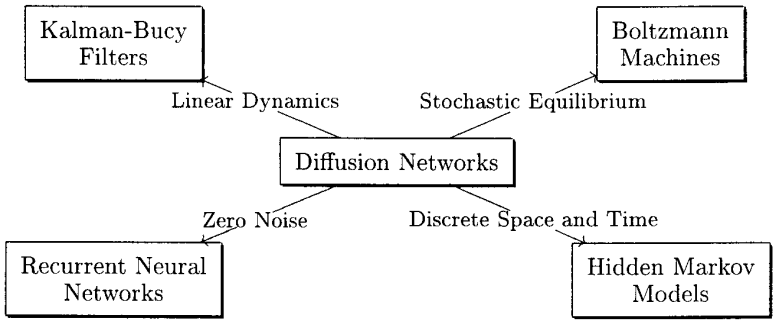

FIG. 2.1 – *Relation entre les réseaux diffusifs et les autres modèles (cette figure est tirée de [?])*

Sans entrer dans les détails, nous pouvons dire que les réseaux diffusifs sont respectivement une généralisation des filtres de Kalman, de la machine de Boltzman, des chaînes de Markov cachées et des réseaux de neurones récurrents, car ils ne sont limités ni à une dynamique linéaire, ni à l'équilibre stochastique, ni à la discrétisation, ni à une dynamique

déterministe (c.f figure **??**).

2.2 Réseaux de Hopfield

La dynamique des réseaux diffusifs est inspirée par un modèle de Hopfield [**?**], auquel est ajouté du bruit. L'architecture du réseau que nous allons considérer est donnée par la figure **??**. Celle-ci est constituée de N neurones entièrement interconnectés. On appelle *neurones observables*, les D neurones qui modélisent le signal désiré. Les $H = N - D$ autres neurones sont appelés *neurones cachés*.

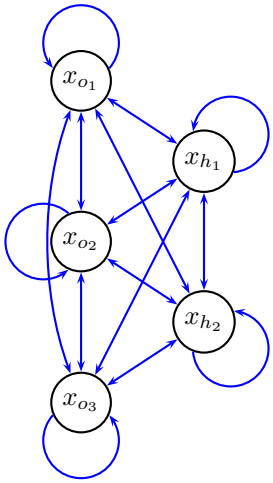

FIG. 2.2 – *Réseau entièrement connecté*

Pour que le modèle soit complet, il faut encore décrire le fonctionnement de chaque neurone individuellement. Celui-ci est inspiré par une modélisation électrique d'un neurone biologique (c.f. figure **??**).

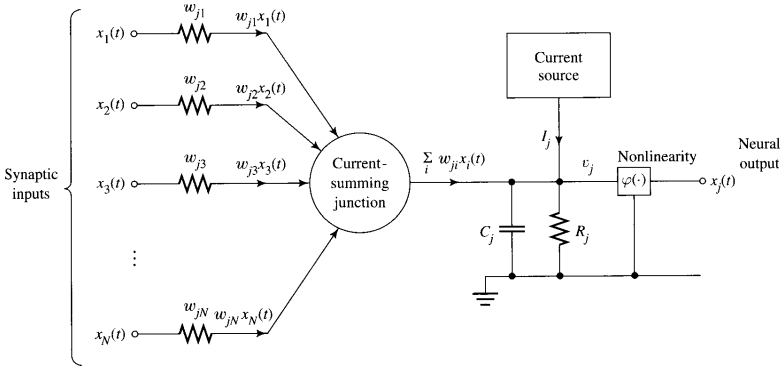

FIG. 2.3 – *Schéma électrique d'un neurone*

Soit x_i, le potentiel membranaire de la $i^{\text{ème}}$ dendrite, et $v_i = \varphi^{-1}(x_i)$, le champ local induit par le potentiel x_i. Les poids W_{ji} peuvent être vus comme les résistances électriques de la $i^{\text{ème}}$ entrée pour le $j^{\text{ème}}$ neurone. Soient R_j, C_j et I_j la résistance, la capacité et le courant d'entrée du $j^{\text{ème}}$ axone.

A l'aide de la loi de Kirshoff appliqué au schéma électrique représenté sur la figure **??**, nous obtenons l'expression suivante:

$$C_j \frac{\mathrm{d}v_j(t)}{\mathrm{d}t} + \frac{v_j(t)}{R_j} = \sum_{i=1}^{N} W_{ji}\varphi(v_i(t)) + I_j \tag{2.1}$$

Pour simplifier quelque peu le modèle, nous pouvons poser $I_j = 0$, $C_j = C = 1\ \forall j$, ce qui nous permet d'exprimer le temps de mémoire par $\tau_j = R_j C$. Le modèle que nous allons utiliser concerne les champs locaux décrits par la variable v_j que nous remplacerons par la notation x_j. Ainsi, la dynamique du réseau de Hopfield est décrite par

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{\mu}(\boldsymbol{x}(t),\lambda) \tag{2.2}$$

où la dérive s'exprime par

$$\mu_j(\boldsymbol{x}(t),\lambda) = -\frac{x_j(t)}{\tau_j} + \sum_{i=1}^{N} \varphi(x_i(t))W_{ij} \quad \forall j = 1,\ldots,N \tag{2.3}$$

et $\lambda = \{W,\gamma\},\gamma_i = \frac{1}{\tau_i}$ représente l'ensemble des paramètres du modèle. Un exemple de fonction d'activation $\varphi(x)$ est donné par

$$\varphi(x) = -1 + \frac{2}{1+e^{-\theta x}} \tag{2.4}$$

Nous noterons $\boldsymbol{\mu}_o \equiv (\mu_1,\ldots,\mu_D)$ et $\boldsymbol{\mu}_h \equiv (\mu_{D+1},\ldots,\mu_N)$, respectivement les partie observable et cachée de la dérive. Ce modèle comporte $N(N+1)$ paramètres qu'il s'agira de déterminer et optimiser.

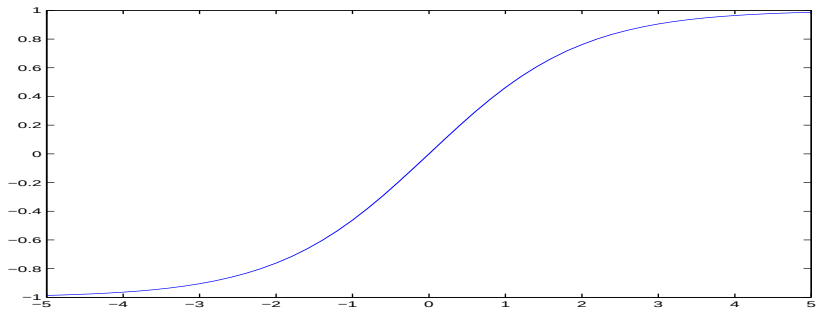

FIG. 2.4 – *Fonction d'activation*

Jusqu'à présent, nous avons décrit un modèle purement déterministe. A ce stade un réseau de neurones peut faire apparaître un point fixe ou des oscillations, mais il ne peut pas décrire un chemin continu quelconque. Pour posséder cette propriété, nous allons ajouter du bruit à la dynamique. Il est important de noter que le bruit n'est pas ajouté à la solution de l'équation (**??**), mais bien à la dynamique de chaque neurone[1]:

$$\frac{d\boldsymbol{X}^{\lambda}(t)}{dt} = \boldsymbol{\mu}(\boldsymbol{X}^{\lambda}(t),\lambda) + \sigma\frac{d\boldsymbol{B}(t)}{dt} \tag{2.5}$$

où σ est une constante et caractérise l'amplitude du bruit. Nous noterons que la formulation $d\boldsymbol{B}(t)/dt$ est un abus de notation puisque le mouvement brownien $\boldsymbol{B}(t)$, que nous

1. La notation $\boldsymbol{X}^{\lambda}(t)$ est choisie pour montrer qu'il ne s'agit pas d'une simple fonction du temps, mais bien d'un processus stochastique

définirons dans le prochain chapitre, est nullepart différentiable avec probabilité une. Avec une notation plus rigoureuse, nous pouvons écrire le processus de l'équation différentielle stochastique (??) par l'équation (??). Pour que la solution puisse être déterminée, il faut encore imposer une condition initiale que nous noterons ν.

$$\begin{aligned} \boldsymbol{X}^{\lambda}(t) &= \boldsymbol{X}^{\lambda}(0) + \int_0^t \boldsymbol{\mu}(\boldsymbol{X}^{\lambda}(s),\lambda)ds + \sigma \boldsymbol{B}(t) \\ \boldsymbol{X}^{\lambda}(0) &\sim \nu \end{aligned} \tag{2.6}$$

Nous noterons $\boldsymbol{O}^{\lambda}(t)$ et $\boldsymbol{H}^{\lambda}(t)$, respectivement les parties observable et cachée du processus $\boldsymbol{X}^{\lambda}(t)$.

2.3 Préliminaires mathématiques

2.3.1 Définitions

Puisque nous avons parlé de bruit qu'il faut ajouter à la dynamique, il convient d'introduire un espace de probabilité approprié. Pour ce faire, il est nécessaire de définir l'espace des épreuves, l'espace des événements et la mesure de probabilité correspondant à notre problème. Pour plus de détails, se référer à l'article de Movellan [?].

Définition 1 *L'espace des épreuves Ω est défini par l'ensemble des fonctions continues à N dimensions sur un intervalle de temps $[0,T]$:*

$$\Omega = \Omega_o \times \Omega_h = \{(\boldsymbol{x}_o(t),\boldsymbol{x}_h(t))|[0,T] \ni t \rightarrow \boldsymbol{x}_o(t) \in \mathbb{R}^D, [0,T] \ni t \rightarrow \boldsymbol{x}_h(t) \in \mathbb{R}^H\} \tag{2.7}$$

Définition 2 *L'espace des événements $\mathcal{F} = \mathcal{F}_o \otimes \mathcal{F}_h$ est la plus petite σ-algèbre des sous-ensembles ouverts de Ω. Un sous-ensemble C de Ω est ouvert si $\forall x \in C$, il existe une boule ouverte $B(x,\delta)$ de rayon δ appartenant à Ω:*

$$B(x,\delta) = \{\omega \in \Omega | \max_{t \in [0,T]} |\boldsymbol{x}(t) - \boldsymbol{\omega}(t)| < \delta\} \tag{2.8}$$

Pour pouvoir définir correctement la notion de mesure de probabilité sur l'espace mesurable $(\Omega,\mathcal{F})$, il est nécessaire de définir premièrement la notion de *mouvement brownien* utilisée dans les équations (??) et (??).

Définition 3 *Un processus stochastique $B = \{B(t), t \in [0,\infty)\}$ est un mouvement brownien par rapport à la mesure P et centré en 0 si les quatre conditions suivantes sont satisfaites.*

1. $P(B(0) = 0) = 1$.
2. $\forall t \in [0,\infty), \Delta t > 0, B(t+\Delta t) - B(t)$ *est une variable aléatoire gaussienne par rapport à P de moyenne 0 et de variance Δt.*
3. $\forall t_k, 0 \leq t_0 < t_1 < \cdots < t_l < \infty$, *les incréments browniens $B(t_k) - B(t_{k-1}), \forall k = 1,\ldots,l$ sont des variables aléatoires indépendantes par rapport à P.*
4. $B(t)$ *est continu.*

Un mouvement brownien à N dimensions consiste en N mouvements browniens indépendants.

Définition 4 *La mesure de probabilité $Q^{\lambda} : \mathcal{F} \rightarrow [0,1]$ est la mesure de probabilité induite par le processus (??)*

$Q^\lambda(A)$ représente la probabilité que le réseau génère un chemin x dans l'ensemble $A \in \mathcal{F}$ étant donné le paramètre λ. De façon naturelle, nous pouvons aussi définir $Q_o^\lambda(A_o) = Q^\lambda(A_o \times \Omega_h)\ \forall A_o \in \mathcal{F}_o$ comme étant la mesure de probabilité de générer un chemin observable x_o dans l'ensemble A_o. De façon similaire $Q_h^\lambda(A_h) = Q^\lambda(\Omega_o \times A_h)\ \forall A_h \in \mathcal{F}_h$.

Par la suite, nous aurons besoin d'une mesure de référence.

Définition 5 *La mesure de probabilité R est la mesure de probabilité induite par un réseau diffusif sans dérive (i.e. $\mu = 0$ dans l'équation (??)). Formellement, nous avons*

$$R(A) = \int_{\mathbb{R}^N} R^u(A) d\nu(u), \forall A \in \mathcal{F} \tag{2.9}$$

où $\{R^u : u \in \mathbb{R}^N\}$ est l'ensemble des mesures de probabilité sur $(\Omega,\mathcal{F})$ induites par les mouvements browniens centrés en u.

Par la suite, nous noterons P, la mesure de probabilité induite par la base de donnée d'entraînement. Notre but sera de trouver λ tel que Q_o^λ approxime au mieux la mesure de probabilité P.

2.3.2 Théorème de Girsanov

Dans ce paragraphe, nous n'allons pas expliquer en détails toutes les bases nécessaires à la compréhension du théorème de Girsanov. Nous nous contenterons de mentionner le résultat qui nous intéresse et présenter un exemple simple qui doit donner une intuition quant à ce théorème. Pour plus d'informations concernant le calcul stoachastique, se référer à [?] et [?].

Soit Q^λ et R, les mesures de probabilités données par les définitions ?? et ??. Il est possible d'effectuer un changement de mesure et exprimer Q^λ en fonction de R:

$$Q^\lambda(A) = \int_{x \in A} L^\lambda(x) dR(x) \tag{2.10}$$

où $L^\lambda(x) = dQ^\lambda/dR(x)$ est appelé la dérivée de *Radon-Nikodym* de Q^λ par rapport à R. Dans ces conditions, le théorème de Girsanov affirme que la densité de probabilité $L^\lambda(x)$ est donnée par:

$$L^\lambda(x) = \exp\left\{\frac{1}{\sigma^2}\int_0^T \boldsymbol{\mu}(\boldsymbol{x}(t),\lambda) \cdot d\boldsymbol{x}(t) - \frac{1}{2\sigma^2}\int_0^T \|\boldsymbol{\mu}(\boldsymbol{x}(t),\lambda)\|^2 dt\right\} \tag{2.11}$$

Pour comprendre quelque peu d'où vient ce résultat, arrêtons-nous un instant sur une situation plus simple. Nous nous restreindrons au cas discret, ce qui nous permet de considérer une suite de variables aléatoires plutôt qu'un processus stochastique.

Exemple 1 (Karatzas & Shreve, 1991, p190) *Soient $Z_1,\ldots,Z_n$ des variables aléatoires de distribution normale sur $(\Omega,\mathcal{F},P)$ avec $\mathbb{E}(Z_i) = 0$ et $\mathbb{E}(Z_i^2) = 1$. Soit $(\mu_1,\ldots,\mu_n) \in \mathbb{R}^n$ un vecteur et $\tilde{P}$ une nouvelle mesure de probabilité sur $(\Omega,\mathcal{F})$ donnée par*

$$\tilde{P} = \exp\left[\sum_{i=1}^n \mu_i Z_i - \frac{1}{2}\sum_{i=1}^n \mu_i^2\right] P \tag{2.12}$$

Alors $\tilde{P}\,[Z_1 \in dz_1,\ldots,Z_n \in dz_n]$ est donné par

$$\begin{aligned} & \exp\left[\sum_{i=1}^{n}\mu_i Z_i - \frac{1}{2}\sum_{i=1}^{n}\mu_i^2\right] P\left[Z_1 \in dz_1,\ldots,Z_n \in dz_n\right] \\ = \quad & \exp\left[\sum_{i=1}^{n}\mu_i Z_i - \frac{1}{2}\sum_{i=1}^{n}\mu_i^2\right] (2\pi)^{-n/2}\exp\left[-\frac{1}{2}\sum_{i=1}^{n} z_i^2\right] dz_1\ldots dz_n \\ = \quad & (2\pi)^{-n/2}\exp\left[-\frac{1}{2}\sum_{i=1}^{n}(z_i-\mu_i)^2\right] dz_1\ldots dz_n \end{aligned} \tag{2.13}$$

Ainsi, par rapport à la mesure $\tilde{P}$, les variables aléatoires $Z_1,\ldots,Z_n$ sont indépendantes et normales avec $\tilde{\mathbb{E}}(Z_i)=\mu_i$ et $\tilde{\mathbb{E}}\left[(Z_i-\mu_i)^2\right]=1$. Autrement dit, si les variables aléatoires $\tilde{Z}_i = Z_i - \mu_i$ définies sur $(\Omega,\mathcal{F},\tilde{P})$ sont indépendantes et de distribution normale, alors la mesure de probabilité $\tilde{P}$ est définie par (??).

Le théorème de Girsanov étend ce résultat d'invariance de distribution gaussienne, sous un changement de mesure de probabilité donné, du cas discret au cas continu.

2.4 Monte-Carlo

Dans notre situation, ce qui nous intéresse réellement, c'est la vraisemblance que le réseau diffusif génère le chemin x_o dans la couche de neurones observables, par rapport à la mesure de référence R_o. Cette fonction de vraisemblance $L_o^\lambda(x_o)$ est donnée par

$$\begin{aligned} L_o^\lambda(x_o) &= \frac{dQ_o^\lambda}{dR_o}(x_o) \\ &= \int_{\Omega_h} L^\lambda(x_o,x_h)dR_h(x_h) \end{aligned} \tag{2.14}$$

Pour calculer cette intégrale, nous utiliserons la méthode de Monte-Carlo. Celle-ci est une méthode probabiliste, ce qui permet de limiter le nombre de calculs par rapport à une méthode directe. En d'autres termes, il s'agit de générer des chemins cachées x_h avec une distribution R_h pour pouvoir approximer l'intégrale (??) par une somme sur ces chemins.

Puisqu'il n'est pas évident a priori de générer des chemins par la distribution R_h, nous sommes libres d'effectuer un nouveau changement de mesure de probabilité. Soit S_h^{λ,x_o}, la mesure de probabilité obtenue en forçant le réseau de produire le chemin x_o dans les neurones observables et en enregistrant les chemins cachés. Formellement, cette dynamique est produite par le processus suivant:

$$\begin{aligned} d\boldsymbol{H}^\lambda(t) &= \boldsymbol{\mu}_h(\boldsymbol{x}_o(t),\boldsymbol{H}^\lambda(t),\lambda)dt + \sigma d\boldsymbol{B}_h(t) \\ \boldsymbol{H}^\lambda(0) &\sim \nu_h \end{aligned} \tag{2.15}$$

Puisque nous avons à nouveau une mesure générée par un processus de diffusion et une mesure de référence sans dérive, nous pouvons à nouveau utiliser le théorème de Girsanov:

$$\frac{dS_h^{\lambda,x_o}}{dR}(x_h) = \exp\left\{\frac{1}{\sigma^2}\int_0^T \boldsymbol{\mu}_h(\boldsymbol{x}(t),\lambda)\cdot d\boldsymbol{x}_h(t) - \frac{1}{2\sigma^2}\int_0^T \|\boldsymbol{\mu}_h(\boldsymbol{x}(t),\lambda)\|^2 dt\right\} \tag{2.16}$$

avec $\boldsymbol{x}(t)=(\boldsymbol{x}_o(t),\boldsymbol{x}_h(t))$. A présent, nous pouvons exprimer à nouveau la fonction de vraisemblance $L_o^\lambda(x_o)$ en utilisant ce dernier résutat:

$$\begin{aligned} L_o^\lambda(x_o) &= \int_{\Omega_h} L^\lambda(x_o,x_h)\frac{dR_h}{dS_h^{\lambda,x_o}}(x_h)dS_h^{\lambda,x_o}(x_h) \\ &= \int_{\Omega_h} \exp\Big\{\frac{1}{\sigma^2}\int_0^T \boldsymbol{\mu}_o(\boldsymbol{x}(t),\lambda)\cdot d\boldsymbol{x}_o(t) \\ &\qquad - \frac{1}{2\sigma^2}\int_0^T \|\boldsymbol{\mu}_o(\boldsymbol{x}(t),\lambda)\|^2 dt\Big\} dS_h^{\lambda,x_o}(x_h) \end{aligned} \tag{2.17}$$

Pour approximer la distribution S_h^{λ,x_o}, il suffit de générer M chemins $\mathcal{H} = \{h^m\}_{m=1}^M$ au moyen du processus (**??**). L'approximation de l'intégrale (**??**) devient

$$\hat{L}_o^\lambda(x_o) = \sum_{m=1}^M p^\lambda(x_o,h^m) \tag{2.18}$$

où $p^\lambda(x_o,h^m)$ est défini pour tout $h^m \in \mathcal{H}$ de la façon suivante:

$$\begin{aligned} p^\lambda(x_o,h^m) &= \tfrac{1}{M} L^\lambda(x_o,h^m)\tfrac{dR_h}{dS_h^{\lambda,x_o}}(h^m) \\ &= \tfrac{1}{M}\exp\Big\{\tfrac{1}{\sigma^2}\int_0^T \boldsymbol{\mu}_o(\boldsymbol{x}^m(t),\lambda)\cdot d\boldsymbol{x}_o(t) - \tfrac{1}{2\sigma^2}\int_0^T \|\boldsymbol{\mu}_o(\boldsymbol{x}^m(t),\lambda)\|^2 dt\Big\} \end{aligned} \tag{2.19}$$

avec $\boldsymbol{x}^m(t) = (\boldsymbol{x}_o(t),\boldsymbol{h}^m(t))$. La fonction $p^\lambda(x_o,h^m)$ peut être vue, à une normalisation près, comme la probabilité que le $m^{\text{ième}}$ chemin caché soit généré au moyen du processus (**??**) étant donné le chemin observable x_o et l'ensemble des paramètres λ.

2.5 Optimisation

2.5.1 Fonction de vraisemblance

Jusqu'à présent nous avons décrit le fonctionnement des réseaux diffusifs, mais nous n'avons pas précisé comment adapter l'ensemble des paramètres pour que le réseau soit optimal. Il s'agit de définir une fonction qu'il faudra maximiser pour approximer au mieux les chemins d'une classe donnée [2].

Soit $\mathcal{E} = \{x_o^s\}_{s=1}^S$, l'ensemble d'entraînement d'une classe donnée et $\widetilde{\mathcal{H}} = \{h^{m,s}\}_{m,s=1}^{M,S}$, l'ensemble des chemins générés par toutes les distributions de $\{S_h^{\lambda,x_o^s}\}_{s=1}^S$. Nous pouvons ainsi définir la fonction de vraisemblance $\phi(\lambda)$ du paramètre λ. Celle-ci représente, à une normalisation près, la probabilité de générer l'ensemble des chemins de $\mathcal{E}$:

$$\phi(\lambda) = \frac{1}{S}\sum_{s=1}^S \log \hat{L}_o^\lambda(x_o^s) \tag{2.20}$$

Pour maximiser $\phi(\lambda)$, par rapport à λ, il suffit de poser:

$$\nabla_\lambda \phi(\lambda) = 0 \tag{2.21}$$

Avant de calculer effectivement le gradient de $\phi(\lambda)$ par rapport à λ, établissons premièrement la dérivée de $\log L^\lambda(x)$ par rapport aux poids W_{ij} et par rapport aux $\gamma_i = \frac{1}{\tau_i}$

2. Puisqu'il faudra élaborer un réseau pour chaque classe, et que la stucture des réseaux est identique, nous omettrons l'indice c mentionnant la classe en question.

$$\begin{aligned}\frac{\partial \log L^{\lambda}(x)}{\partial W_{ij}} &= \frac{1}{\sigma^2}\int_0^T \frac{\partial \boldsymbol{\mu}(\boldsymbol{x}(t),\lambda)}{\partial W_{ij}}\left(d\boldsymbol{x}(t) - \boldsymbol{\mu}(\boldsymbol{x}(t),\lambda)dt\right) \\ &= \frac{1}{\sigma^2}\left(b_{ji}(x) - \sum_{n=1}^{N} W_{jn}a_{in}(x)\right) \end{aligned} \tag{2.22}$$

avec

$$b_{ij}(x) = \int_0^T \varphi(x_i(t))dx_j(t) + \frac{1}{\tau_j}\int_0^T \varphi(x_i(t))x_j(t)dt \tag{2.23}$$

$$a_{ij}(x) = \int_0^T \varphi(x_i(t))\varphi(x_j(t))dt \tag{2.24}$$

et

$$\begin{aligned}\frac{\partial \log L^{\lambda}(x)}{\partial \gamma_i} &= \frac{1}{\sigma^2}\int_0^T \frac{\partial \boldsymbol{\mu}(\boldsymbol{x}(t),\lambda)}{\partial \gamma_i}\left(d\boldsymbol{x}(t) - \boldsymbol{\mu}(\boldsymbol{x}(t),\lambda)dt\right) \\ &= \frac{1}{\sigma^2}\left(-\gamma_i\int_0^T x_i(t)^2dt + \sum_{j=1}^{N}\int_0^T \varphi(x_j(t))W_{ji}x_i(t)dt \right. \\ &\qquad \left. -\int_0^T x_i(t)dx_i(t)\right) \end{aligned} \tag{2.25}$$

Pour pouvoir utiliser les résultats (**??**) et (**??**), il faut pouvoir exprimer $\nabla_\lambda \log \hat{L}_o^{\lambda}(x_o)$ en fonction de $\nabla_\lambda \log L^{\lambda}(x)$[3]:

$$\nabla_\lambda \log \hat{L}_o^{\lambda}(x_o^s) = \sum_{m=1}^{M} p_{h|o}^{\lambda}(h^{m,s}|x_o^s)\nabla_\lambda \log L^{\lambda}(x_o^s,h^{m,s}) \quad \forall h^{m,s} \in \widetilde{\mathcal{H}} \tag{2.26}$$

avec

$$p_{h|o}^{\lambda}(h^{m,s}|x_o^s) = \frac{p^{\lambda}(x_o^s,h^{m,s})}{L_o(x_o^s)} \tag{2.27}$$

Le gradient de la fonction de vraisemblance devient:

$$\nabla_\lambda \phi(\lambda) = \frac{1}{S}\sum_{s=1}^{S}\sum_{m=1}^{M} p_{h|o}^{\lambda}(h^{m,s}|x_o^s)\nabla_\lambda \log L^{\lambda}(x_o^s,h^{m,s}) \tag{2.28}$$

Avec ce résultat, nous pouvons exprimer la dérivée de $\phi(\lambda)$ par rapport au poids W. Soit $\lambda^* = (W^*,\tau)$ tel que

$$\tilde{b}^{\lambda^*} - \tilde{a}^{\lambda^*}W^* = 0 \tag{2.29}$$

où

3. Le détail de ce calcul figure dans l'annexe **??**.

$$\tilde{b}^{\lambda^*} = \sum_{s=1}^{S}\sum_{m=1}^{M} p_{h|o}^{\lambda^*}(h^{m,s}|x_o^s)b(x_o^s,h^{m,s}) \tag{2.30}$$

$$\tilde{a}^{\lambda^*} = \sum_{s=1}^{S}\sum_{m=1}^{M} p_{h|o}^{\lambda^*}(h^{m,s}|x_o^s)a(x_o^s,h^{m,s}) \tag{2.31}$$

Résoudre l'équation (**??**) est un problème non trivial à résoudre, car il s'agit d'une équation non-linéaire en W^*. Par ailleurs, même si la fonction d'activation $\varphi(x)$ était linéaire, cette équation resterait non-linéaire.

2.5.2 Algorithme EM

Pour résoudre l'équation (**??**), nous utiliserons l'algorithme *Expectation-Maximisation* (EM). Celui-ci permet de trouver de façon itérative les paramètres maximisant une fonction de vraisemblance dépendant de données observables et cachées. Dans notre situation, nous voulons trouver le paramètre λ qui maximise la fonction de vraisemblance $\phi(\lambda)$.

L'algorithme EM se déroule en deux étapes. Dans la première phase (E-step), il calcule l'espérance de $\Phi(\lambda) = \frac{1}{S}\sum_{s=1}^{S}\log L^{\lambda}(x_o^s,x_h)$ étant donné le paramètre $\lambda^{(k-1)}$ de l'itération précédente et les données observables $\mathcal{E} = \{x_o^s\}_{s=1}^{S}$ où $\Phi(\lambda)$ est la fonction de vraisemblance des données complètes:

$$\begin{aligned} G(\lambda,\lambda^{(k-1)}) &= \frac{1}{S}\sum_{s=1}^{S}\mathbb{E}\left(\log L^{\lambda}(x_o^s,x_h)|\mathcal{E},\lambda^{(k-1)}\right) \\ &= \frac{1}{S}\sum_{s=1}^{S}\int_{\Omega_h}\log L^{\lambda}(x_o^s,x_h)\frac{dQ_{h|o}^{\lambda^{(k-1)}}}{dS_h^{\lambda^{(k-1)},x_o^s}}(x_h|x_o^s)dS_h^{\lambda^{(k-1)},x_o^s}(x_h) \\ &\simeq \frac{1}{S}\sum_{s=1}^{S}\sum_{m=1}^{M} p_{h|o}^{\lambda^{(k-1)}}(h^{m,s}|x_o^s)\log L^{\lambda}(x_o^s,h^{m,s}) \end{aligned} \tag{2.32}$$

Pour obtenir ce résultat, nous avons utilisé la relation suivante qui est détaillée dans l'annexe **??**:

$$\frac{dQ_{h|o}^{\lambda}}{dS_h^{\lambda,x_o}}(h|x_o) = p_{h|o}^{\lambda}(h|x_o) \tag{2.33}$$

Nous noterons $\hat{G}(\lambda,\lambda^{(k-1)})$, l'approximation de $G(\lambda,\lambda^{(k-1)})$ donnée par l'équation (**??**). Dans la deuxième phase (M-step), il cherche les paramètres qui maximisent l'espérance de l'étape précédente.

$$\lambda^{(k)} = \arg\max_{\lambda}\hat{G}(\lambda,\lambda^{(k-1)}) \tag{2.34}$$

Pour trouver λ, il suffit de dériver $\hat{G}(\lambda,\lambda^{(k-1)}$ par rapport à λ

$$\nabla_{\lambda}\hat{G}(\lambda,\lambda^{(k-1)}) = \frac{1}{S}\sum_{s=1}^{S}\sum_{m=1}^{M} p_{h|o}^{\lambda^{(k-1)}}(h^{m,s}|x_o^s)\nabla_{\lambda}\log L^{\lambda}(x_o^s,h^{m,s}) \tag{2.35}$$

Plus précisément, pour trouver les poids W, nous pouvons utiliser l'équation (**??**) pour obtenir:

$$\nabla_W \hat{G}(\lambda,\lambda^{(k-1)}) = \frac{1}{\sigma^2}\left(\tilde{b}^{\lambda^{(k-1)}} - \tilde{a}^{\lambda^{(k-1)}} W\right) = 0 \tag{2.36}$$

et ainsi exprimer $W^{(k)}$ en fonction de $W^{(k-1)}$

$$W^{(k)} = \left(\tilde{a}^{\lambda^{(k-1)}}\right)^{-1} \tilde{b}^{\lambda^{(k-1)}} \tag{2.37}$$

Pour ce qui est des constantes de temps, l'utilisation de l'équation (**??**) permet d'obtenir:

$$\begin{aligned}
\nabla_{\gamma_i} \hat{G}(\lambda,\lambda^{(k-1)}) &= \frac{1}{S}\sum_{s=1}^{S}\sum_{m=1}^{M} p_{h|o}^{\lambda^{(k-1)}}(h^{m,s}|x_o^s)\nabla_{\gamma_i} \log L^{\lambda}(x_o^s,h^{m,s}) \\
&= \frac{1}{S}\sum_{s=1}^{S}\sum_{m=1}^{M} p_{h|o}^{\lambda}(h^{m,s}|x_o^s)\frac{1}{\sigma^2}\left(-\gamma_i \int_0^T x_i^{m,s}(t)^2 dt\right. \\
&\quad \left. + \sum_{j=1}^{N}\int_0^T \varphi(x_j^{m,s}(t)) W_{ji} x_i^{m,s}(t) dt - \int_0^T x_i^{m,s}(t) dx_i^{m,s}(t)\right) \\
&= 0
\end{aligned} \tag{2.38}$$

et donc

$$\gamma_i^{(k)} = \frac{\sum_{s,m=1}^{S,M} p_{h|o}^{\lambda^{(k-1)}}(h^{m,s}|x_o^s)\left(\sum_{j=1}^{N}\int_0^T \varphi(x_j^{m,s}(t)) W_{ji}^{(k-1)} x_i^{m,s}(t) dt - \int_0^T x_i^{m,s}(t) dx_i^{m,s}(t)\right)}{\sum_{s=1}^{S}\sum_{m=1}^{M} p_{h|o}^{\lambda^{(k-1)}}(h^{m,s}|x_o^s)\int_0^T x_i(t)^2 dt}$$

où $\boldsymbol{x}^{m,s}(t) = (\boldsymbol{x}^s(t),\boldsymbol{h}^{m,s}(t))$. Pour obtenir le paramètre λ^* optimal, il faut itérer successivement les étapes 1 et 2 de l'algorithme jusqu'à ce que λ converge vers λ^*.

2.5.3 Recuit simulé

Dans le paragraphe précédent, nous avons montré comment procéder pour déterminer le paramètre λ de façon à maximiser la probabilité que le réseau génère la base d'entraînement $\mathcal{E}$. Il reste encore le paramètre σ à fixer: celui qui contrôle l'amplitude du bruit.

Si le bruit est trop fort, le réseau peut générer des courbes fortement différentes de celles que nous voudrions approximer. Si, au contraire, il est trop faible, il se peut que les réseau ne parvienne pas à effectuer des écarts suffisamment grands pour pouvoir suivre la dynamique. Il s'agit donc de trouver un σ optimal: ni trop élevé, ni trop faible.

La méthode du *recuit simulé* [**?**] propose une technique pour trouver le maximum global d'une fonction d'énergie au moyen d'un paramètre de contrôle qui est la température. Dans notre cas, nous pouvons faire l'analogie entre la fonction $\phi(\lambda)$ et la fonction d'énergie, ce que fait de l'amplitude du bruit σ le correspondant de la température.

L'idée est la suivante: pour trouver le maximum global dans un espace de paramètres, il faut pouvoir effectuer premièrement de grands déplacements pour déterminer la meilleure région, puis restreindre l'étendue de la recherche pour trouver un maximum local. Le paramètre permettant de contrôler l'étendue de la recherche est la température. Il faudra donc se donner un programme de "refroidissement" qui sera, dans notre cas, une suite

de σ qui diminue en fonction du nombres d'itérations. Par la suite, nous utiliserons une simple suite géométrique donnée par:

$$\sigma^{(k)} = \alpha\sigma^{(k-1)}, \quad \alpha \leq 1 \tag{2.39}$$

A chaque itération de l'algorithme EM, nous trouvons un nouvel ensemble de paramètres $\lambda^{(k)}$ qui dépend de $\lambda^{(k-1)}$ et $\sigma^{(k-1)}$. D'autres variantes sont possibles: il est également envisageable d'itérer un certain nombre de fois l'algorithme EM avant de diminuer le niveau du bruit d'un facteur α.

L'algorithme, décrit dans l'annexe **??**, résume toute cette section concernant l'entraînement des poids et des constantes de temps.

2.6 Classification

Puisque nous pouvons élaborer un réseau capable déterminer la vraisemblance d'un chemin, il suffit de construire quatre réseaux pour que chaque classe possède sa fonction de vraisemblance. Ainsi, pour classifier un chemin donné, il suffit de lui attribuer la classe pour laquelle la vraisemblance est maximale. En d'autres termes, si λ_c^* représente l'ensemble des paramètres optimisés pour la classe c, le nouveau chemin x_o^{new} appartiendra à la classe

$$c^{\text{new}} = \arg \max_{c=1,\ldots,4} \log L_o^{\lambda_c^*}(x_o^{\text{new}}) \tag{2.40}$$

Chapitre 3

Support Vector Machine

Puisque l'algorithme SVM est connu depuis une dizaine d'années [**?**], nous n'allons pas présenter la théorie complète, mais seulement une idée générale.

Soit $\mathcal{D} = \{(\mathbf{x}^s,d^s)\}_{s=1}^{S}$ une base de données labellée en deux classes ($d^s = \pm 1$). Puisque l'algorithme SVM est un algorithme de classification statique, il faut comprendre les vecteurs $\mathbf{x}^s \in \mathbb{R}^{TE}$, comme l'ensemble des potentiels pour un essai donné. Plus précisément, $x^s_{(t-1)E+e}$ représente le potentiel au temps t de l'électrode e de l'essai s, tandis que E correspond au nombre d'électrodes et T au nombre de pas de temps.

L'idée des SVM est de trouver un hyperplan optimal de séparation (OSH) de la base de donnée $\mathcal{D}$ délimitant la zone de l'espace contenant la cible -1 de celle contenant la cible $+1$. Dans ce contexte, la notion d'optimalité est atteinte lorsque la distance entre l'OSH et le vecteur de support le plus proche est maximale. Cette situation a été représentée sur la figure **??** pour le cas à deux dimensions. Le cas général à plusieurs classes sera discuté à la fin du chapitre.

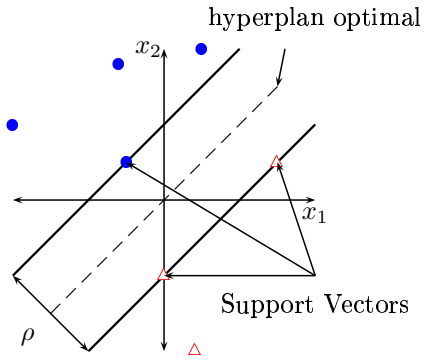

FIG. 3.1 – *Situation linéairement séparable*

Pour augmenter la probabilité de séparation linéaire entre les deux classes, nous pouvons plonger les données dans un espace de dimension supérieure au moyen de la fonction $\boldsymbol{\psi}(\cdot) : \mathbb{R}^{TE} \rightarrow \mathcal{H}$ (Cover, 1965). La base de donnée $\mathcal{D}$ est linéairement séparable dans le nouvel espace $\mathcal{H}$, s'il existe un *vecteur poids* $\mathbf{w} \in \mathbb{R}^N$ et une constante $b \in \mathbb{R}$ tels que

$$d^s(\mathbf{w}^T\boldsymbol{\psi}(\mathbf{x}_n) + b) \geq 1 \qquad \forall(\mathbf{x}_n,d_n) \in \mathcal{D} \tag{3.1}$$

Nous noterons $\mathbf{w}_0$ et b_0, les valeurs telles que l'hyperplan de séparation soit optimal et $\mathbf{x}_{\text{sv}}$, les "Support Vectors" (SV), c'est-à-dire les vecteurs les plus proches de l'hyperplan

optimal de séparation. Nous pouvons remarquer que la marge ρ (c.f. figure **??**) est donnée par l'expression suivante:

$$\rho = \frac{2}{\|\mathbf{w}_0\|} \tag{3.2}$$

Si nous voulons trouver l'hyperplan optimal de séparation, il s'agira de maximiser la marge ρ et donc de minimiser la norme du vecteur $\mathbf{w}$, ce qui peut être fait en minimisant la fonction de coût $\phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$ sous la contrainte (**??**). Sans entrer dans les détails, nous pouvons mentionner que ce problème peut être résolu au moyen de la méthode des multiplicateurs de Lagrange en posant la fonction de Lagrange suivante:

$$J(\mathbf{w},b,\boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{s=1}^{S} \alpha_s(d^s(\mathbf{w}^T\boldsymbol{\psi}(\mathbf{x}^s) + b) - 1) \tag{3.3}$$

Après avoir minimisé cette fonction par rapport à $\mathbf{w}$,b et maximisé par rapport à $\boldsymbol{\alpha}$, nous pouvons calculer les multiplicateurs de Lagrange optimaux $\boldsymbol{\alpha}_0$ et ainsi exprimer le vecteur poids optimal $\mathbf{w}_0$ ainsi que le paramètre b_0:

$$\mathbf{w}_0 = \sum_{s=1}^{S} \alpha_{0,s} d_s \boldsymbol{\psi}(\mathbf{x}^s) \tag{3.4}$$

$$b_0 = 1 - \mathbf{w}_0^T\mathbf{x}_{\mathrm{sv}} \tag{3.5}$$

La fonction Lagrangienne devient alors:

$$Q(\boldsymbol{\alpha}) = \sum_{s=1}^{S} \alpha_s - \frac{1}{2}\sum_{s=1}^{S}\sum_{s'=1}^{S} \alpha_s\alpha_{s'}d^s d^{s'} \underbrace{\boldsymbol{\psi}(\mathbf{x}^s)^T\boldsymbol{\psi}(\mathbf{x}^{s'})}_{=K(\mathbf{x}^s,\mathbf{x}^{s'})} \tag{3.6}$$

Plutôt que de considérer le produit $\boldsymbol{\psi}(\mathbf{x}_n)^T\boldsymbol{\psi}(\mathbf{x}^{s'})$, nous utiliserons un noyau $K(\mathbf{x}^s,\mathbf{x}^{s'})$. Les noyaux les plus couramment utilisés sont les suivantes:

- linéaire: $K_{\mathrm{l}}\left(\mathbf{x}^s,\mathbf{x}^{s'}\right) = \mathbf{x}^{sT}\mathbf{x}^{s'}$
- polynomiale: $K_{\mathrm{p}}\left(\mathbf{x}^s,\mathbf{x}_{s'}\right) = (\mathbf{x}^{sT}\mathbf{x}^{s'} + c)^d$
- gausienne: $K_{\mathrm{g}}\left(\mathbf{x}^s,\mathbf{x}^{s'}\right) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}^s - \mathbf{x}^{s'}\|^2\right)$

Nous pouvons aisément généraliser le problème en utilisant M classes ($d^s \in \{1,\ldots,M\}$). Dans un problème à M classes, nous avons M hyperplans optimaux de séparation. Pour les déterminer, il suffit de se ramener M fois à la situation à deux classes en considérant la classe désirée et l'ensemble des autres classes.

Dans ce cas, la fonction de discrimination ne correspond pas aux hyperplans de séparation (c.f. figure **??**). Un nouveau vecteur appartient à la classe pour laquelle la distance entre ce vecteur et l'hyperplan est minimal.

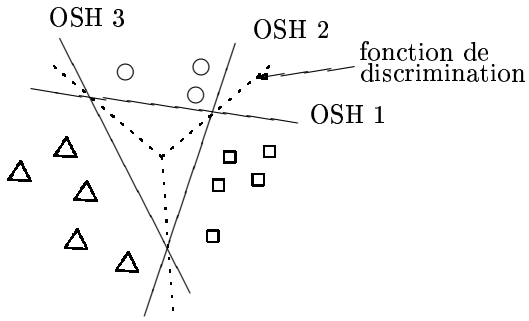

FIG. 3.2 – *Classes multiples*

Chapitre 4

Base de données

4.1 Description de l'expérience

Commençons par décrire l'expérience qui a fourni les données utilisées pour la classification.

Quatre sujets en bonne santé ont été testés. Il leur a été demandé de fixer une croix centrée sur un écran et de répondre aussi vite que possible à un stimulus visuel latéral (point noir de largeur correspondant à un angle de 0.5^0) au moyen d'une action prédéfinie.

Au cours des sessions d'enregistrement, les stimuli sont apparus toutes les 5 à 6 secondes de façon aléatoire dans le champ visuel gauche (LVF) ou le champ visuel droit (RVF). Les stimuli ont été présentés durant 60 ms à un angle horizontal de 4^0 par rapport à la croix. Celle-ci tient également lieu d'avertissement précédant le stimulus de 3 à 4 secondes et reste visible jusqu'à la fin de la réponse.

Dans la moitié des sessions, les réponses ont été données par l'index de la main gauche (LH) et dans l'autre moitié avec l'index de la main droite (RH).

L'ensemble des données à disposition contient 959 essais. Ceux-ci sont répartis non uniformément sur quatre sujets et quatre classes: C_1 = (LVF,LH), C_2 = (LVF,RH), C_3 = (RVF,LVF), C_4 = (RVF,RH). Chaque essai est composé de $\bar{T} = T/\Delta t$ cartes dont chacune d'elles contient l'information venant de $E = 122$ électrodes.

Un quart des données sont laissées de côté pour la base de test $\mathcal{T}$ et le reste appartient à la base d'entraînement $\mathcal{E}$. Si S_c représente le nombre de sujets d'entraînement appartenant à la classe c, nous avons $S_1 = 169$,$S_2 = 192$,$S_3 = 177$,$S_4 = 180$.

4.2 Prétraitement

Dans une première phase, nous avons centré et normalisé chaque essai pour qu'ils puissent être comparés sur un pied d'égalité. Ce choix de prétraitement est motivé par les résultats obtenus dans "Support Vector Machines: une Application à la Classifiaction des EEG". Plus précisément, le centrage et la normalisation sont donnés par

$$\tilde{x}_e^s(t) = P2(x_e^s(t)) = P1\left(x_e^s(t) - \frac{1}{\bar{T}E}\sum_{e'=1}^{E}\sum_{t'=1}^{\bar{T}} x_{e'}^s(t')\right) \tag{4.1}$$

où

$$P1(x_e^s(t)) = \frac{x_e^s(t)}{\sum_{e'=1}^{E}\sum_{t'=1}^{\bar{T}} x_{e'}^s(t')^2} \tag{4.2}$$

4.3 Réduction de la dimensionalité avec PCA

L'idéal serait de pouvoir effectuer la classification directement sur les données telles que nous venons de les normaliser. En ce qui concerne les réseaux diffusifs, cela est tout simplement impossible dû au temps de calcul extrêmement important. En effet l'algorithme est en N^2, ce qui signifie qu'il est impossible de faire correspondre un neurone à chaque électrode.

Pour réduire le temps de calcul, il faudrait pouvoir réduire le nombre d'électrodes utilisées par un facteur 10. Mais comme nous ne savons pas quelles sont les électrodes qui contiennent le plus d'information quant à la classification, nous avons décidé d'utiliser une PCA *(Principal Component Analysis)* pour réduire la dimensionalité d'entrée.

Soit C, la matrice de covariance de toutes les cartes de la base d'entraînement $\mathcal{E}$ définie comme suit:

$$C = \sum_{t=1}^{\bar{T}} V(t)V(t)^T \tag{4.3}$$

où $V(t)$ est un matrice $E \times S$ dont l'élément $V_{es}(t) = x_e^s(t)$ représente le potentiel de l'électrode e, du sujet s et au temps t. Soient $\Gamma_1, \ldots, \Gamma_P$, les P vecteurs propres de C associés aux P valeurs propres les plus grandes. Ces vecteurs propres sont sensés représenter les structures spatiales (au sens des cartes) dominantes. Il est ainsi possible de définir une nouvelle base de donnée d'entraînement $\widetilde{\mathcal{E}}$ et de test $\widetilde{\mathcal{T}}$ dont les composantes de chaque élément sont données par les projection de $x^s(t)$ sur $\Gamma_p, \quad p = 1, \ldots, P$:

$$\tilde{x}_p^s(t) = x^s(t)\Gamma_p, \quad \forall(s,p,t) \in \mathbb{N}_S \times \mathbb{N}_P \times \mathbb{N}_{\bar{T}} \tag{4.4}$$

Les 10 premières composantes que nous avons conservées sont représentées en annexe **??**. La somme de leur variance représente 85.2% de la variance totale. La figure **??** montre bien que si nous voulions augmenter un peu le pourcentage de variance expliquée, il faudrait ajouter beaucoup de composantes.

La taille des vecteurs décrivant l'état à un temps t donné est passée de $E = 122$ à $P = 10$. Nous avons ainsi un réseau possédant 10 neurones observables et nous avons fixé à 4 le nombre de neurones cachés.

Pour donner une idée de la difficulté de la classification, nous avons représenté sur la figure **??**, les projections de 20 essais appartenant à la même classe sur les 3 premières composantes principales. En d'autres termes, cette figure représente la distribution de probabilité de chemins qu'il s'agira d'apprendre.

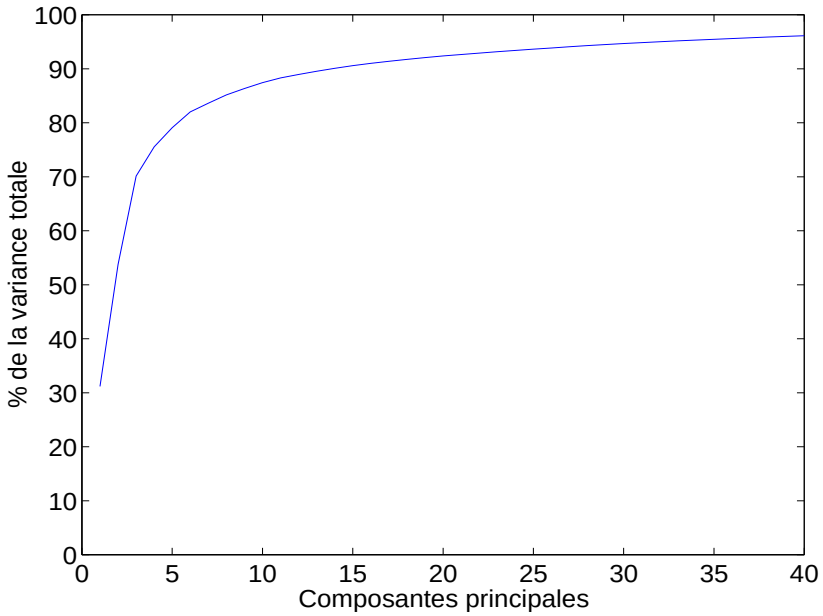

FIG. 4.1 – *La somme de la variance des premières composantes*

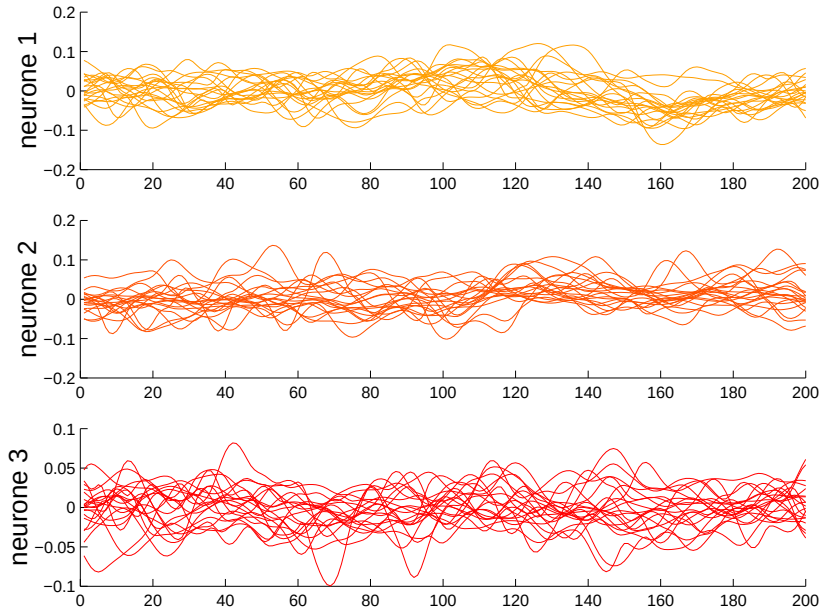

FIG. 4.2 – *Les 3 premières composantes de* $x^1, \ldots, x^{20} \in \widetilde{\mathcal{E}}$ *appartenants à la classe 1.*

Chapitre 5

Résultats

5.1 Classification avec des réseaux diffusifs

5.1.1 Choix des paramètres

Pour pouvoir entraîner λ sur une base d'entraînement $\widetilde{\mathcal{E}} = \{x_o^s\}_{s=1}^S$, l'algorithme (décrit en annexe **??**) a besoin d'un certain nombre de paramètres. Parmi ceux-ci figure une condition initiale pour les poids $W^{(0)}$, pour les constantes de temps $\boldsymbol{\gamma}^{(0)}$, pour l'amplitude du bruit $\sigma^{(0)}$ et pour la valeur des neurones cachés $\boldsymbol{h}(1)$. Il est encore nécessaire de préciser le paramètre θ de la fonction d'activation, le paramètre α contrôlant le taux de diminution de σ ainsi que sa borne inférieure σ_{end}.

Le nombre K d'itérations est fixé au départ mais il pourrait être déterminé en cours de calcul lorsque la fonction de vraisemblance pour la base d'entraînement ne diffère pas plus d'un ϵ d'une itération à l'autre. Le nombre M de chemins dans la couche cachée doit être suffisamment grand pour approximer correctement l'intégrale sur ces chemins, mais suffisamment petit car le temps de calcul en est directement proportionnel (c.f. paragraphe **??**).

Il reste un dernier choix que nous caractériserons δ. Si $\delta = 1$, l'algorithme entraîne les constantes de temps et les mets à jour à chaque itération. Dans le cas contraire, $\boldsymbol{\gamma}$ est fixé au départ et reste inchangé au cours des itérations. Le paragraphe **??** nous permettra de saisir la différence entre ces deux types d'entraînement.

Pour toutes les simulations effectuées, nous avons laissé certains paramètres identiques, soient $W_{ij}^{(0)} = 0\ \forall i,j = 1,\ldots,N$, $\boldsymbol{h}(1) = 0$ et $M = 10,\theta = 1$.

Pour condenser l'écriture, notons $\xi = (\lambda^{(0)},\boldsymbol{h}(1),\sigma^{(0)},\sigma_{\text{end}},\alpha,\delta,K,\theta,M,\Delta t)$ l'ensemble des paramètres nécessaires à l'entraînement de λ. Puisque le nombre de paramètres est élevé et que le temps de calcul est excessivement long (c.f.paragraphe **??**), il a falllu se restreindre à 4 configurations que nous avons choisies de la façon suivante:

- $\xi_1 = ((0,\boldsymbol{\gamma}),\mathbf{0},0.2,6.7\cdot 10^{-3},0.98,0,199,10,1,1)$ où $\gamma_i = \frac{1}{5}\ i = 1,\ldots,N$
- $\xi_2 = ((0,\boldsymbol{\gamma}),\mathbf{0},0.1,9\cdot 10^{-3},0.98,0,110,10,1,1)$ où $\gamma_i = \frac{1}{5}\ i = 1,\ldots,N$
- $\xi_3 = ((0,\boldsymbol{\gamma}),\mathbf{0},0.05,9.7\cdot 10^{-3},0.95,0,50,10,1,1)$ où $\gamma_i = \frac{1}{5}, i = 1,\ldots,D$ et $\gamma_i = \frac{1}{2^{(i-D)}}, i = D+1,\ldots,N$
- $\xi_4 = ((0,\boldsymbol{\gamma}),\mathbf{0},0.1,4.3\cdot 10^{-3},0.96,1,98,10,1,1)$

Après avoir déterminé les paramètres nécessaires à l'entraînement, il faut préciser les paramètres nécessaires à la classification. Soit $\chi = (\lambda,\boldsymbol{h}(1),\sigma,\theta,\Delta t,M)$, l'ensemble de paramètres dont la fonction de vraisemblance $\hat{L}_o^\lambda$ a besoin. Le paramètre λ correspond évidemment à $\lambda^{(K)}$ obtenu lors de la dernière itération d'entraînement. En ce qui concerne $\boldsymbol{h}(1),\theta,\Delta t$ et M, nous utiliserons les mêmes valeurs que celles utilisées lors de l'entraînement. Finalement, pour ce qui est du paramètre σ nécessaire au calcul de $\hat{L}_o^\lambda$, nous avons choisis la dernière valeur de la phase d'entraînement, soit σ_{end}.

5.1.2 Temps de calcul

Dans la situation où l'entraînement est le plus rapide, c'est-à-dire lorsqu'il n'est pas nécessaire d'adapter les constantes de temps, le temps de calcul de l'algorithme est proportionnel à $KSM\bar{T}N^2$ si K,S,$\bar{T}$ et N sont suffisamment grands.

Pour donner un exemple, une itération pour un réseau avec $S = 180$,$M = 10$,$\bar{T} = 200$,$N = 14$ prend environ deux heures de calcul sur une station Unix SunBlade 100, 500 MHz! En prenant 200 itérations, chacune des 4 classes nécessite plus de 16 jours de calcul! Si, de plus, nous désirons entraîner un réseau sur les données non-réduites $\mathcal{E}$, il faudrait plus de 5 années de calcul pour une seule configuration!

Il est clair que l'algorithme n'a pas été optimisé au niveau du temps de calcul puisque tel n'est pas le but de projet. Il aurait été possible de programmer en Fortran et de paralléliser le calcul, ce qui aurait permis un énorme gain de temps. Il aurait été aussi possible de calculer l'exponentielle d'un grand nombre de valeurs et de les garder en mémoire afin d'éviter la répétion de calculs identiques. En effet, la grande partie du temps est passée à calculer la fonction d'activation φ.

Dans la situation où l'entraînement est plus lent (i.e. $\delta = 1$), il suffit de multiplier le temps de calcul mentionné par un facteur 2, ce qui rend d'autant plus difficile l'ajustement des paramètres d'entraînement.

5.1.3 Taux de classification

Configuration ξ_1

Dans cette première situation, l'amplitude du bruit commence à une valeur relativement élevée ($\sigma_0 = 0.2$) et le taux de diminution $\alpha = 0.98$ est également élevé. Il faut donc beaucoup d'itérations avant d'atteindre la valeur d'arrêt $\sigma_{\text{end}} = 6.71 \cdot 10^{-3}$. Ce lent "refroidissement" permet théoriquement de trouver une meilleure solution qu'un shéma plus rapide, mais il faut en payer le prix en temps de calcul.

Avant de poursuivre dans l'analyse des résultats, définissons la quantité $q(k)$ [1] définie par

$$
\begin{aligned}
q(k) &= \frac{1}{S}\sum_{s=1}^{S} \max_m \log\left(\sigma^2 w^s(m)\right) \qquad (5.1)\\
&= \frac{1}{S}\sum_{s=1}^{S} \max_m \left(\sum_{t=1}^{\bar{T}-1} \boldsymbol{\mu}_o(\boldsymbol{x}^{m,s}(t),\lambda^{(k)}) \cdot d\boldsymbol{x}_o(t) - \frac{1}{2}\|\boldsymbol{\mu}_o(\boldsymbol{x}^{m,s}(t),\lambda^{(k)})\|^2 \Delta t \right)
\end{aligned}
$$

Celle-ci représente l'importance ou le poids du meilleur chemin moyenné sur tous les essais. En effet, sur les M chemins générés par la distribution S_h^{λ,x_o}, certains sont plus probables que d'autres et ont ainsi une importance différente. Plutôt que de considérer l'évolution de la fonction de vraisemblance, nous étudierons l'évolution du meilleur chemin, même si la forme qualitative des deux courbes est semblable. L'apprentissage doit permettre au meilleur chemin d'augmenter d'importance au cours des itérations, ce que nous voyons sur la figure **??**. Celle-ci représente l'évolution du meilleur chemin des classes une à quatre avec la configuration ξ_1. La ligne traitillée indique l'itération pour laquelle σ atteint sa valeur minimale.

Après avoir entraîné les quatre réseaux, il possible de classifier la base de test $\widetilde{\mathcal{T}}$. Nous avons ainsi obtenu les résultats du tableau **??**. Celui-ci représente le nombres d'essais appartenant à la classe C_c et classifiés dans la classe $R_{c'}$, c,$c' = 1,\ldots,4$ pour la configurations ξ_1. Tous les nombres en dehors de la diagonales représentent donc les essais mal classifiés.

1. $w^s(m)$ est une notation allégée de $p^{\lambda}(x_o^s,h^{m,s})$ que nous retrouvons dans l'annexe **??**.

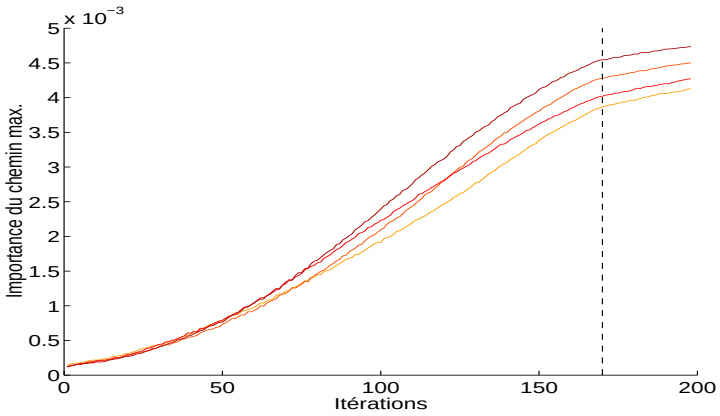

FIG. 5.1 – *Courbes d'apprentissage des classes 1, 2, 3 et 4 avec la configuration* ξ_1

ξ_1	C_1	C_2	C_3	C_4
R_1	1	1	1	1
R_2	6	12	9	10
R_3	2	5	3	3
R_4	47	47	46	44

TAB. 5.1 – *Classification des 241 essais de test avec la configuration* ξ_1

Par exemple, sur les 59 essais appartenant à la classe 3 (i.e. stimulus visuel droit et réponse à l'aide de la main gauche), 46 ont été classifiés dans la classe 4, ce qui est très mauvais. En fait cette situation est identique pour les 4 classes, car les essais sont classifiés pour la grande majorité dans la classe 4, ce qui signifie que la fonction de vraisemblance de la classe 4 est en moyenne plus élevée que celle des autres classes.

Ce phénomène peut être en partie expliqué par les courbes d'apprentissages que nous venons de mentionner. En effet, si le classificateur lié à la tâche 1 apprend plus rapidement que celui lié à la tâche 2, il est fort probable qu'un nouvel essai, indépendemment de la classe à laquelle il appartient, possédera une valeur de vraisemblance supérieur pour la classe 1 que pour la classe 2.

Puisque chaque classe de $\widetilde{\mathcal{T}}$ possède un nombre différent d'essais ($S_1 = 56$, $S_2 = 65$, $S_3 = 59$ et $S_4 = 61$), il existe plusieurs façons de définir le taux de classification. La manière la plus simple, et c'est celle que nous choisirons, consiste à effectuer le rapport du nombre d'essais classifiés correctement sur le nombre total d'essais ($S_{\text{tot}} = 241$). Une autre définition du taux de classification consiste en la moyenne du taux de classification pour chaque classe.

Nous avons donc ici, au sens de la première définition, un taux de 24.9% pour une classification à 4 tâches et respectivement 52.1% et 47.6% pour une classification de C_1/C_4 et C_2/C_3, ce qui correspond à un classificateur aléatoire! Les classes C_1 et C_4 sont des *classes de transfert hémisphérique*. En effet, si le stimulus se trouve dans le champ visuel gauche, la zone activée doit se trouver principalement dans l'aire primaire visuelle droite et devra se déplacer dans l'autre hémisphère cérébral pour finalement atteindre le cortex moteur de l'index de la main droite.

Configuration ξ_2

Cette configuration est similaire à la première que nous venons d'étudier. La différence majeure est le programme de diminution de l'amplitude du bruit. En effet celui-ci est plus

rapide que le précédent puisque α et $\sigma^{(0)}$ sont plus faibles.

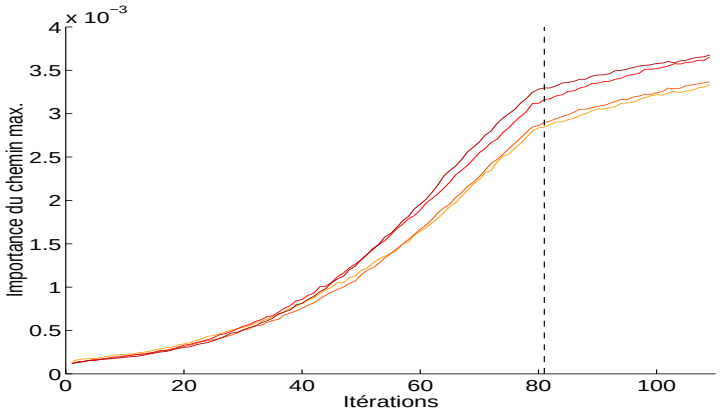

FIG. 5.2 – *Courbes d'apprentissage des classes 1, 2, 3 et 4 avec la configuration* ξ_2

Si nous comparons les courbes d'apprentissage représentées sur la figure **??** avec celles de la figure **??**, nous pouvons remarquer que la configuration ξ_2 permet une augmentation plus rapide de la vraisemblance, mais elle sature également plus rapidement, ce qui est dû au schéma de "refroidissement" plus rapide.

Nous pouvons également remarquer que contrairement aux courbes d'apprentissage de la figure **??**, la vraisemblance des classes 2 et 3 de la configuration ξ_2 ne se croisent pas, ce qui veut dire que la forme des courbes d'apprentissage n'est pas conservée d'une configuration à l'autre.

Pour ne pas alourdir la présentation, les détails de classification figureront, à partir d'ici, dans l'annexe **??**. Le tableau **??** est l'analogue du tableau **??** pour la la configuration ξ_2. Le même problème de classification s'y trouve même s'il est légèrement moins marqué.

Les taux de classification (c.f. tableau **??**) sont tout aussi insignifiants que ceux obtenus avec la paramétrisation ξ_1 en raison du biais de la fonction de vraisemblance.

Configuration ξ_3

La principale différence entre la configuration ξ_3 et les deux précédentes se trouve dans les constantes de temps. Bien que celles-ci soient les mêmes dans les neurones observables, elles diffèrent dans les couches cachées. Ceci permet de donner des mémoires différentes à chaque neurone caché, puisque nous considérons τ_i comme le temps de mémoire du $i^{\text{ème}}$ neurone.

La figure **??** représente les courbes d'apprentissage avec la configuration ξ_3 et le tableau **??** nous présente le détail de la classification des 241 essais avec la paramétrisation ξ_3.

Configuration ξ_4

Cette configuration est passablement différente des précédentes puisque les constantes de temps sont mises à jour à chaque itération, ce qui ralentit d'un facteur 2 le temps de calcul, mais qui doit permettre également d'obtenir des résultats plus précis.

En observant la figure **??**, nous pouvons remarquer des oscillations de l'importance du meilleur chemin lorsque le nombre d'itérations est élevé et donc lorsque σ est petit. Ceci est peut-être dû au programme de diminution du bruit. En effet, après chaque diminution du bruit, un pas EM est effectué, alors qu'il en faudrait certainement plusieurs avant de changer à nouveau le paramètre σ.

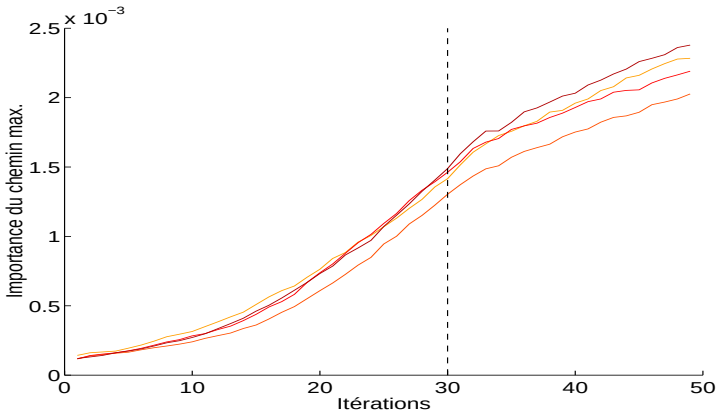

FIG. 5.3 – *Courbes d'apprentissage des classes 1, 2, 3 et 4 avec la configuration* ξ_3

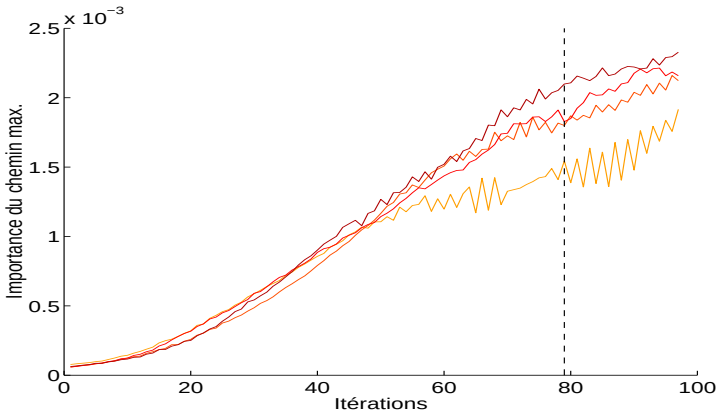

FIG. 5.4 – *Courbes d'apprentissage des classes 1, 2, 3 et 4 avec la configuration* ξ_4

En ce qui concerne le taux de classification, le tableau **??** nous montre que le classificateur gauche-gauche réagit comme les autres classificateurs gauche-gauche que nous avons vus jusqu'à présent. En effet la valeur moyenne de l'importance du meilleur chemin est systématiquement plus faible que celle produite par les autres classificateurs (c.f. figures **??**-**??**).

5.1.4 Correction de la vraisemblance

Une façon de corriger le problème du biais mentionné plus haut est d'arrêter l'entraînement de chaque réseau pour la même valeur de q. En d'autres termes, l'entraînement du réseau c s'arrête à la k_c^{ime} itération de façon à ce que

$$q_1(k_1) = \cdots = q_4(k_4) \tag{5.2}$$

Nous noterons $\hat{\xi}_2$ = la nouvelle configuration décrivant les 4 réseaux où ξ_2^c possède les mêmes valeurs que ξ_2 à l'exception du paramètre K_c qui satisfait la condition (**??**). La figure **??** montre les courbes d'apprentissage pour les configurations $\hat{\xi}_2$ et $\hat{\xi}_3$. Ainsi, la classification s'effectue avec le paramètre λ tel que le réseau c utilise le paramètre $\lambda^{(k_c)}$.

Même en utilisant cette technique, les taux de classification ne s'améliorent pas. Par contre, nous pouvons voir sur le tableau **??** qu'à l'exception de la classe 1, le problème du

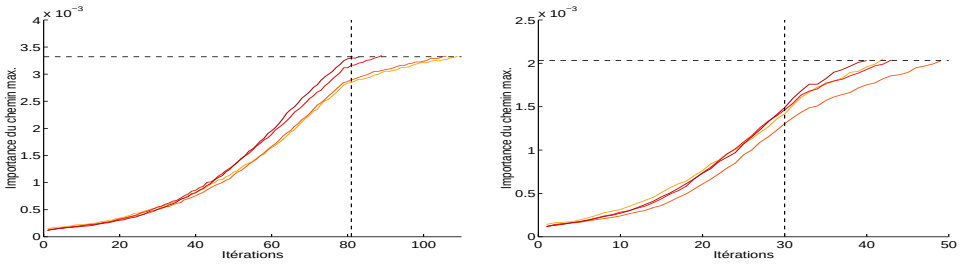

FIG. 5.5 – *Courbes d'apprentissage des classes 1, 2, 3 et 4 avec la configuration $\hat{\xi}_2$ à gauche et $\hat{\xi}_3$ à droite en s'arrêtant à la même valeur de $q(k)$*

biais est résolu.

5.2 Classification avec des SVM

5.2.1 Classification de $\widetilde{\mathcal{E}}$

Pour pouvoir comparer les résultats obtenus dans le paragraphe précédent avec d'autres valeurs, nous avons utilisé l'algorithme SVM sur la même base d'entraînement $\widetilde{\mathcal{E}}$, i.e. la base de donnée réduite aux projections sur les composantes principales.

Le tableau **??** représente le nombre d'essais appartenants à la classe C_c et classifiés dans la classe $R_{c'}$ c,$c' = 1,\ldots,4$. Nous avons utilisé le noyau linéaire, le noyau polynomial ($d = 2$,$c = 1$) et le noyau gaussien pour lesquels nous obtenons des taux de classification de 36.9%,38.6% et 39.8%, ce qui n'est pas excellent, mais nettement meilleur que ceux obtenus avec les réseaux diffusifs.

noyau	C_1/C_4		C_2/C_3		C_c,$c = 1,\ldots,4$				
	%	NSV	%	NSV	%	NSV1	NSV2	NSV3	NSV4
K_l	74.4	161	71	169	36.9	310	340	328	305
K_p,$d = 2$	72.7	256	70.2	280	38.6	476	511	503	484
K_g,$\sigma = 0.5$	66.7	349	63.7	369	39.8	718	718	718	718

TAB. 5.2 – *Classification avec PCA à 2 tâches (classe 1 contre 4 et classe 2 contre 3) et à 4 tâches*

Pour ce qui est de la classification à 2 tâches, le taux s'élève à 71% (C_2/C_3) et 74.4% (C_1/C_4) avec un noyau linéaire (c.f. tableau **??**). Les détails de classification sont présentés sur le tableau **??**.

5.2.2 Classification de $\mathcal{E}$

Puisque les taux de classification obtenus avec des réseaux diffusifs sont médiocres, nous pouvons nous poser la question si la technique PCA est judicieuse pour notre situation. Or, comme il est impossible d'entraîner un réseau diffusif sur la base de donnée $\mathcal{E}$, nous avons entraîné l'algorithme SVM sur cette base de données. Les vecteurs d'entrée sont ainsi de dimension $E\bar{T} = 24400$.

Le tableau **??** ainsi que le tableau plus détaillé **??** nous montre que le meilleur taux de classification est obtenu avec le noyau linéraire, ce qui était déjà le cas pour la classification à 2 tâches sur la base $\widetilde{\mathcal{T}}$.

noyau	C_1/C_4		C_2/C_3		C_{1-4}				
	%	NSV	%	NSV	%	NSV1	NSV2	NSV3	NSV4
K_l	76.9	205	71.8	219	44.4	394	413	407	388
K_p,$d=2$	74.4	287	75	302	43.6	519	544	534	528
K_g,$\sigma=0.5$	54.7	349	58.8	369	35.7	718	718	718	718

TAB. 5.3 – *Classification sans PCA à 2 tâches (classe 1 contre 4 et classe 2 contre 3) et à 4 tâches*

5.3 Récapitulatif

Pour garder une vue synthétique de tous les résultats obtenus, nous avons représenté sur le tableau **??** les taux de classification obtenus grâce aux réseaux de neurones diffusifs (DNN) et aux SVM avec différentes paramétrisations.

alg.	prétrait.	param.	C_{1-4}	C_1/C_4	C_2/C_3
			%	%	%
DNN	PCA	ξ_1	24.9	52.1	47.6
DNN	PCA	ξ_2	27.4	56.4	46
DNN	PCA	ξ_3	24.9	52.1	54
DNN	PCA	ξ_4	29	53	54
DNN	PCA	$\hat{\xi}_2$	25.7	58.1	41.9
DNN	PCA	$\hat{\xi}_3$	24.9	50.4	55.7
SVM	PCA	K_l	36.9	74.4	71
SVM	PCA	K_p,$d=2$	38.6	72.7	70.2
SVM	PCA	K_g,$\sigma=0.5$	39.8	66.7	63.7
SVM		K_l	44.4	76.9	71.8
SVM		K_p,$d=2$	43.6	74.4	75
SVM		K_g,$\sigma=0.5$	35.7	54.7	58.8

TAB. 5.4 – *Tableau récapitulatif des taux de classification*

Nous remarquons aisément que les SVM sont clairement supérieurs aux réseaux diffusifs pour ce qui est de la classification à 4 tâches (C_{1-4}) ou à 2 tâches (C_1/C_4 et C_2/C_3). En fait, nous pouvons nous demander si les réseaux diffusifs ont effectivement classifié quoi que ce soit puisque le taux de classification à 2 tâches avoisine 50% tandis que celui à 4 tâches avoisine 25%!

Puisque le taux de classification de l'algorithme SVM sur la base donnée $\widetilde{\mathcal{T}}$ (SVM & PCA) n'est pas si loin de celui obtenu sur $\mathcal{T}$ (SVM), nous ne pouvons pas prétendre que les 10 composantes principales ne représentent que du bruit et ainsi expliquer entièrement le mauvais taux de classification des réseaux diffusifs.

Pour ce qui est des configurations ξ_1 à ξ_4, nous avons vu qu'il est nécessaire d'arrêter le calcul à différentes itérations de façon à ce que $q_1(k_1) = \cdots = q_4(k_4)$, ce qui permet d'enlever le biais de vraisemblance entre les différentes classes.

La raison pour laquelle les paramétrisations $\hat{\xi}_2$ et $\hat{\xi}_3$ n'ont pas donné satisfaction est certainement liée au petit nombre d'itérations, ce qui ne permet pas d'obtenir un réelle convergence de la fonction de vraisemblance lors de l'apprentissage (c.f. figure **??**).

A titre de comparaison, le taux de classification obtenu sur des moyennes d'une cinquantaine d'essais vaut 54.2% pour le problème à 2 taches (C_1/C_4) et 91.7% pour le problème à 4 tâches [**?**].

Chapitre 6

Conclusion

En commençant ce projet, la question était de savoir s'il était possible de classifier des essais individuels plutôt que des moyennes sur différents essais. Puisque cette tâche est loin d'être triviale, nous nous sommes tournés vers les réseaux diffusifs étant donné que l'algorithme SVM, bien que très efficace, n'est pas adapté à la classification de séquences temporelles.

Contrairement à ce que nous attendions, les réseaux diffusifs n'ont pas été la solution magique pour la classification de signaux EEG, du moins avec les choix que nous avons faits.

Le principal inconvénient de ce type d'algorithme est le temps de calcul gigantesque. En effet, s'il était plus raisonnable, nous pourrions tester un plus grand nombre de configurations différentes de paramètres et ainsi espérer de meilleurs résultats.

Puisque l'utilisation d'une PCA pour l'algortihme SVM ne détériore pas de façon significative le taux de classification, nous pouvons penser que cette technique permet de conserver la majorité de l'information importante et, ainsi, qu'il n'est pas nécessaire de chercher à entraîner le réseau sur toutes les électrodes.

Etant donné qu'en théorie les réseaux diffusifs sont capables d'approximer n'importe quelle distribution de probabilité de chemins continus, il serait intéressant, dans une perspective future, de poursuivre dans cette voie tout en cherchant à optimiser le temps de calcul de l'algorithme d'entraînement des paramètres. Pour ce faire il faudrait certainement développer un algorithme différent de l'algorithme EM qui a besoin de la méthode de Monte-Carlo.

Une autre façon d'améliorer le taux de classification, et du reste plus intéressante, serait d'améliorer notre compréhension des potentiels évoqués du cerveau et de permettre ainsi un prétraitement plus judicieux.

Même si résultats obtenus avec les réseaux diffusifs ne sont pas convaincants, il faut tout de même mentionner que les taux de classification à 2 tâches avec les SVM s'élève à plus de 70%, ce qui est tout à fait remarquable puisqu'il s'agit d'un premier résultat concernant la classification d'essais uniques.

Finalement, ce travail met bien en évidence la complexité des signaux EEG ainsi que notre difficulté à en élaborer un modèle cohérent.

Bibliographie

[1] B. E. Boser, I. M. Guyon et V. N. Vapnik, 1992. "A training algorithm for optimal margin classifiers," *Proceedings of the 5th Anuual ACM Workshop on Computational Learning Theory*, pp 144 - 152. ACM Press.

[2] T. M. Cover, 1965. "Geometricaland statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Transactions on Electronic Computer*, vol. EC-14, pp. 326-334.

[3] A.P. Dempster, N. M. Laird et D. B. Rubin, 1977. "Maximum-likelyhood from incomplete data via the em algorithm," *J. Royal Statist. Soc. Ser. B.*, vol. 39.

[4] S. Haykin, 1999. *Neural Networks, a comprehensive foundation*, Prentice-Hall.

[5] J. Hopfield, 1984. "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the National Academy of Science*, vol. 81, pp. 3088-3092.

[6] I. Karatzas et S. E. Shreve, 1991. *Brownien motion and stochastic calculus*, Springer.

[7] S. Kirkpatrick, C. D. Gelatt et M. P. Vecchi, 1983. "Optimisation by Simulated Annealing," *Science*, vol. 220, No 4598, pp. 671-680.

[8] D. Lehmann et W. Skrandes, 1984. "Spatial analysis of evoked potentials in man - A review," *Prog. Neurobiol.,* vol. 23, pp. 227-250.

[9] C. M. Michel, M. Seeck et T. Landis, 1999. "Spatiotemporal Dynamics of Human Cognition," *News Physiol. Sci.*, vol. 14, pp. 206-214.

[10] J. del R. Millán, J. Mouriño, M. Franzé, F. Cincotti, M. Varsta, J. Heikkonen et F. Babiloni, 2002. "A Local Neural Classifier for the Recognition of EEG Patterns Associated to Mental Tasks," *IEEE Transactions on Neural Networks*, à venir.

[11] J. R. Movellan, P. Mineiro et R. J. Williams, 2000. "A monte-Carlo EM Approach for Training Partially Observable Diffusion Networks," *INC Technical Report.*

[12] B. Oksendal, 1991. *Stochastic differential equation*, Springer-Verlag.

[13] R. D. Pascual-Marqui, C. M. Michel et D. Lehmann, 1995. "Segmentation of Brain Electrical Activity into Microstates: Model Estimation and Validation," *IEEE Transactions on Biomedical Endineering*, vol. 42, pp. 658-665.

[14] J.-P. Pfister et W. Gerstner, 2001. "Support Vector Machines: Une application à la Classification des EEG", non publié.

[15] B. Schölkopf, P. Simard, A. Smola et V. Vapnik 1998. "Prior Knowledge in Support Vector Kernels," *Advances in Neural Information Processing Systems 10*, Cambridge, MA, pp. 640 - 646. MIT Press.

[16] G. Thut, C.-A. Hauert, S. Morand, M. Seeck, T. Landis et C. Michel, 1999. "Evidence for interhemispheric motor-level transfer in a simple reaction time task: an EEG study," *Exp. Brain Res.*, vol. 128, pp. 256-261.

Annexe A

Détails de calcul

Avant de démontrer la relation (**??**), commençons par établir le résultat suivant que nous avons également utilisé au paragraphe **??**:

$$\frac{dQ^{\lambda}_{h|o}}{dS^{\lambda,x_o}_{h}}(h|x_o) = p^{\lambda}_{h|o}(h|x_o) \quad \forall h \in \mathcal{H} \tag{A.1}$$

où $\mathcal{H} = \{h^m\}_{m=1}^{M}$ est un ensemble de chemins de la couche cachée générés par la distribution S^{λ,x_o}_{h}. Pour ce faire, rappelons premièrement la définition de $dQ^{\lambda}_{h|o}(x_h|x_o)$.

$$dQ^{\lambda}_{h|o}(x_h|x_o) = \frac{dQ^{\lambda}(x_o,x_h)}{dQ^{\lambda}_{o}(x_o)} \quad \forall (x_o,x_h) \in \Omega_o \times \Omega_h \tag{A.2}$$

En raison de l'indépendance de chacune des composantes du mouvement brownien, nous avons

$$dR(x_o,x_h) = dR_o(x_o)dR_h(x_h) \tag{A.3}$$

A l'aide du théorème de Girsanov (**??**) et de la relation (**??**), nous pouvons écrire

$$\begin{aligned} \frac{L^{\lambda}(x_o,x_h)}{L^{\lambda}_{o}(x_o)} &= \frac{dQ^{\lambda}}{dR}(x_o,x_h)\frac{dR_o}{dQ^{\lambda}_{o}}(x_o)\frac{dR_h}{dR_h}(x_h) \\ &= \frac{dQ^{\lambda}_{h|o}}{dR_h}(x_h|x_o) \end{aligned} \tag{A.4}$$

Pour obtenir la relation (**??**), il suffit d'utiliser le résultat précédent et la définition de $p^{\lambda}_{h|o}(h|x_o)$ donnée par l'équation (**??**).

$$\begin{aligned} \frac{dQ^{\lambda}_{h|o}}{dS^{\lambda,x_o}_{h}}(h|x_o) &= \frac{L^{\lambda}(x_o,h)}{L^{\lambda}_{o}(x_o)}\frac{dR_h}{dS^{\lambda,x_o}}(h) \\ &= \frac{p^{\lambda}(x_o,h)}{L_o(x_o)} \\ &= p^{\lambda}_{h|o}(h|x_o) \end{aligned} \tag{A.5}$$

La relation (**??**) étant établie, nous pouvons calculer le gradient du logarithme de la fonction de vraisemblance $L^{\lambda}_{o}(x_o)$.

$$\begin{aligned}
\nabla_\lambda \log L_o^\lambda(x_o) &= \int_{\Omega_h} \frac{\nabla_\lambda \log L^\lambda(x_o,x_h)}{L_o^\lambda(x_o)} dR_h(x_h) \\
&= \int_{\Omega_h} \frac{L^\lambda(x_o,x_h)}{L_o^\lambda(x_o)} \nabla_\lambda \log L^\lambda(x_o,x_h) dR_h(x_h) \\
&= \int_{\Omega_h} \frac{dQ_{h|o}^\lambda}{dR_h}(x_h|x_o) \nabla_\lambda \log L^\lambda(x_o,x_h) dR_h(x_h) \\
&= \int_{\Omega_h} \frac{dQ_{h|o}^\lambda}{dS_h^{\lambda,x_o}}(x_h|x_o) \nabla_\lambda \log L^\lambda(x_o,x_h) dS_h^{\lambda,x_o}(x_h) \qquad \text{(A.6)}
\end{aligned}$$

Finalement, pour obtenir l'équation (**??**), nous pouvons approximer l'expression précédente, ce qui nous donne:

$$\nabla_\lambda \log \hat{L}_o^\lambda(x_o) = \sum_{m=1}^{M} p_{h|o}^\lambda(h^m|x_o) \nabla_\lambda \log L^\lambda(x_o^s,h^m), \quad h^m \in \mathcal{H} \qquad \text{(A.7)}$$

Annexe B

Algorithme

Nous présentons dans cette annexe les algotihmes nécessaires à l'entrainement des paramètres et au calcul de la fonction de vraisemblace. Nous noterons

$$\boldsymbol{x}^{m,s}(t) = (\boldsymbol{x}_o^s(t),\boldsymbol{h}^{m,s}(t)) = (x_1^s(t),\ldots,x_D^s(t),h_1^{m,s}(t),\ldots,h_H^{m,s}(t)) \tag{B.1}$$

les valeurs au temps t des N neurones pour l'essai s et le $m^{\text{ième}}$ chemin d'intégration. La variable $\boldsymbol{Z}$ est une variable aléatoire de distribution normale et prend ses valeurs dans $\mathbb{R}^H$.

Pour plus de détails concernant les symboles utilisés, se référer au chapitre **??**.

Fonction de vraisemblance
Entrée: x_o,λ,$\boldsymbol{h}(1)$,σ,θ,M,Δt
Sortie: $\hat{L}_o^{\lambda}(x_o)$

for $m := 1$ **to** M
  $\boldsymbol{h}^m(1) = \boldsymbol{h}(1)$
  for $t := 1$ **to** $\bar{T} - 1$
    $\boldsymbol{h}^m(t+1) = \boldsymbol{h}^m(t) + \boldsymbol{\mu}_h(\boldsymbol{x}^m(t),\lambda) + \sigma\sqrt{\Delta t}\boldsymbol{Z}$
  end
  $w(m) = \exp\left\{\frac{1}{\sigma^2}\sum_{t=1}^{\bar{T}-1}\boldsymbol{\mu}_o(\boldsymbol{x}^m(t),\lambda)\cdot d\boldsymbol{x}_o(t) - \frac{1}{2}\|\boldsymbol{\mu}_o(\boldsymbol{x}^m(t),\lambda)\|^2\Delta t\right\}$
end
$L_o = \log(\frac{1}{M}\sum_{m=1}^{M} w(m))$

Algorithme d'entraînement des poids et des constantes de temps

Entrée: $\mathcal{E} = \{x_o^s\}_{s=1}^S$,$\lambda^{(0)}$,$\boldsymbol{h}(1)$,$\sigma^{(0)}$,$\sigma_{\text{end}}$,$\alpha$,$\delta$,$K$,$\theta$,$M$,$\Delta t$
Sortie: λ

$\sigma = \sigma^{(0)}$
for $k := 1$ **to** K *Itérations*
  for $s := 1$ **to** S *Essais*
    for $m := 1$ **to** M *Chemins dans la couche cachée*
      $\boldsymbol{h}^{m,s}(1) = \boldsymbol{h}(1)$
      for $t := 1$ **to** $\bar{T} - 1$ *Pas de temps*
        $\boldsymbol{h}^{m,s}(t+1) = \boldsymbol{h}^{m,s}(t) + \boldsymbol{\mu}_h(\boldsymbol{x}^{m,s}(t), \lambda^{(k-1)}) + \sigma\sqrt{\Delta t}\boldsymbol{Z}$
      end

$$w^s(m) = \exp\Big\{\frac{1}{\sigma^2}\sum_{t=1}^{\bar{T}-1}\boldsymbol{\mu}_o(\boldsymbol{x}^{m,s}(t),\lambda^{(k-1)})\cdot d\boldsymbol{x}_o(t) - \tfrac{1}{2}\|\boldsymbol{\mu}_o(\boldsymbol{x}^{m,s}(t),\lambda^{(k-1)})\|^2\Delta t\Big\}$$

$$a_{ij}^s(m) = \sum_{t=1}^{\bar{T}-1}\varphi(x_i^{m,s}(t))\varphi(x_j^{m,s}(t))\Delta t \quad \forall i,j = 1,\dots N$$

$$b_{ij}^s(m) = \sum_{t=1}^{\bar{T}-1}\varphi(x_i^{m,s}(t))dx_j^{m,s}(t) + \frac{1}{\tau_j}\sum_{t=1}^{\bar{T}-1}\varphi(x_i^{m,s}(t))x_j^{m,s}(t)\Delta t \quad \forall i,j = 1,\dots N$$

      if $\delta = 1$
        then

$$c_i^s(m) = \sum_{t=1}^{\bar{T}-1}\left(\sum_{j=1}^{N}\varphi(x_j^{m,s}(t))W_{ji}^{(k-1)}x_i^{m,s}(t)\Delta t - x_i^{m,s}(t)dx_i^{m,s}(t)\right) \quad \forall i = 1,\dots N$$

$$d_i^s(m) = \sum_{t=1}^{\bar{T}-1}x_i^{m,s}(t)^2\Delta t \quad \forall i = 1,\dots N$$

      fi
    end

$$\tilde{w}^s(m) = \frac{w^s(m)}{\sum_{m'=1}^{M}w^s(m')}$$

  end

$$\tilde{a} = \sum_{s=1}^{S}\sum_{m=1}^{M}\tilde{w}^s(m)a^s(m);\ \tilde{b} = \sum_{s=1}^{S}\sum_{m=1}^{M}\tilde{w}^s(m)b^s(m)$$

$$\textbf{if } \delta = 1 \textbf{ then } \gamma_i^{(k)} = \frac{\sum_{s=1}^{S}\sum_{m=1}^{M}\tilde{w}^s(m)c_i^s(m)}{\sum_{s=1}^{S}\sum_{m=1}^{M}\tilde{w}^s(m)d_i^s(m)} \textbf{ else } \gamma_i^{(k)} = \gamma_i^{(k-1)} \textbf{ fi} \quad \forall i = 1,\dots N$$

  $W^{(k)} = \tilde{a}^{-1}\tilde{b}$
  if $\sigma > \sigma_{\text{end}}$ **then** $\sigma = \alpha^k\sigma^{(0)}$ **fi**
end

Annexe C

Composantes principales

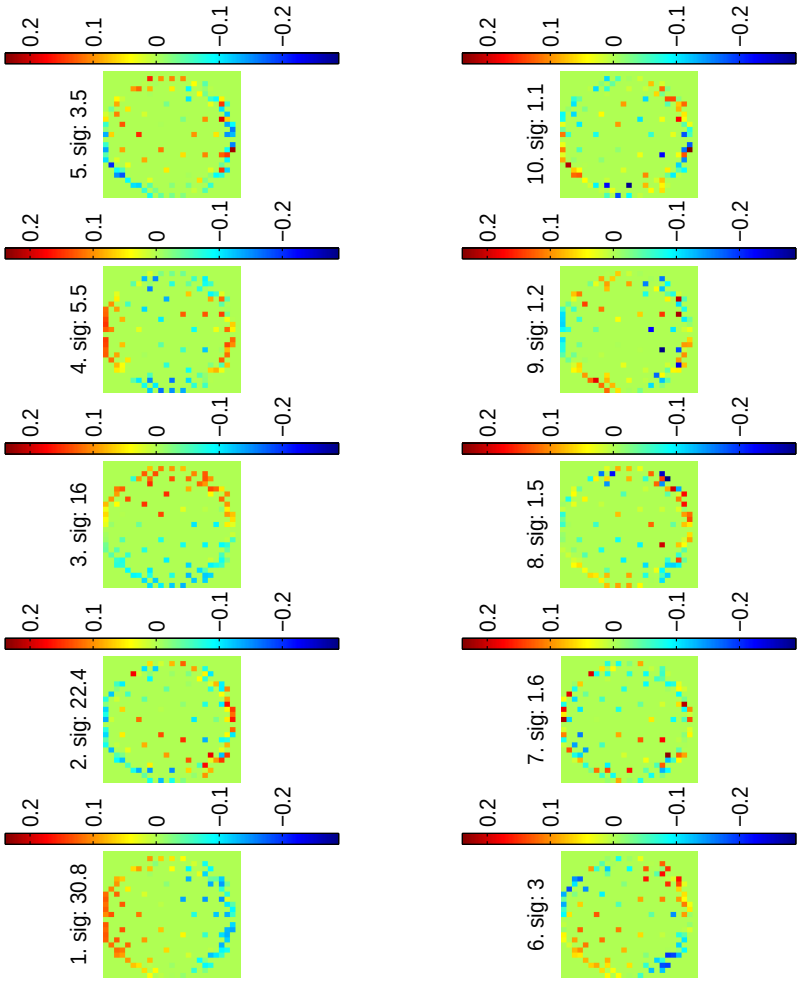

FIG. C.1 – *10 premières composantes principales (la valeur "sig" représente le pourcentage de la variance totale)*

Annexe D

Détails des résultats

ξ_2	C_1	C_2	C_3	C_4
R_1	3	1	0	4
R_2	6	8	15	5
R_3	16	14	11	8
R_4	31	42	33	44

TAB. D.1 – *Classification des 241 essais de test avec la configuration* ξ_2

ξ_3	C_1	C_2	C_3	C_4
R_1	19	12	12	14
R_2	2	3	3	7
R_3	11	19	11	13
R_4	24	31	33	27

TAB. D.2 – *Classification des 241 essais de test avec la configuration* ξ_3

ξ_4	C_1	C_2	C_3	C_4
R_1	2	1	0	3
R_2	18	26	13	17
R_3	18	18	21	20
R_4	12	20	25	21

TAB. D.3 – *Classification des 241 essais de test avec la configuration* ξ_4

	$\hat{\xi}_2$				$\hat{\xi}_3$			
	C_1	C_2	C_3	C_4	C_1	C_2	C_3	C_4
R_1	16	10	5	9	20	11	12	12
R_2	20	21	34	23	14	14	10	27
R_3	6	7	5	9	8	16	13	9
R_4	14	27	15	20	14	24	24	13

TAB. D.4 – *Classification des 241 essais de test avec les configurations ξ_2 et ξ_3 en s'arrêtant à la même valeur de $q(k)$*

	$K_{\mathrm{l}}(\mathbf{x},\mathbf{x}')$				$K_{\mathrm{p}}(\mathbf{x},\mathbf{x}'),d=2$				$K_{\mathrm{g}}(\mathbf{x},\mathbf{x}'),\sigma=0.5$			
	C_1	C_2	C_3	C_4	C_1	C_2	C_3	C_4	C_1	C_2	C_3	C_4
R_1	18	19	8	9	24	19	11	9	12	11	4	6
R_2	24	26	12	9	22	27	13	11	36	44	18	13
R_3	11	12	24	22	8	9	21	20	8	5	15	17
R_4	3	8	15	21	2	10	14	21	0	5	22	25
NSV	310	340	328	305	476	511	503	484	718	718	718	718

TAB. D.5 – *Classification des 241 essais de $\widetilde{\mathcal{E}}$ à l'aide des SVM*

	$K_{\mathrm{l}}(\mathbf{x},\mathbf{x}')$				$K_{\mathrm{p}}(\mathbf{x},\mathbf{x}'),d=2$				$K_{\mathrm{g}}(\mathbf{x},\mathbf{x}'),\sigma=0.5$			
	C_1	C_2	C_3	C_4	C_1	C_2	C_3	C_4	C_1	C_2	C_3	C_4
R_1	21	15	12	5	22	15	8	7	4	2	2	3
R_2	23	32	12	12	24	32	11	8	45	57	33	28
R_3	8	6	25	15	7	8	23	18	5	3	6	11
R_4	4	12	11	29	3	10	17	28	2	3	18	19
NSV	394	413	407	388	519	544	534	528	718	718	718	718

TAB. D.6 – *Classification des 241 essais de $\mathcal{E}$ à l'aide des SVM*