CLASSIFICATION NON LINÉAIRE À L'AIDE DE MÉLANGE DE MODÈLES DISCRIMINATIFS

par

Joyce Elvis Mahoutondji Togban Cokouvi

Thèse présentée au Département d'informatique
en vue de l'obtention du grade de philosophiæ doctor (Ph.D.)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, mai 2021

Le 17 Mai 2021

le jury a accepté la thèse de Monsieur TOGBAN COKOUVI Joyce Elvis Mahoutondji *dans sa version finale.*

Membres du jury

Professeur Djemel Ziou
Directrice / Directeur de recherche
Département d'informatique

Professeur Taoufik Bouezmarni
Membre interne
Département de Mathématique

Professeur Mohand Said Allili
Membre externe
Département d'informatique et d'ingénierie
Université du Québec en Ouataouais

Professeur Shengrui Wang
Président-rapporteur
Département d'informatique

Sommaire

Cette thèse a pour objectif la classification non linéaire à l'aide de mélange hiérarchique de modèles discriminatifs. Dans ce cadre, nous avons proposé un modèle général pour le mélange hiérarchique de modèles discriminatifs. Ce dernier permet de combiner un ensemble de modèles discriminatifs à l'aide de fonctions que nous avons nommées fonctions de sélection. De ce modèle, ont été extraits deux exemples afin de montrer qu'il est possible d'utiliser un nombre réduit de modèles discriminatifs comparé aux précédents modèles. Cette réduction se fait tout en gardant des performances de classification élevées. Ceci est possible en effectuant un choix de fonctions de sélection qui ont permis une répartition efficiente des tâches. Par la suite, nous avons proposé deux modèles discriminatifs pour la classification des données proportionnelles. Ces modèles sont basés sur la distribution de la Dirichlet généralisée. Afin d'estimer convenablement les paramètres de ces deux modèles, nous avons établi une borne supérieure au mélange de Dirichlet généralisée. Des expériences ont montré l'intérêt de ces modèles de même que leurs limites.

Mots-clés: apprentissage machine, mélange hiérarchique de modèles discriminatifs, mélange de Dirichlet généralisée, approximation variationnelle.

Remerciements

Je tiens à exprimer ma profonde gratitude au professeur Djemel Ziou pour la confiance qu'il m'a témoignée, son encadrement et son suivi ainsi que sa grande patience et ses qualités humaines. Son rôle a été capital dans l'accomplissement de ce travail. Je remercie également les membres du jury d'avoir pris le temps de lire et d'évaluer cette thèse. Je voudrais également remercier mes collègues du centre MOIVRE (MOdélisation en Imagerie, Vision et REseaux de neurones) pour le temps passé ensemble et les nombreuses discussions.

Je tiens à remercier mes parents et plus particulièrement, à titre posthume, mon frère et ma sœur qui sont partis trop tôt sans avoir vu la fin de ce travail. J'adresse également mes sincères remerciements à tous mes amis qui sont devenus, le long du chemin, des frères et des sœurs.

Abréviations

HME Mélange hiérarchique d'experts (*Hierarchical Mixture of Experts* en anglais)

LME Mélange local d'experts (*Localized mixture of experts* en anglais)

HMD Modèle général du mélange hiérarchique de classifieurs discriminatifs

HMD1 Premier exemple issu du modèle général HMD

HMD2 Deuxième exemple issu du modèle général HMD

DGD Modèle discriminatif basé sur la Dirichlet généralisée (*Discriminative Generalized Dirichlet* en anglais)

HMGD Mélange hiérarchique basé sur la Dirichlet généralisée (*Hierarchical mixture of Discriminative Generalized Dirichlet* en anglais)

Table des matières

Table des matières

Liste des figures

Liste des figures

Liste des tableaux

Liste des algorithmes

Introduction

La classification est l'une des tâches que les êtres humains de même que les machines exécutent le plus en matière de prise de décision. Elle consiste à affecter un objet, en se basant sur ses caractéristiques, à un groupe ou à une classe. Ces caractéristiques peuvent être récupérées à partir de capteurs ou de collections existantes. Elles peuvent être sous diverses formes : texte, son, image et vidéo [97, 7]. La classification est utilisée dans de nombreuses applications telles que la transmission [39], la compression [101], la recherche par le contenu [57], la reconnaissance d'objets [35] et la détection de fraudes [17]. Lorsque la classification est exécutée par une machine, elle peut être mise en œuvre par un modèle génératif ou un modèle discriminatif. Un modèle génératif a pour objectif de trouver le processus ou la distribution ayant généré les données. Dans ce cas, la classification n'est qu'une étape secondaire [85]. L'efficacité de ce type de modèle, en ce qui concerne la classification, dépend de la pertinence de l'hypothèse émise sur la distribution des données. En contraste, un modèle discriminatif détermine directement le contour des classes sans chercher à comprendre comment ces dernières sont générées [93]. Ce fonctionnement est plus naturel, car il s'agit de traiter directement le problème principal sans passer par la résolution d'un problème plus large. Certains travaux ont combiné les modèles discriminatifs et génératifs afin de tirer le meilleur des deux approches [19, 14, 89]. Cette thèse se limite aux modèles discriminatifs, étant donné qu'ils font de meilleures prédictions [98]. Ce choix étant fait, nous nous limitons ainsi aux données étiquetées.

Lorsque l'unique frontière de décision que peut produire un modèle de classification est un hyperplan, il est désigné comme étant un modèle linéaire. Durant les deux derniers siècles, de nombreux travaux ont été réalisés sur ce type de modèles et plusieurs applications en sont la résultante. Cependant, lorsque les classes ne peuvent être sé-

parées par des hyperplans, la classification devient un peu plus difficile. Une approche possible pour aller au-delà de ces limitations est de décomposer un problème complexe en plusieurs tâches simples. Ceci est la façon dont la plupart des tâches complexes sont traitées par les organismes vivants : les objets complexes sont reconnus par le cerveau comme une combinaison d'objets simples [130] ; une compagnie généralement divise un projet en plusieurs champs d'expertise et la détection de fraude à l'échelle nationale est faite en traitant les informations par régions homogènes. Ce processus peut être hiérarchisé, une tâche est divisée en plusieurs sous-tâches, ces dernières sont à leur tour divisées et ainsi de suite. Vu que ce sont des données qui sont segmentées selon ce paradigme, nous utiliserons le mot «mélange» pour désigner ces tâches et les relations qui les lient. Autant dans la littérature des modèles génératifs [12] que discriminatifs [74], des modèles de mélange hiérarchique ont été proposés afin de modéliser des tâches complexes. Plus précisément dans le contexte de l'apprentissage discriminatif, le mélange hiérarchique d'experts (HME) a été introduit par Jordan et Jacobs [74]. L'idée de base derrière le modèle HME est la division récursive d'un problème complexe en un ensemble de sous-problèmes identifiés par des régions. La délimitation spatiale de chaque région est formée par des hyperplans décrits par une fonction. Cette dernière indique le degré d'appartenance d'une instance de données à une région. Le modèle HME suit le même procédé que les algorithmes basés sur les arbres de décision en découpant de façon récursive l'espace des données en plusieurs régions et sous-régions. Cependant, les arbres de décision tels que CART [32] et C 4.5 [106] effectuent un découpage rigide de l'espace. Au contraire, le modèle HME découpe de façon flexible l'espace des données à l'aide d'un ensemble d'hyperplans et les feuilles ici sont des modèles discriminatifs agissant comme des «experts». Les experts ici sont modélisés comme étant des régressions logistiques. Un modèle semblable au HME, mais dépourvu d'une structure hiérarchique a été également introduit par Takagi et Sugeno pour le contrôle de systèmes non linéaires [125] et est connu sous le nom de multi-modèle. HME peut être aussi vu comme une manière de combiner des classifieurs. Il existe également d'autres méthodes de combinaisons de classifieurs [66, 36, 134]. Leur but est d'entraîner chaque classifieur sur une portion des données ou sur un sous-espace des caractéristiques. Le résultat final est obtenu en utilisant différentes stratégies de combinaison comme le vote ou la pondération.

Introduction

La capacité de généralisation d'un modèle (faible erreur de prédiction) dépend entre autre de sa complexité et des données d'apprentissage. Afin d'augmenter la capacité du modèle HME à généraliser, il est possible d'augmenter le nombre d'experts ou d'utiliser des techniques de régularisation [1, 2, 69]. Cependant d'une part, l'augmentation du nombre d'experts entraîne un nombre de paramètres élevés à estimer, une augmentation du temps d'apprentissage, une consommation accrue d'énergie et une augmentation de l'espace nécessaire pour le stockage du modèle. D'autre part, quoique les techniques de régularisation permettent d'imposer des contraintes sur les paramètres, elles ne réduisent pas pour autant la complexité du modèle. Réduire la complexité du modèle HME, peut se faire en utilisant peu d'experts.

Dans cette thèse nous abordons deux problèmes liés à la classification selon le paradigme du modèle HME : la réduction du nombre de classifieurs utilisés et le mélange de classifieurs dans le cadre de la classification de données à support compact. Pour le premier problème, il faudrait utiliser peu d'experts tout en garantissant une bonne généralisation. Pour le second problème, rappelons que le modèle HME a été proposé pour des données à support non compact. Élaborer un modèle hiérarchique pour la classification de données à support compact pourrait permettre d'améliorer les prédictions et de permettre également de réduire le nombre d'experts utilisés.

Dans le premier chapitre, nous présentons différentes méthodes utilisées dans la littérature pour effectuer la classification non linéaire des données. Dans un premier temps, nous présentons l'astuce du noyau suivi de quelques méthodes appartenant à la famille des réseaux de neurones. Par la suite, différentes méthodes de combinaison de classifieurs sont également.

Dans le deuxième chapitre, nous proposons un modèle général pour le mélange hiérarchique de classifieurs (HMD). En nous basant sur ce modèle général, nous répondons à la question suivante : comment réduire le nombre de classifieurs utilisés tout en ayant un taux élevé de bonnes prédictions ? Notre hypothèse est qu'une réduction du nombre de classifieurs passe par une répartition efficace des tâches entre eux. Nous validons cette hypothèse à travers deux exemples (HMD1 et HMD2) issus du modèle HMD.

Dans le troisième chapitre , nous nous intéressons à la classification supervisée des données proportionnelles. Ces dernières sont multidimensionnelles par nature, posi-

tives et sont définies sur un support compact. Nous présentons un classifieur basé sur la distribution de Dirichlet généralisée (DGD). À l'aide de ce classifieur et du modèle HMD, nous proposons un nouveau modèle de mélange hiérarchique (HMGD). L'estimation des paramètres des modèles DGD et HMGD à l'aide d'une approximation variationnelle est ensuite présentée. Les modèles DGD et HMGD ont été validés à l'aide de plusieurs expériences telles que la détection de courriers indésirables et l'identification de l'espace de couleur d'une image.

Pour finir, nous présentons la conclusion et les perspectives liées à cette thèse.

Chapitre 1

Classification non linéaire

Afin d'effectuer une tâche de classification, un classifieur apprend à affecter des exemples x aux étiquettes y qui leur sont associées. Ces exemples sont représentés par des vecteurs de dimension D : $x \in \mathbb{R}^D$ et $y \in \{1 \cdots C\}$ où C est le nombre de classes ou groupes. Les exemples x sont des caractéristiques décrivant un objet ou un phénomène observé et sont généralement extraits avant la classification. Chaque dimension de x représente une caractéristique. L'apprentissage se fait à partir d'un ensemble de données étiquetées $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$. Cet ensemble est appelé ensemble d'apprentissage et contient N exemples. Le n-ème exemple (observation) est nommé x_n et y_n est son étiquette. L'ensemble des exemples et des étiquettes sont respectivement représentés par la matrice $\mathbf{X} \in \mathbb{R}^{D \times N}$ et le vecteur $\mathbf{Y} \in \mathbb{R}^N$. Une fois entraîné, le classifieur est évalué à l'aide des prédictions qu'il effectue sur un ensemble de tests $\mathcal{D}_t$. Cet ensemble est composé de nouveaux exemples n'appartenant pas à l'ensemble d'apprentissage. Parfois, un ensemble de validation $\mathcal{D}_v$ est également utilisé pour sélectionner la meilleure configuration du classifieur (ex. nombre de couches cachées pour les réseaux de neurones). La classification se déroule donc souvent en deux phases :

- Phase d'apprentissage (ou d'entraînement) : Dans cette phase, un classifieur est construit à partir de l'ensemble d'apprentissage.
- Phase de test (ou de prédiction) : Dans cette phase, le classifieur construit est

utilisé pour prédire la classe $\hat{y}$ d'une nouvelle instance non étiquetée.

La capacité d'un classifieur à correctement prédire les classes pour de nouvelles observations est appelée la généralisation. On parle de surapprentissage lorsqu'un classifieur affecte correctement les classes aux données d'apprentissage, mais n'a pas une bonne généralisation. Nous distinguons deux types de classifieurs à savoir les classifieurs linéaires et les classifieurs non linéaires. Lorsqu'un classifieur ne produit que des hyperplans comme frontière de décision, il est qualifié de linéaire. Ce type de classifieur est rapide à entraîner, facile à implémenter et ses prédictions sont simples à expliquer. En général, face à la complexité des tâches de classification, la frontière de décision est arbitraire (autre qu'un hyperplan). Dans ce cas, un classifieur non linéaire est utilisé. Il est possible de construire ce type de classifieur en se basant sur la décomposition des classes de données [5, 9]. Dans un premier temps, chaque classe est décomposé en sous-classes. Ensuite, un classifieur multi-classe est entraîné sur les données re-étiquetées. D'autres méthodes telles que les arbres de décision et les plus proches voisins (Ppv) peuvent être aussi utilisées pour la classification non linéaire [82, 139]. Dans ce chapitre, nous présentons trois principales catégories de méthodes pour la classification non linéaire à savoir la méthode du noyau, les Réseaux de neurones artificiels et la pondération de classifieur(s).

1.1 Méthode du noyau

L'idée de base de la méthode du noyau est de transformer les données d'entrée x_n en des données ayant de très grandes dimensions. Cette transformation se fait par une fonction $\Phi : \mathbb{R}^D \to \mathbb{R}^f$ où $f > D$. L'espace $\mathbb{R}^f$ est appelé espace transformé. Une fois les données décrites dans l'espace transformé, elles peuvent être utilisées avec des techniques d'analyse des données tels que la classification, la réduction de dimensions ou l'apprentissage non supervisé [119]. En ce qui concerne la classification, elle est supposée plus aisée dans le nouvel espace. À titre d'illustration, soit le cas où $D = 2$, $f = 3$ et la fonction Φ définie par :

$$\begin{array}{rccl} \Phi : & \mathbb{R}^2 & \longmapsto & \mathbb{R}^3 \\ & (r_1, r_2) & \longmapsto & (z_1,\, z_2,\, z_3) = \left(r_1^2,\, \sqrt{2} r_1 r_2,\, r_2^2\right) \end{array} . \tag{1.1}$$

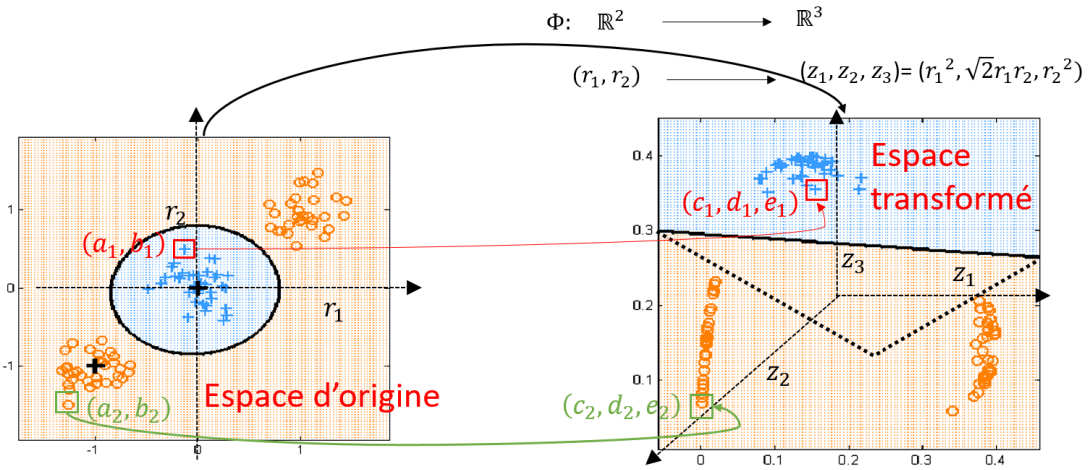


figure 1.1 – Illustration de la méthode du noyau à l'aide d'une fonction Φ connue.

La figure 1.1 présente un problème de classification où les données doivent être réparties en deux classes («o» et «+»). Dans l'espace transformé, la frontière de décision (tracé en noir) est un hyperplan ce qui correspond à une forme circulaire dans l'espace original.

En général, les coordonnées de x_n dans l'espace transformé ne sont jamais calculées, car il n'est pas toujours aisé de trouver une fonction Φ appropriée. Par conséquent, pour mettre en œuvre la méthode du noyau, une fonction noyau $k_f : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$ est introduite où $k_f(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$. La notation $\langle , \rangle$ représente le produit scalaire. La fonction k_f est une mesure de similarité entre paires d'observations décrites dans l'espace transformé. Il est intéressant de noter que la plupart des propriétés géométriques des instances de données sont contenues dans le produit scalaire. En effet, nous avons dans l'espace transformé :

- la distance entre deux instances de données : $\|\Phi(x) - \Phi(x')\|^2 = k_f(x, x) - 2k_f(x, x') + k_f(x', x')$,
- la distance par rapport à l'origine : $\|\Phi(x)\|^2 = k_f(x, x)$,
- l'angle entre deux instances : $\cos(\theta) = \frac{k_f(x,x')}{\sqrt{k_f(x,x)}\sqrt{k_f(x',x')}}$.

Il n'est pas nécessaire de connaître la fonction Φ pour définir l'expression de la fonction k_f. Cette approche est connue sous l'appellation astuce du noyau (*kernel trick* en anglais). Les fonctions suivantes sont souvent utilisées pour le calcul du noyau [119] :

1. Noyau linéaire : $k_f(x_i, x_j) = \langle x_i, x_j \rangle$;
2. Noyau gaussien (RBF) : $k_f(x_i, x_j) = \exp\left(-\alpha \|x_i - x_j\|_2^2\right)$ où $\alpha > 0$ est la largeur du noyau.
3. Noyau polynomial : $k_f(x_i, x_j) = (\langle x_i, x_j \rangle + \alpha)^d$ où $d \in \mathbb{N}$ est le degré du polynôme. Le paramètre $\alpha \geq 0$ sert à équilibrer l'influence des termes d'ordre supérieur par rapport aux termes d'ordre inférieur au sein du polynôme. En augmentant la valeur de α, on diminue l'influence des termes d'ordre supérieur.
4. Noyau Dirichlet : $k_f(x_i, x_j) = \frac{\sin[(N+1/2)(x_i - x_j)]}{\sin\left(\frac{x_i - x_j}{2}\right)}$.

Le choix du noyau est la plupart du temps effectué *a priori* sans aucune analyse des données [102]. Pour un problème de classification, la fonction k_f doit être évaluée pour toutes les paires de données de l'ensemble d'apprentissage. Pour N instances d'apprentissage, nous obtenons la matrice suivante :

$$\mathbf{K} = \begin{bmatrix} k_f(x_1, x_1) & k_f(x_1, x_2) & \cdots & k_f(x_1, x_N) \\ k_f(x_2, x_1) & k_f(x_2, x_2) & \cdots & k_f(x_2, x_N) \\ \cdots & \cdots & \cdots & \cdots \\ k_f(x_N, x_1) & k_f(x_N, x_2) & \cdots & k_f(x_N, x_N) \end{bmatrix}. \tag{1.2}$$

En utilisant l'astuce du noyau, chaque instance d'apprentissage est donc décrite dans un espace de dimensions N. D'après le théorème de Mercer [119], pour que la fonction k_f soit un noyau valide, il faut que la matrice $\mathbf{K}$ soit semi-définie positive et symétrique. Une fonction k_f valide implique qu'il existe une fonction Φ dont l'ensemble d'arrivée correspond à l'espace transformé $\mathbb{R}^f$.

N'importe quel classifieur (linéaire ou non) peut apprendre à partir de la matrice $\mathbf{K}$ et des étiquettes $\mathbf{Y}$. Cependant, le séparateur à vastes marges (SVM) est le classifieur le plus utilisé en combinaison avec la méthode du noyau [119, 16, 21, 22, 29]. Ce choix peut être justifié par le faible coût de calcul de la matrice $\mathbf{K}$ [18]. En effet, le SVM [131] cherche l'hyperplan qui maximise la marge entre les classes. La marge ici est la

distance entre l'hyperplan de séparation et les instances les plus proches. Les instances situées sur la marge (les instances de support) sont les seules qui interviennent dans la classification. Ce faisant, la taille de la matrice du noyau $\mathbf{K}$ est réduite (la taille de la matrice change de N au nombre d'instances support). Par contre, ce nombre augmente avec la taille des données d'apprentissage. Dans le même ordre que le SVM avec noyau, Zhu *et al.* [142] ont proposé une méthode appelée *import vector machine* (IVM) et l'ont employé en combinaison avec la régression logistique. Cette méthode donne une matrice du noyau de taille inférieure à celle obtenue avec le SVM. La décomposition spectrale est également appliquée pour réduire le coût lié à l'utilisation de la matrice du noyau [117]. Cependant, cette décomposition peut amener à une perte du pouvoir discriminatoire de la matrice $\mathbf{K}$ [107]. Lorsque les données proviennent de différentes sources, l'utilisation d'un seul noyau peut être inefficace. Afin d'aller au-delà de cette limite, la combinaison de noyaux a été introduite (une fonction noyau par source) [60, 79]. Ainsi, différents noyaux peuvent être combinés linéairement ou non. Pour plus de détails sur la combinaison des noyaux, le lecteur peut se référer à la revue de littérature proposée par Motai *et al.* [96].

En résumé, la méthode du noyau est une manière d'aborder le problème de la classification non linéaire de données. L'approche la plus utilisée est l'astuce du noyau qui ne nécessite pas le calcul explicite des coordonnées dans l'espace transformé. Cependant, l'utilisation de l'astuce du noyau est dépendent du nombre de données d'apprentissage. Ceci conduit à un calcul excessif lorsque nous sommes en présence d'une grande collection de données [119, 79]. Un autre inconvénient de l'astuce du noyau est le besoin de toutes les données d'apprentissage pour la prédiction de la classe d'une nouvelle instance. Par exemple, une nouvelle instance x_{new} est décrite par le vecteur $\left(k_f\left(x_{new}, x_1\right), \cdots, \cdots, k_f\left(x_{new}, x_N\right)\right)^T$. De plus, l'utilisation de l'astuce du noyau rend difficile la compréhension de l'impact qu'ont les caractéristiques de départ sur la décision du classifieur [10].

1.2 Réseaux de neurones artificiels (RNA)

Dans la section précédente, nous avons présenté la méthode du noyau. Cette dernière transforme les données d'entrée en des données ayant de très grandes dimensions. Dans cette section, nous présentons les réseaux de neurones artificiels (RNA) utilisés avec succès pour la classification non linéaire des données [67, 112, 63, 33, 62]. Dans la formulation des RNA, la dimension des données d'entrée reste fixe. Les RNA ont en général, une structure stratifiée qui peut être décomposée en trois parties :

- une couche d'entrée incorporant les données ;
- une ou plusieurs couche(s) cachée(s) souvent constituée(s) d'un ensemble de fonctions d'activation. Ces dernières capturent la structure non linéaire des données ;
- une couche de sortie qui donne le résultat désiré (étiquette ou probabilité).

Les RNA sont aussi vus comme un ensemble interconnecté d'unités appelées neurones. Lorsque plusieurs couches cachées sont utilisées, on parle de RNA profonds [13]. Plusieurs types de RNA peuvent être appliqués pour la classification des données [4]. Nous présentons ici deux des plus utilisés et étudiés : le perceptron multicouche (MLP) et les réseaux de neurones convolutionnel (CNN).

Le classifieur MLP possède généralement une seule couche cachée (Fig. 1.2) et est décrit par les équations suivantes [16] :

$$\begin{aligned} y_c\left(x_n, \mathbf{\Omega}\right) &= f\left(\sum_{j=1}^{M} \Omega_{cj}^{(2)} z_j + \Omega_{c0}^{(2)}\right) \\ z_j &= h\left(\sum_{i=1}^{D} \Omega_{ji}^{(1)} x_{n,i} + \Omega_{j0}^{(1)}\right) \end{aligned}, \tag{1.3}$$

où $\Omega_{ji}^{(1)}$ et $\Omega_{j0}^{(1)}$ sont respectivement les poids et les biais liant la première et la seconde couche. Les poids $\Omega_{cj}^{(2)}$ et les biais $\Omega_{c0}^{(2)}$ lient la couche cachée à celle de sortie. L'ensemble des paramètres est désigné par $\mathbf{\Omega} = \left\{\Omega^{(1)}, \Omega^{(2)}\right\}$ où $\Omega^{(1)} = \left\{\Omega_{ji}^{(1)}\right\}$ et $\Omega^{(2)} = \left\{\Omega_{cj}^{(2)}\right\}$ avec $j = 1 \cdots M$, $i = 1 \cdots D$ et $c = 1 \cdots C$. La $i^{ème}$ composante de l'instance x_n est $x_{n,i}$. Dans la littérature, $h\left(.\right)$ est appelé fonction d'activation et est souvent une sigmoïde. Elle sert à activer les unités cachées. Chaque unité cachée reçoit une somme pondérée des sorties de la couche précédente. La couche de sortie suit le

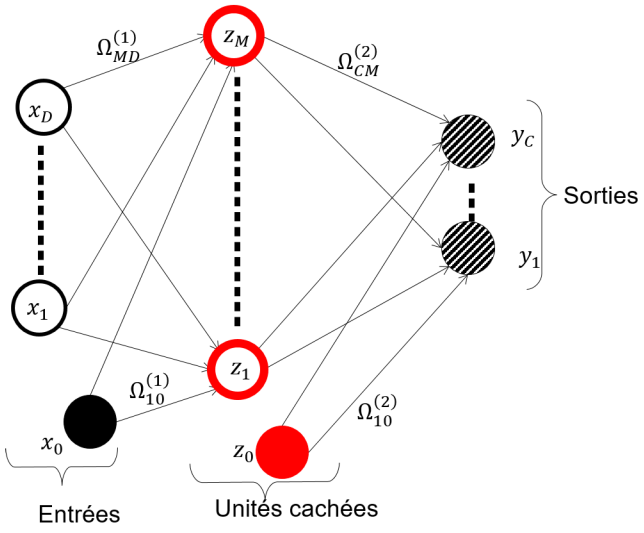


figure 1.2 – Architecture du MLP. Les cercles représentent les neurones (unités) et les arcs les connections entres ces derniers. Les cercles en rouge sont les unités de la couche cachée et les cercles pleins sont des biais. Les unités de la couche de sortie sont représentées par les cercles hachurés.

même principe que les couches précédentes à l'exception de la fonction d'activation qui est notée ici f. Dans le cas de la classification, la fonction $f(.)$ est choisie comme étant une softmax :

$$\frac{\exp(\theta_c)}{\sum_{l=1}^{C} \exp(\theta_l)}, \tag{1.4}$$

où C est le nombre de classes, c désigne une classe donnée et $\theta_c = \sum_{j=1}^{M} \Omega^{(2)}_{cj} z_j + \Omega^{(2)}_{c0}$. L'algorithme standard pour l'apprentissage du classfieur MLP est la rétropropagation (RP). Cependant, l'algorithme RP converge lentement et a besoin d'une grande quantité de données afin d'obtenir de bons scores. Plusieurs travaux se sont penchés sur l'amélioration des performances de l'algorithme RP [99, 132, 34, 67]. D'autres travaux ont changé à la fois la fonction d'action d'activation h et la procédure d'apprentissage [122, 72, 140]. Par exemple, lorsque la fonction d'activation utilisée dans la couche cachée est une fonction noyau RBF (voir section précédente), on obtient le réseau RBF (RBFN) [72]. Par contre pour le réseau RBF, la fonction f est choisie

comme la fonction identité.

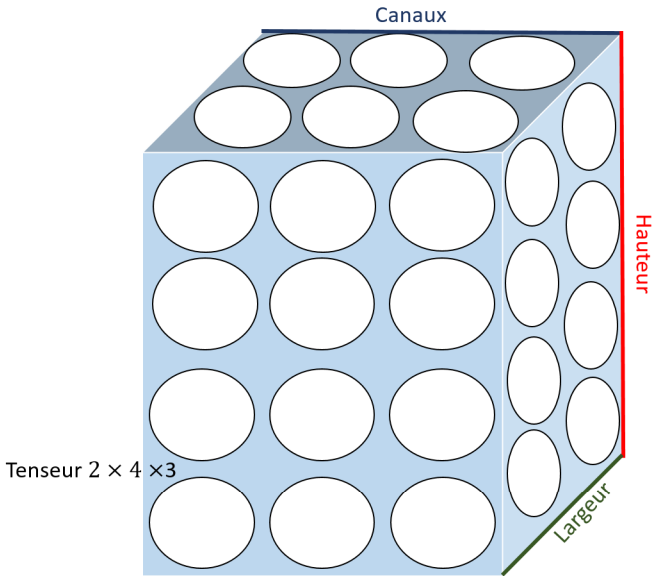


figure 1.3 – Organisation des neurones au niveau des couches d'un réseau CNN.

Les réseaux de neurones convolutionnels (CNN) sont un sous-groupe des RNA profonds. Ils sont utilisés dans de nombreuses applications notamment le traitement d'images [124, 50, 121, 40] et le traitement de documents [80, 62]. Dans un réseau CNN, les neurones sont organisés au niveau de chaque couche sous forme d'un volume (voir 1.3). Chaque couche est donc représentée par un tenseur d'ordre trois. Par exemple, si l'entrée du réseau CNN est une image couleur, chaque pixel est associé à un neurone. La couche d'entrée dans ce cas est représentée par un tenseur à trois dimensions de forme [largeur, hauteur, nombre de canaux]. Les différents types de couches utilisés sont : couche de convolution, couche de réduction de dimensions (*pooling layer*) et couche entièrement connectée (*Fully Connected Layer*). Ces couches peuvent être empilées les unes après les autres. La figure 1.4 illustre l'architecture d'un réseau CNN nommé VGG16 [120]. Au niveau de la couche de convolution, on effectue un produit de convolution entre la sortie de la couche précédente et un filtre de taille prédéfinie. Les coefficients du filtre sont déterminés à travers l'algorithme d'apprentissage. Généralement, une fonction d'activation non linéaire nommée RELU (*Rectified Linear Unit*) est appliquée au résultat de la convolution. La fonction RELU

est définie comme suit :

$$RELU(z) = max(0, z),$$

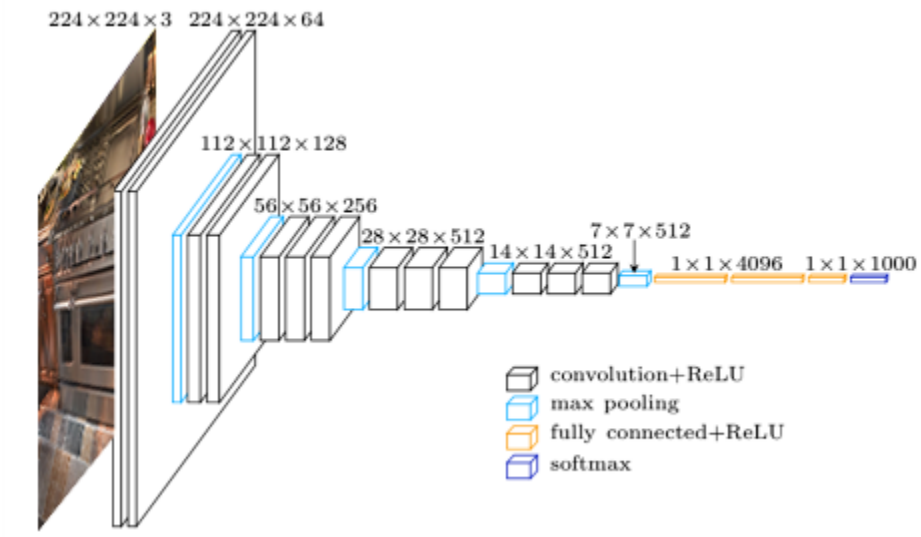


figure 1.4 – Architecture d'un réseau de neurones convolutionnel.

où z est la sortie d'une couche de convolution. La fonction RELU a une sortie linéaire si l'entrée est positive et nulle si non. Elle est utilisée à la place de la sigmoïde afin d'accélérer l'apprentissage et de réduire la saturation des neurones. En effet, avec la sigmoïde comme fonction d'activation, le gradient est presque nul au fur et à mesure qu'on s'éloigne de la couche de sortie. Ce problème est connu sous le nom de «disparition du gradient» et provoque la lenteur de l'apprentissage ou son arrêt [103]. Différentes stratégies de *pooling* sont utilisées afin de réduire la dimension des sorties tout en préservant les caractéristiques pertinentes [115]. Le *max pooling* est la stratégie la plus utilisée et consiste à sélectionner la valeur maximale dans une fenêtre de *pooling*. Elle s'est avérée empiriquement plus performante que les autres stratégies [115, 123].

L'un des principaux avantages de l'utilisation des RNA est le fait que la phase d'extraction des caractéristiques peut se faire au travers des couches du réseau. En particulier dans le cas des réseaux profonds, l'utilisateur n'a pas besoin de définir une procédure d'extraction de caractéristiques avant d'effectuer la classification. Par

contre, il y a un nombre considérable de paramètres à estimer. Par exemple, le réseau Alexnet [77] contient 60 millions de paramètres à estimer tandis que VGG16 en a 138 millions. Néanmoins, plusieurs travaux ont entrepris de compresser les RNA profonds (CNN en particulier) afin de réduire leur taille. Plusieurs méthodes de compression des RNA sont présentées par Cheng *et al.* [41]. Néanmoins, malgré leurs avantages, ils présentent un certain nombre d'inconvénients. En général, la forte interconnexion entre les unités d'un RNA conduit à un modèle à boîte noire (*black box*). En effet, il est presque impossible d'établir une relation de cause à effet concernant une décision prise par un RNA. Lorsqu'une erreur inattendue survient, il est souvent difficile de comprendre comment cette dernière est apparue. Pour des décisions importantes, le concepteur d'un RNA a très peu de connaissances théoriques disponibles. Il existe des tentatives d'interprétation des RNA, mais elles sont peu résistantes à une légère perturbation des données d'entrée [58]. De plus, les RNA ont besoin d'une quantité massive de données ce qui augmente considérablement le temps d'apprentissage. En général, un ensemble de validation est également utilisé afin de trouver la configuration appropriée pour les RNA et de réduire le risque de surapprentissage.

1.3 Pondération de classifieur(s)

Les classifieurs linéaires sont construits pour assigner des étiquettes aux données en se basant sur une fonction discriminative linéaire. Malheureusement, dans plusieurs cas, les données ne peuvent pas être discriminées par des hyperplans. Dans la première section, nous avons brièvement présenté la méthode du noyau qui applique une transformation sur les données. Nous avons également présenté les avantages et limites d'une telle méthode. La même démarche a été adoptée dans la seconde section en ce qui concerne les RNA. Le principal inconvénient des RNA est leur fonctionnement en boîte noire. Ainsi, il est difficile d'établir un lien entre les caractéristiques de départ et les décisions prises par les RNA. Dans cette section, nous présentons une autre manière de faire de la classification non linéaire à savoir la pondération de classifieur(s). Cette dernière ne nécessite pas de décrire les données dans un espace de très grandes dimensions comme c'est le cas avec la méthode du noyau. De plus, elle offre une relative facilité explicative comparée aux RNA. En effet, en utilisant

des classifieurs linéaires à travers les différents schémas de pondération, nous pouvons expliquer localement les prédictions effectuées.

1.3.1 Pondération d'un classifieur

L'idée est d'entraîner un classifieur $\mathcal{C}\ell$ (souvent linéaire) à l'aide de différents sous-ensembles d'apprentissage. L'apprentissage est différé jusqu'à l'arrivée d'une nouvelle instance x_q (instance requête) pour laquelle il faut prédire la classe. Le sous-ensemble d'apprentissage est sélectionné dans le voisinage de l'instance requête à l'aide d'une fonction de pondération $\pi(x_n)$. Cette dernière décrit l'importance de l'instance d'apprentissage x_n. Cette pondération est faible lorsque l'instance requête x_q est éloignée de x_n. Une fois que tous les poids des instances d'apprentissage sont calculés, une version pondérée du classifieur $\mathcal{C}\ell$ est entraînée. Cette classification est non linéaire, car les poids $\{\pi(x_n)\}_{n=1}^{N}$ changent avec chaque nouvelle instance x_q. Cette manière d'apprendre est appelée apprentissage pondéré local ou *Locally Weighted Learning* (LWL) en anglais. Deng [47, chap 4] a utilisé une régression logistique comme classifieur et la fonction $\pi(x_n)$ est basée sur la distance euclidienne. Frank *et al.* [54] ont dans un premier temps trouvé les K plus proches voisins de l'instance requête x_q. Ensuite, les poids sont calculés comme suit :

$$\pi(x_n) = max\left(0, 1 - \frac{d_n}{d_K}\right) = \begin{cases} 1 - \frac{d_n}{d_K} & si\ d_n < d_K \\ 0 & si\ non \end{cases},$$

où d_K est la distance euclidienne entre x_q et son $K-$ème plus proche voisin et d_n la distance euclidienne entre x_q et l'instance x_n. Un classifieur Bayes naïf pondéré est construit par la suite à l'aide des poids calculés. Une approche similaire a été proposée par Bevilacqua *et al.* [15]. Par contre, le classifieur utilisé est l'analyse discriminante basée sur les moindres carrées partielles ou *partial least squares-discriminant analysis* en anglais (PLS-DA). L'avantage principal de l'apprentissage pondéré local est le fait que la construction du classifieur prend en compte l'instance requête [3]. Par contre, un algorithme complexe d'apprentissage augmentera le temps de traitement d'une requête.

1.3.2 Pondération de plusieurs classifieurs

Les approches basées sur l'utilisation de plusieurs classifieurs sont connus sous le nom de méthodes d'ensemble [134, 43]. Ces dernières sont généralement utilisées pour résoudre une tâche complexe de classification à partir d'un ensemble de classifieurs «faibles». Un classifieur est qualifié de faible lorsqu'il obtient une grande erreur de prédiction pour une tâche de classification donnée. Nous parlons de tâche complexe de classification, lorsqu'un classifieur linéaire ne peut être utilisé pour séparer les données. Soit $\mathcal{E} = \{\mathcal{C}\ell_1, \cdots, \mathcal{C}\ell_K\}$ un ensemble de K classifieurs et l'ensemble d'apprentissage $\mathcal{D} = \{\mathcal{D}_1, \cdots, \mathcal{D}_K\}$ où $\mathcal{D}_i$ est un sous-ensemble d'apprentissage. L'objectif est d'obtenir la sortie de l'ensemble $\mathcal{E}$ en combinant par pondération la sortie des classifieurs $\mathcal{C}\ell_i$. Ces derniers peuvent avoir pour sortie un ensemble de probabilités $p(y|\mathcal{C}\ell_i, x) = \{p(y = 1|\mathcal{C}\ell_i, x), \cdots, p(y = C|\mathcal{C}\ell_i, x)\}$ ou une étiquette $\hat{y}^{(i)}$. Il est possible d'obtenir l'étiquette $\hat{y}^{(i)}$ à partir de $p(y|\mathcal{C}\ell_i, x)$:

$$\hat{y}^{(i)} = argmax_{c\in\{1,\cdots,C\}}\, p(y = c|\mathcal{C}\ell_i, x).$$

Lorsque les sorties des classifieurs $\mathcal{C}\ell_i$ sont des étiquettes, leur combinaison est donnée par :

$$\hat{y}^{(\mathcal{E})} = argmax_{c\in\{1,\cdots,C\}} \sum_{i=1}^{K} \pi_i \mathbb{1}_{(c=\hat{y}^{(i)})}, \tag{1.5}$$

où π_i est un poids donnant l'importance relative du classifieur $\mathcal{C}\ell_i$ et $\mathbb{1}_{(c=\hat{y}^{(i)})}$ est égale à 1 si $\hat{y}^{(i)} = c$ et 0 sinon. Lorsque les sorties sont des probabilités, nous avons :

$$p(y = c|\mathcal{E}, x) \propto \sum_{i=1}^{K} \pi_i\, p(y = c|\mathcal{C}\ell_i, x). \tag{1.6}$$

Selon la définition de π_i nous pouvons dégager les règles de combinaisons suivantes :

1. Vote de la majorité : $\pi_i = 1$ pour $i = 1 \cdots, K$
2. Vote de la moyenne : $\pi_i = 1/K$ pour $i = 1 \cdots, K$
3. Vote pondéré : π_i est défini à partir de l'estimation du score de classification du classifieur $\mathcal{C}\ell_i$
4. Sélection : seul un nombre $K^* < K$ de poids π_i ont une valeur non nulle. Dans ce cas, l'estimation des poids se fait durant la phase de test.

Dans une méthode d'ensemble, il est nécessaire que les classifieurs $\mathcal{C}\ell_i$ soit diversifiés [43]. En d'autres termes, chaque classifieur de l'ensemble doit avoir des sorties globalement différentes de celles des autres. Cette diversification peut se faire en utilisant soit :

1. différents types de classifieurs (ex. régression logistique, SVM et MLP).
2. le même type de classifieurs, mais avec différentes configurations (ex. SVM avec différents noyaux ou MLP ayant différents nombres d'unités cachées).
3. le même type de classifieurs mais entraîné sur différents sous-ensembles d'apprentissage $\mathcal{D}_i$ pour $i = 1, \cdots, K$.

Ces trois stratégies de diversification des classifieurs peuvent être combinées.

Ensemble hétérogène de classifieurs

L'ensemble $\mathcal{E}$ est dit hétérogène lorsque les classifieurs $\mathcal{C}\ell_i$ respectent au moins l'une des deux premières règles de diversification énumérées précédemment. Essentiellement dans le cas d'un ensemble hétérogène, l'objectif est de trouver les poids π_i. À cet effet, chacune des quatre règles de combinaison citées plus haut peut être utilisée. Par exemple, Li *et al.* [83] ont créé un ensemble composé de K classifieurs (arbre C4.5 et K plus proches voisins). Ensuite lors de la prédiction, un sous-ensemble K^* de classifieurs est sélectionné. Ce sous-ensemble est pondéré à l'aide d'une mesure de confiance estimée pour chaque classifieur. Dans le cas des méthodes d'ensemble hétérogènes, le lecteur peut se référer à une récente revue traitant de différentes techniques de combinaison [43].

Ensemble homogène de classifieurs

L'ensemble $\mathcal{E}$ est dit homogène lorsque les classifieurs $\mathcal{C}\ell_i$ sont identiques (même type et même configuration). On cherche donc à diversifier les classifieurs entre eux ainsi qu'à trouver la bonne stratégie pour les combiner. La diversification se fait en général en utilisant la troisième stratégie mentionnée ci-dessus. Les sous-ensembles $\mathcal{D}_i$ sont soit connus à l'avance [126], soit ils doivent être construits [31, 114, 108]. Dans le premier cas, l'ensemble d'apprentissage $\mathcal{D}$ est formé par K sous-ensembles $\mathcal{D}_i$ provenant de diverses sources. Dans le second cas, les sous-ensembles $\mathcal{D}_i$ peuvent

être sélectionnés soit aléatoirement [31], soit par apprentissage non supervisé [108] ou par pondération des instances d'apprentissage [114]. Les classifieurs $\mathcal{C}\ell_i$ peuvent être entraînés séquentiellement ou indépendamment. Par exemple, pour les méthodes de type *boosting* notamment *Adaboost*, le premier classifieur $\mathcal{C}\ell_1$ de la séquence est entraîné sur l'ensemble $\mathcal{D}$ [55, 56, 114, 68, 141]. Les classifieurs $\{\mathcal{C}\ell_i\}_{i>1}$ restants sont pondérés à l'aide d'un coefficient dépendant des erreurs du classifieur précédent $\mathcal{C}\ell_{i-1}$. La sortie finale de la méthode *Adaboost* est obtenue à l'aide de la troisième règle de combinaison mentionnée plus haut. Les méthodes comme le *bagging* [31] et le *random subspace* [65] entraînent les classifieurs indépendamment et utilisent le vote de la majorité pour le résultat final. Il est aussi possible d'utiliser un algorithme d'apprentissage pour estimer les poids π_i. Cevikalp *et al.* [36] ont utilisé une somme convexe de classifieurs MLP sur un sous-ensemble de données. Ce dernier est sélectionné à l'aide des plus proches voisins de la nouvelle instance à classifier. La diversité des classifieurs est produite par la méthode *Adaboost*. Par contre, les poids utilisés au sein de la somme convexe sont déterminés à l'aide d'un modèle quadratique. Une autre approche connue est le mélange d'experts [70] où les classifieurs sont entraînés de manière à encourager la coopération entre eux. Nous présenterons cette méthode dans la section suivante.

1.3.3 Mélange d'experts

Le mélange d'experts (ME) [70] est une méthode d'ensemble qui consiste à diviser une tâche complexe de classification (ou de régression) en K sous-tâches identifiées par des régions. Une sous-tâche ici représente une tâche de classification réalisé à partir d'un sous-ensemble de l'ensemble d'apprentissage $\mathcal{D}$. Chaque sous-ensemble est délimité dans l'espace par des hyperplans décrits par une fonction paramétrique $\pi(x)$ ayant K sorties $\{\pi_i(x)\}_{i=1}^{K}$. Chaque sous-ensemble $\mathcal{D}_i$ est affecté à un classifieur probabiliste $p(y|\mathcal{C}\ell_i, x)$. La fonction $\pi(x)$ et les classifieurs sont respectivement appelés fonction de sélection et experts. Dans le modèle ME, les experts sont des régressions logistiques. La fonction de sélection $\pi(x)$ sert à définir la région de compétence de chaque expert de même que le sous-ensemble qui leur est assignée. La fonction de sélection $\pi(x)$ donne également la probabilité d'appartenance d'un exemple x à une

région donnée. Le mélange d'experts fait partie de la catégorie des ensembles homogènes de classifieurs mais diffère des méthodes présentées précédemment. Ici, le poids du $i^{ème}$ expert varie en fonction de l'exemple x et est donné par $\pi_i(x)$. De plus, la sélection des sous-ensembles d'apprentissage et l'apprentissage se font de façon itérative. À chaque itération, les performances des experts lors de l'itération précédente servent à définir leur nouvelle région de compétence. Ainsi, le long des itérations, les experts coopèrent entre eux afin d'obtenir une meilleure performance.

Plusieurs alternatives au mélange d'experts ont été proposées [94, 1, 74]. L'une de ces alternatives consiste à modéliser la fonction de sélection par un modèle non linéaire basé sur la gaussienne [135, 110, 94] ou sur les réseaux de neurones [94, 1] ou encore sur les modèles de Markov caché [137]. Une autre alternative revient à effectuer une division récursive de l'espace ce qui donne dans ce cas le modèle appelé mélange hiérarchique d'experts (HME) [74]. Pour un modèle HME à L niveaux, cette division se fait au travers d'un ensemble $\mathcal{F} = \left\{\pi^{(l)}(x)\right\}_{l=0}^{L-1}$ de fonctions de sélection. Dans ce cas, les sorties des experts sont hiérarchiquement combinées à l'aide des fonctions $\pi^{(l)}(x)$. Au mieux de nos connaissances, les fonctions de sélection au sein du modèle HME n'ont été modélisées qu'avec un modèle linéaire (division récursive de l'espace par des hyperplans). L'expression mathématique du modèle HME sera détaillée dans le chapitre 2.

1.4 Conclusion

Dans ce chapitre, nous avons présenté différentes méthodes utilisées pour la classification non linéaire des données. À travers l'analyse de ces méthodes, nous avons pu ressortir les propriétés suivantes (tab. 1.1) :

- Fonctionnement : la méthode de classification prends elle en compte la structure locale des données ?
- Nombre de paramètres : il s'agit du nombre de paramètres à estimer. Plus il y a de paramètres plus le temps d'apprentissage augmente.
- Taille de l'ensemble d'apprentissage : ceci fait référence au nombre d'observations requis par le classifieur lors de l'apprentissage.

- Temps d'apprentissage : le temps nécessaire à un classifieur pour apprendre. Un temps d'apprentissage moyen corresponds à quelques centaines de secondes tandis qu'un long temps peut aller à des semaines.
- Vitesse de prédiction : le temps mis par le classifieur pour prédire la classe d'une nouvelle instance.
- Utilisation de la distance : ceci nous indique si une quelconque distance est utilisé dans le processus d'apprentissage. Si tel est le cas, il faudra choisir une distance appropriée (euclidienne, mahalanobis, etc) ce qui n'est pas évident.
- Caractère explicatif : est ce que le classifieur retourne des résultats qui peuvent être facilement interprétés ? Le caractère explicatif est faible lorsqu'il est difficile de comprendre les décisions prises par le classifieur.

À partir de la lecture du tableau 1.1, nous notons que les méthodes comme celle du noyau et les RNA traitent les données comme un ensemble homogène sans s'intéresser à leur structure locale. Pour la méthode du noyau, le nombre de paramètres à estimer augmente avec la quantité de données. Par contre, pour les RNA le nombre de paramètres peut aller de quelques dizaines (pour les structures simples) à des centaines de millions. Ceci justifie par ailleurs le temps d'apprentissage très long des RNA (parfois plusieurs semaines). De plus, les RNA nécessitent en général un nombre élevé de données d'apprentissage et ont un caractère explicatif faible. La pondération de plusieurs classifieurs regroupe un ensemble de méthodes qui diffèrent entre elles, tant par la méthode de combinaison que par le type de classifieurs utilisé. Par exemple le choix de classifieurs linéaires réduira le temps d'apprentissage, augmentera la vitesse de prédiction et améliorera le caractère explicatif. Ce qui est le cas du mélange d'experts. Les travaux de cette thèse se basent sur le paradigme du mélange hiérarchique d'experts. Rappelons que dans le cadre du mélange hiérarchique d'experts, les fonctions de sélection sont modélisées par des modèles linéaires. Ce choix donne peu de flexibilité en ce qui concerne la définition des régions d'expertises. Dans cette thèse, nous nous intéressons à la réduction du nombre d'experts utilisés de même qu'à l'amélioration des prédictions. À cet effet, dans le prochain chapitre, nous proposons un modèle général pour le mélange hiérarchique de classifieurs.

	Méthode du noyau	**RNA**	**Pondération d'un classifieur**	**Pondération de plusieurs classifieurs**	**Mélange d'experts**	**Ppv**	**Arbres de décision**
Fonctionnement	Global	Global	Local	Dépend du type de combinaison	Local	Local	Local
Nombre de paramètres	Dépend du nombre d'instances d'apprentissage	Peut atteindre des centaines de millions	Quelques dizaines	Quelques dizaines	Quelques dizaines	-	-
Taille de l'ensemble d'apprentissage	Quelconque	Très grande	Quelconque	Quelconque	Quelconque	Quelconque	Quelconque
Temps d'apprentissage	Moyen	Long	Court	Dépend des classifieurs utilisés et du type de combinaison	Moyen	Moyen	Court
Vitesse de prédiction	Lente	Lente	Lente	Peut être lente ou moyenne	Moyenne	lente	Rapide
Utilisation de la distance	Oui	Non	Oui	Dépend de la méthode	Non	Oui	Non
Caractère explicatif	Faible	Faible	Peut être faible, moyen ou fort dépendemment du classifieur	Peut être faible ou fort dépendemment des classifieurs	Fort	Fort	Fort

tableau 1.1 – Comparaison des méthodes de classification non linéaires.

Chapitre 2

Mélange hiérarchique de classifieurs discriminatifs : comment utiliser peu de ressources tout en gardant des performances élevées ?

Afin de résoudre une tâche complexe au sein d'une entreprise, les ressources humaines sont souvent amenées à mettre en place une équipe d'experts. Il est souvent nécessaire de choisir un nombre restreint d'experts tout en garantissant des performances élevées. Il existe plusieurs motivations derrière la réduction du nombre d'experts telles que le coût financier, les outils disponibles (ex. licences de logiciels) et la durée de formation de ces experts. Pour mettre en œuvre la réduction du nombre d'experts, un ensemble de superviseurs et d'experts sont sélectionnés. Les superviseurs sont organisés de sorte qu'ils puissent attribuer efficacement une sous-tâche précise à chaque expert. Dans le contexte d'une tâche de classification des données par le modèle HME (voir section 1.3.3), un expert et un superviseur représentent respectivement un classifieur et une fonction de sélection. Affecter une sous-tâche à un expert est équivalent à sélectionner un sous-ensemble d'apprentissage à partir duquel un classifieur sera construit. Réduire le nombre d'experts à travers le paradigme du

modèle HME revient à : 1) la sélection d'un nombre adéquat d'experts en utilisant des méthodes d'apprentissage, 2) choisir un ensemble approprié de fonctions de sélection, 3) organiser efficacement les experts et les fonctions de sélection.

Il a été montré que la division hiérarchique de l'espace des données par le modèle HME contribue à une meilleure répartition des sous-tâches entre les experts [111]. Cependant, plus le nombre de niveaux augmente dans la hiérarchie, plus il y a de paramètres à estimer et plus le temps d'apprentissage augmente. Xu *et al.* [135] ont proposé comme alternative au HME, un modèle de mélange sans hiérarchie où la fonction de sélection est basée sur la gaussienne. Avec ce modèle, le temps d'apprentissage peut être réduit tout ayant une meilleure généralisation comparé au modèle HME [111, 95]. De plus, il a été noté qu'avec ce modèle alternatif, on peut utiliser moins d'experts que HME [110]. Le nombre d'experts utilisé dépend donc de la manière dont l'espace des données est divisé. Cependant, les précédents travaux se sont limités à un modèle sans structure hiérarchique lorsque les fonctions de sélection sont modélisées par des modèles non linéaires [95, 118, 1].

Dans ce chapitre, nous proposons un modèle général suivant le même paradigme que le modèle HME mais qui permet l'utilisation de modèles non linéaires pour modéliser les fonctions de sélection. Le but est de garder l'avantage de la structure hiérarchique [111] de même que celui de la division non linéaire de l'espace. Comme exemples pour montrer comment réduire le nombre d'experts, nous déduisons deux nouveaux modèles à partir du modèle général. Au travers d'expériences sur plusieurs collections de référence, nous avons montré que ces deux modèles peuvent utiliser peu d'experts tout en gardant une erreur de prédiction faible. Les modèles déduits du modèle général utilisent des classifieurs linéaires comme experts. De ce fait, il est possible d'expliquer localement les prédictions faites. Le restant de ce chapitre est organisé en trois sections. Dans la section 2.1, nous introduisons le modèle général du mélange hiérarchique de classifieurs discriminatifs, ses liens avec quelques modèles et la stratégie de réduction du nombre d'experts. Dans la section 2.2, nous présentons les algorithmes d'apprentissage de nos modèles. Finalement, les résultats expérimentaux sont présentés dans la section 2.3.

2.1 Modèle général du mélange hiérarchique de classifieurs discriminatifs

2.1.1 Modèle

Un classifieur discriminatif est défini en établissant un lien entre le vecteur x et l'étiquette y à travers une distribution conditionnelle $p(y|x, \Omega)$ ou un score $Conf(y|x, \Omega)$ contrôlé par un ensemble de paramètres Ω. Soit le problème de classification illustré à la figure 2.1 où les données sont réparties en deux classes («o» en bleu et «$*$» en rouge). Il est évident qu'un classifieur linéaire ne peut séparer correctement ces deux classes.

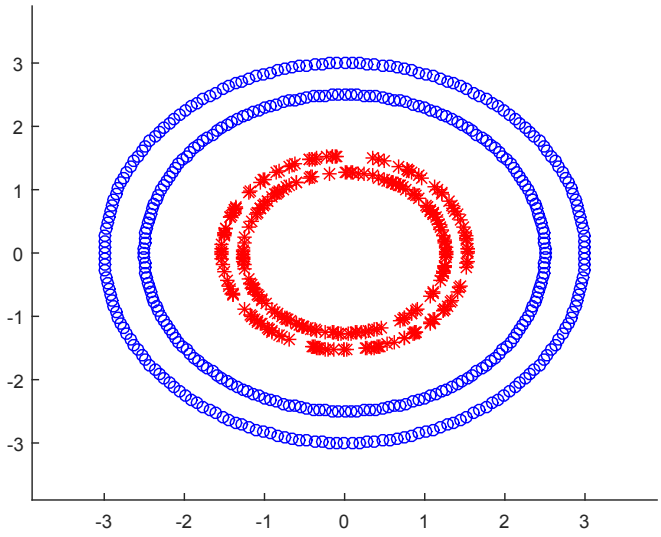
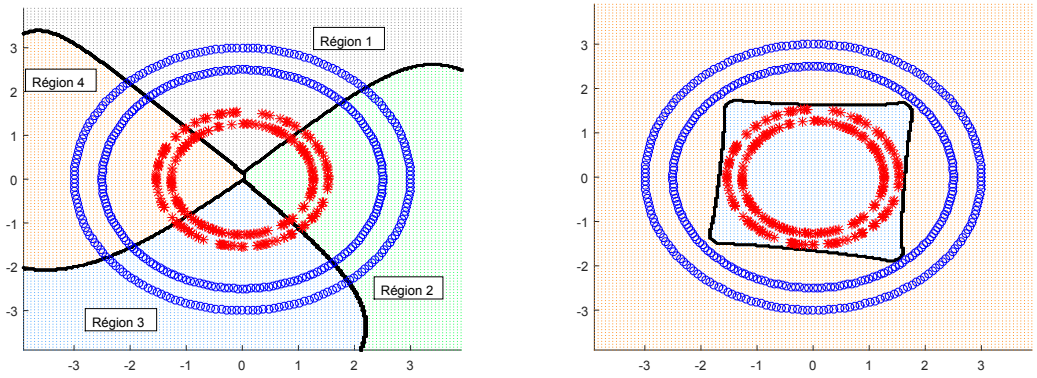


figure 2.1 – Illustration de données non linéairement séparables.

Nous pouvons cependant, diviser l'espace des données en quatre régions délimitées par les courbes en noires (Fig. 2.2a). Un ensemble de quatre classifieurs linéaires (experts) peut être utilisé pour séparer les données présentes dans chaque région. La frontière de décision issue de ce processus est illustrée à la figure 2.2b par la courbe en noire. De manière générale, ce processus peut être modélisé par le mélange suivant :



(a) Division en Régions.

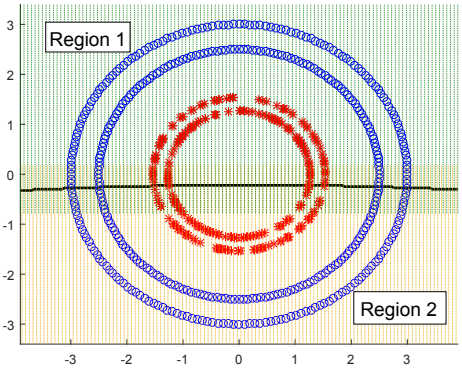
(b) Frontière générée par un mélange de 4 classifieurs linéaires.

figure 2.2 – Illustration du mélange de classifieurs.

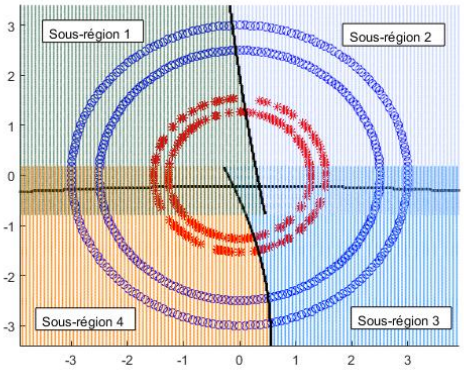
$$\begin{aligned} p\left(y|x,\,\Omega\right) &= \sum_{i=1}^{K} \pi_i^{(0)}\left(x\right) p(y|x,\Omega_i^{(1)}); \\ s.t\ \pi_i^{(0)}\left(x\right) &\geq 0;\ \sum_{i=1}^{K} \pi_i^{(0)}\left(x\right) = 1 \end{aligned}, \tag{2.1}$$

où Ω est l'ensemble des paramètres, K le nombre de composantes du mélange. Les paramètres du classifieur $p(y|x,\Omega_i^{(1)})$ sont représentés par $\Omega_i^{(1)}$. La probabilité que l'instance de données x se situe dans la i-ème région est donnée par $\pi_i^{(0)}\left(x\right)$. Chaque classifieur $p(y|x,\Omega_i^{(1)})$ discrimine les données appartenant à la $i^{ème}$ région. La fonction de sélection $\pi^{(0)}\left(x\right)$ ayant K sorties $\left\{\pi_i^{(0)}(x)\right\}_{i=1}^{K}$ est une fonction paramétrique dépendante d'un ensemble de paramètres $\Omega^{(0)}$. Cette fonction décrit les limites spatiales de chaque région (Fig. 2.2a).

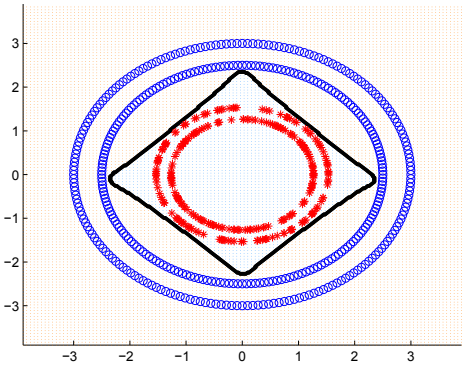
Pour le problème illustré dans la figure 2.1, il est également possible de diviser l'espace récursivement. Dans un premier temps, on effectue une division de l'espace en deux régions délimitées par la courbe noire (Fig. 2.3a). Ensuite, chaque région est divisée en deux sous-régions (Fig. 2.3b). Les quatre classifieurs utilisés précédemment peuvent alors séparer les données se trouvant dans chaque sous-région. La frontière de décision issue de ce processus est illustrée à la figure 2.3c par la courbe en noire. Dans ce cas, chaque classifieur $p\left(y|x,\,\Omega_i^{(1)}\right)$ peut être exprimé à son tour sous la forme d'un mélange :



(a) Division en régions.



(b) Division en sous-régions.



(c) Frontière générée par un mélange hiérarchique de 4 classifieurs.

figure 2.3 – Illustration du mélange hiérarchique de classifieurs.

$$\begin{array}{c} p\left(y|x,\,\Omega_i^{(1)}\right) = \sum\limits_{j=1}^{M_i} \pi_{j|i}^{(1)}\left(x\right) p(y|x,\Omega_{ij}^{(2)}); \\ s.t\ \pi_{j|i}^{(1)}\left(x\right) \geq 0;\ \sum\limits_{i=1}^{M_i} \pi_{j|i}^{(1)}\left(x\right) = 1 \end{array}, \tag{2.2}$$

où M_i désigne le nombre de sous-régions associées à la région indexée par i. Pour une

hiérarchie à deux niveaux, le classifieur final $p\left(y|x,\,\Omega\right)$ peut s'écrire comme suit :

$$\begin{aligned} p\left(y|x,\,\Omega\right) = \sum_{i=1}^{K} \pi_i^{(0)}\left(x\right) \sum_{j=1}^{M_i} \pi_{j|i}^{(1)}\left(x\right) p\left(y|x,\,\Omega_{ij}^{(2)}\right); \\ s.t\, \pi_i^{(0)}\left(x\right) \geq 0;\; \pi_{j|i}^{(1)}\left(x\right) \geq 0;\;, \\ s.t \sum_{i=1}^{K} \pi_i^{(0)}\left(x\right) = 1;\; \sum_{j=1}^{M_i} \pi_{j|i}^{(1)}\left(x\right) = 1 \end{aligned} \tag{2.3}$$

où les nombres K et $\{M_i\}_{i=1}^{K}$ sont dépendants des données, du choix des classifieurs $p(y|x,\Omega_{ij}^{(2)})$ et de l'ensemble des fonctions de sélection $\mathcal{F} = \left\{\pi^{(0)}(x), \pi_{|1}^{(1)}(x), \cdots, \pi_{|K}^{(1)}(x)\right\}$. La probabilité que l'instance x appartienne à une sous-région sachant qu'elle appartient à la $i^{ème}$ région est donnée par $\pi_{|i}^{(1)}(x)$. La fonction $\pi_{|i}^{(1)}(x)$ est contrôlée par un ensemble de paramètres $\Omega_i^{(1)}$ et a M_i sorties. En répétant ce raisonnement sur chaque sous-région, nous pouvons déduire un modèle hiérarchique ayant plusieurs niveaux. Cependant, dans ce chapitre, nous nous limitons à une hiérarchie à deux niveaux (voir figure 2.4). Ce choix a été effectué afin de réduire la complexité algorithmique. En effet, une hiérarchie plus profonde introduira de nouveaux paramètres à estimer (augmentation du nombre de fonctions de sélection).

Les fonctions de sélection peuvent prendre diverses formes tant que leur sortie est une probabilité. Nous choisissons la forme suivante qui garantit les contraintes imposées aux fonctions de sélection (Eq. 2.3) :

$$\pi_i^{(0)}\left(x\right) = \frac{\exp\left(\zeta_i\left(x\right)\right)}{\sum_{l=1}^{K} \exp\left(\zeta_l\left(x\right)\right)};\; \pi_{j|i}^{(1)}\left(x\right) = \frac{\exp\left(\vartheta_{j|i}\left(x\right)\right)}{\sum_{l=1}^{M_i} \exp\left(\vartheta_{l|i}\left(x\right)\right)},$$

où $\zeta_l\left(x\right)$ et $\vartheta_{l|i}\left(x\right)$ sont deux fonctions paramétriques. Ainsi, la forme de $\pi_i^{(0)}$ et $\pi_{j|i}^{(1)}$ dépend du choix de $\zeta_l\left(x\right)$ et $\vartheta_{l|i}\left(x\right)$. Nous avons nommé le modèle décrit dans l'équation 2.3 : modèle général du mélange hiérarchique de classifieurs discriminatifs (HMD). L'ensemble des paramètres du modèle HMD est $\Omega = \{\Omega^{(l)}\}_{l=0}^{2}$ où $\Omega^{(1)} = \left\{\Omega_i^{(1)}\right\}_{i=1}^{K}$ et $\Omega^{(2)} = \left\{\Omega_{ij}^{(2)}\right\}$ avec $i = 1 \cdots K$ et $j = 1 \cdots M_i$.

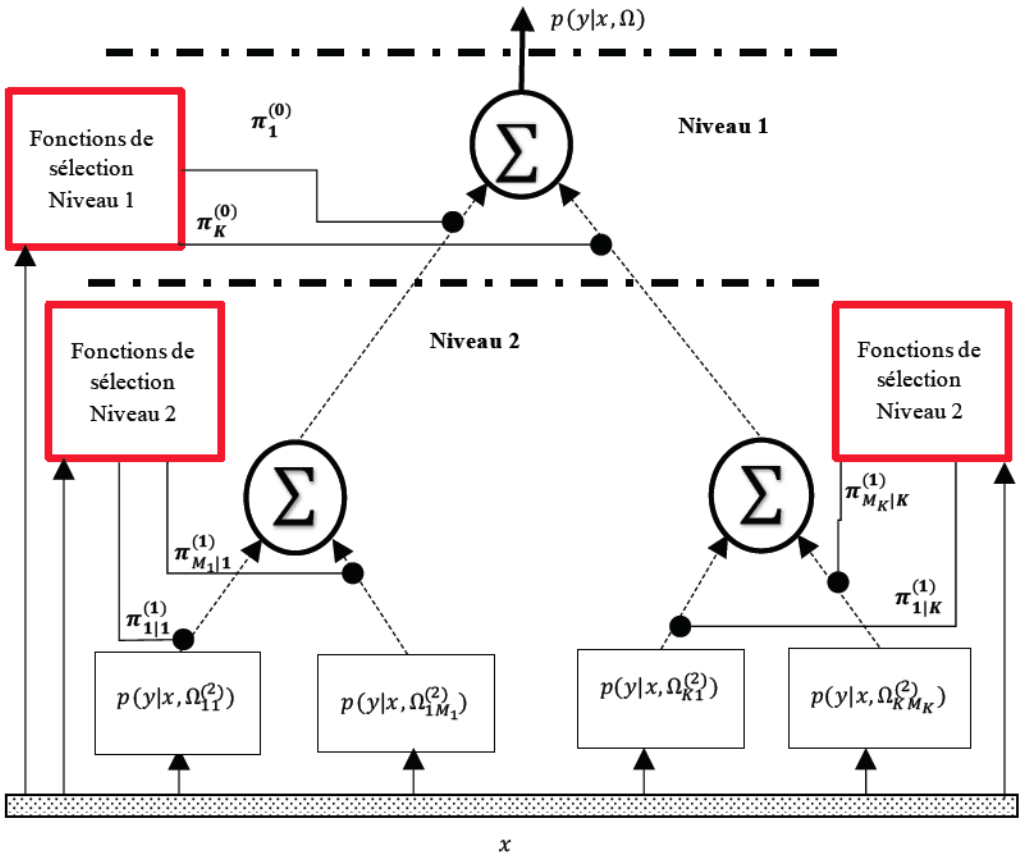


figure 2.4 – Illustration de l'architecture du modèle HMD ayant une hiérarchie à deux niveaux.

Dans le reste de ce chapitre, chaque classifieur discriminatif est choisi comme une régression logistique multinomiale et nous avons pour $i = 1 \cdots K,\ j = 1 \cdots M_i$:

$$p\left(y = c|x;\ \Omega_{ij}^{(2)}\right) = \begin{cases} \dfrac{\exp\left(\theta_{ijc}^T \tilde{x}\right)}{1 + \sum_{h=1}^{C-1} \exp\left(\theta_{ijh}^T \tilde{x}\right)} & c \neq C \\ \dfrac{1}{1 + \sum_{h=1}^{C-1} \exp\left(\theta_{ijh}^T \tilde{x}\right)} & c = C \end{cases}, \tag{2.4}$$

où C est le nombre de classes, $\Omega_{ij}^{(2)} = \{\theta_{ijh}^{(h)}\}_{h=1}^{C-1}$, $\theta_{ijh} \in \mathbb{R}^{D+1}$ et $\tilde{x} = \begin{pmatrix} 1 \\ x \end{pmatrix}$.

Le choix de ce classifieur est motivé par le fait que les modèles linéaires sont rapides à entraîner, faciles à implémenter et leurs prédictions sont simples à expliquer. Cependant, notre modèle peut utiliser des classifieurs plus complexes comme les réseaux de neurones et autres (voir chapitre 3). Dans la suite de cette thèse, nous utiliserons le terme expert pour désigner un classifieur lorsqu'il est utilisé au sein d'un mélange.

2.1.2 Relation entre HMD, le modèle HME et ses alternatives

Le mélange hiérarchique d'experts [74] (*HME*) est un modèle discriminatif de mélange bien connu. Les modèles HME et HMD partagent la même structure arborescente (voir figure 2.4). La différence entre HME et HMD est le fait que dans le modèle HME, les fonctions de sélection ne produisent que des hyperplans. Nous pouvons obtenir le modèle *HME* à partir de l'équation 2.3 en prenant $\zeta_l(x)$ et $\vartheta_{l|i}(x)$ comme des fonctions linéaires de x. Nous avons alors les expressions suivantes :

$$\pi_i^{(0)}(x) = \frac{\exp\left(\eta_i^T \tilde{x}\right)}{\sum_{l=1}^{K} \exp\left(\eta_l^T \tilde{x}\right)};\ \pi_{j|i}^{(1)}(x) = \frac{\exp\left(\nu_{ij}^T \tilde{x}\right)}{\sum_{l=1}^{M_i} \exp\left(\nu_{il}^T \tilde{x}\right)}, \tag{2.5}$$

où η_l et $\nu_{il} \in \mathbb{R}^{D+1}$. Une alternative au HME est appelé *Localized mixture of experts* (LME) où la fonction de sélection est modélisée en se basant sur la gaussienne [135]. LME est un modèle de mélange sans hiérarchie (Eq. 2.2) et l'expression de sa fonction de sélection est la suivante :

$$\begin{gathered} \pi_i^{(0)}(x) = \frac{\alpha_i g(x, \mu_i, \Sigma_i)}{\sum_{l=1}^{K} \alpha_l g(x, \mu_l, \Sigma_l)};\ i = 1 \cdots K \\ s.\text{à} \sum_{l=1}^{K} \alpha_l = 1;\ \alpha_l > 0 \end{gathered}, \tag{2.6}$$

où $g(x, \mu_l, \Sigma_l)$ est la distribution gaussienne ayant pour moyenne μ_l et Σ_l comme matrice de covariance. LME est équivalent au modèle HMD sans hiérarchie où $\zeta_i(x) = \ln(\alpha_i) - \frac{1}{2}\Big[(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i) + \ln|\Sigma_i| + D\ln(2\pi)\Big]$. Lorsque des modèles plus complexes ont été utilisés pour modéliser la fonction de sélection, les travaux précé-

dents sur les mélanges discriminatifs n'ont aucune structure hiérarchique [95, chap 4],[138, 90, 1]. Il est à noter que la plupart des fonctions de sélection utilisées dans les travaux précédents peuvent être définies compte tenu de l'expression donnée à ζ_i. Les modèles résultants ne sont alors qu'une instance du modèle HMD.

2.1.3 Choix des fonctions de sélection

Étant donné que les fonctions de sélection sont utilisées dans notre modèle (HMD) pour classifier les données en sous-populations, elles peuvent être modélisées soit par une approche paramétrique [1] ou non paramétrique [118]. D'une part, un modèle paramétrique résume les données par un ensemble de paramètres (de taille fixe) qui est indépendant du nombre d'instances d'apprentissage. L'efficacité de ce type de modèle dépend du choix de la fonction paramétrique. Pour avoir de meilleurs scores, il est possible d'utiliser une fonction plus flexible (ex. RNA [1]) et obtenir par ce même fait un modèle peu explicatif. Il y a donc un compromis entre flexibilité de la fonction paramétrique et la capacité à expliquer les décisions de cette dernière (fonctionnement en boîte noire). D'autre part, dans un modèle non paramétrique, le nombre de paramètres est lié au nombre d'instances d'apprentissage. Les modèles non paramétriques sont plus flexibles que les paramétriques. Cependant, l'entraînement des modèles non-paramétriques est lent. Par conséquent, afin de réduire le temps d'apprentissage, nous avons choisi une approche paramétrique pour modéliser les fonctions de sélection. Une stratégie qui peut être appliquée pour disposer ces fonctions dans la structure du modèle HMD consiste à modéliser les données par un mélange hétérogène de distributions [49]. Dans ce cas, une procédure similaire à celle appliquée par El-Zaart *et al.* [49] peut être utilisée pour sélectionner chacune des distributions. Les fonctions de sélection sont alors choisies comme étant la probabilité *a posteriori* d'appartenir à une région. Afin de simplifier nos calculs, nous nous sommes limités au cas particulier du mélange homogène de distributions.

Considérons la fonction de sélection utilisée dans le modèle *LME* (Eq. 2.6), elle peut être réécrite comme suit :

$$\pi_i^{(0)}(x) = \frac{\exp\left(-\frac{x^T\Sigma_i^{-1}x+\ln|\Sigma_i|}{2}\right)\exp\left(\ln(\alpha_i) - \frac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i + \mu_i^T\Sigma_i^{-1}x\right)}{\sum_{l=1}^{K}\exp\left(-\frac{x^T\Sigma_l^{-1}x+\ln|\Sigma_l|}{2}\right)\exp\left(\ln(\alpha_l) - \frac{1}{2}\mu_l^T\Sigma_l^{-1}\mu_l + \mu_l^T\Sigma_l^{-1}x\right)} \quad . \tag{2.7}$$

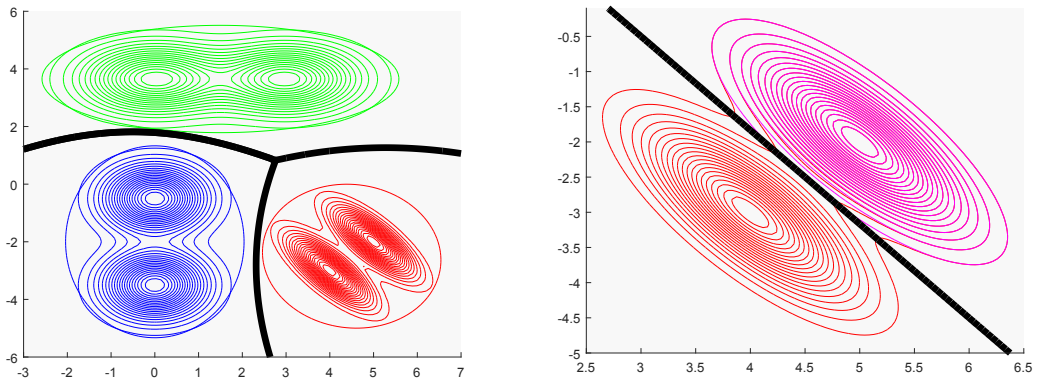
En définissant $a_i = \mu_i^T\Sigma_i^{-1}$ et $b_i = \ln(\alpha_i) - \frac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i$, cette expression devient

$$\pi_i^{(0)}(x) = \frac{\exp\left(-\frac{x^T\Sigma_i^{-1}x+\ln|\Sigma_i|}{2}\right)\exp\left(\eta_i^T\tilde{x}\right)}{\sum_{l=1}^{K}\exp\left(-\frac{x^T\Sigma_l^{-1}x+\ln|\Sigma_l|}{2}\right)\exp\left(\eta_l^T\tilde{x}\right)} \quad , \tag{2.8}$$

où $\eta_l = (a_l, b_l)^T$. En supposant que $\Sigma_i = \Sigma, \forall i = 1 \cdots K$, le terme quadratique $x^T\Sigma_l^{-1}x$ est éliminé. Dans ce cas, l'expression de $\pi_i^{(0)}(x)$ dans l'équation 2.8 est équivalent à celle définie à l'équation 2.5. Les fonctions de sélection formées à partir de la gaussienne décrivent donc une frontière linéaire lorsque les données présentes dans les régions partagent les mêmes covariances. En revanche, une frontière incurvée est décrite lorsque les régions ont des covariances différentes (voir Fig. 2.5). Partant de ces faits, nous déduisons deux exemples montrant que le choix approprié de la fonction de sélection peut permettre de réduire le nombre d'experts utilisés. Le premier choix consiste à utiliser différents types de fonctions de sélection à chaque niveau de la hiérarchie du modèle HMD. Quant au second choix, le même type de fonctions de sélection est utilisé à tous les niveaux. Au travers d'expériences, nous verrons lequel des deux exemples est efficace. Ce même principe peut être appliqué à un modèle HMD ayant plus de deux niveaux.

Ensemble hétérogène des fonctions de sélection

L'ensemble $\mathcal{F}$ des fonctions de sélection est dite hétérogène lorsqu'il est composé de fonctions ayant différentes formes. À titre d'exemple, la forme prise par les fonctions de sélection du premier niveau est différente de celle prise par les fonctions de sélection du second niveau. Dans ce cas, nous assumons que les données présentes dans les régions décrites par $\left\{\pi_i^{(0)}(x)\right\}_{i=1}^{K}$ peuvent avoir différentes covariances. Nous choisissons donc



(a) Régions avec différentes matrices de covariance.

(b) Régions partageant la même matrice de covariance.

figure 2.5 – Illustration des frontières décrites par les fonctions de sélection basées sur la gaussienne.

l'expression donnée par l'équation 2.6 pour la fonction de sélection du premier niveau. En effet, en utilisant cette forme de fonction, les experts se spécialisent sur des régions moins étendues ce qui leur permet d'être plus efficaces [110]. Par contre, au second niveau, vu que les données présentes dans chaque région représentent des tâches de classification moins complexes, nous supposons que les données présentes dans les sous-régions partagent une même covariance. Dans ce cas afin d'éviter l'estimation de Σ ($\frac{D^2+D}{2}$ éléments), nous utilisons directement un modèle linéaire défini par :

$$\pi^{(1)}_{j|i}(x) = \frac{\exp\left(\delta^T_{ij}\tilde{x}\right)}{\sum_{h=1}^{M_i}\exp\left(\delta^T_{ih}\tilde{x}\right)};\ i = 1\cdots K;\ j = 1\cdots M_i, \tag{2.9}$$

où $\delta_{ij} \in \mathbb{R}^{D+1}$. Nous appelons ce premier exemple HMD1.

Ensemble homogène des fonctions de sélection

L'ensemble $\mathcal{F}$ des fonctions de sélection est dite homogène lorsqu'il est composé de fonctions ayant les mêmes formes. Dans la littérature des mélanges discriminatifs, plusieurs fonctions de sélection décrivant des frontières non linéaires ont été utilisées. Nous pouvons citer comme fonctions, celles basées sur la gaussienne [135, 110, 94] et

les réseaux de neurones [94, 1]. Au meilleur de nos connaissances, ces fonctions ont été uniquement utilisées dans des modèles de mélange n'ayant aucune structure hiérarchique. Le modèle HMD permet d'étendre, par exemple, l'utilisation de la fonction de sélection basée sur la gaussienne à tous les niveaux de la hiérarchie. Contrairement au premier exemple, nous voulons prendre en compte la possibilité que les données présentes dans les sous-régions aient également différentes covariances. Ce second exemple est appelé HMD2. En suivant la notation adoptée dans la section 2.1.2, pour i fixé, les fonctions de sélection du deuxième niveau sont décrites comme suit :

$$\begin{aligned} \pi^{(1)}_{j|i}(x) = \frac{\beta_{ij} g(x, \xi_{ij}, \sigma_{ij})}{\sum_{l=1}^{M_i} \beta_{il} g(x, \xi_{il}, \sigma_{il})};\ j = 1 \cdots M_i; \\ s.\text{à} \sum_{l=1}^{M_i} \beta_{il} = 1;\ \beta_{il} > 0. \end{aligned} \tag{2.10}$$

Dans un souci de simplicité, nous choisissons ici σ_{ij} comme une matrice diagonale. La fonction de sélection $\pi^{(0)}(x)$ garde la même expression définie pour le premier exemple (HMD1).

2.2 Estimation des paramètres via le maximum de vraisemblance

Soit $\mathbf{Obs} = \{\mathbf{X}, \mathbf{Y}\}$ l'ensemble des données observées où $\mathbf{X} = \{x_1, \cdots, x_N\} \in \mathbb{R}^{D \times N}$ est un ensemble de vecteurs indépendants et identiquement distribués et $\mathbf{Y} = (y_1, \cdots, y_N)^T \in \mathbb{R}^N$ leurs étiquettes. Le nombre d'observations est noté N. Étant donné que notre modèle général (Eq. 2.3) est un modèle de mélange, l'algorithme EM (Espérance - Maximisation) [46] peut être utilisé pour le processus d'apprentissage. Il est utile de rappeler que l'algorithme EM est itératif et est composé de deux phases : le calcul de l'espérance suivi de la maximisation de cette dernière. Soit $\mathcal{L}$ la *log*-vraisemblance incomplète du modèle HMD :

$$\mathcal{L} = \sum_{n=1}^{N} \ln \left[\sum_{i=1}^{K} \pi_i^{(0)}(x_n) \sum_{j=1}^{M_i} \pi^{(1)}_{j|i}(x_n) p(y_n | x_n, \Omega^{(2)}_{ij}) \right]. \tag{2.11}$$

Conformément à la littérature de l'algorithme EM, nous introduisons un ensemble de variables aléatoires binaires «cachées» : $z_{n,i}^{(0)}$ et $z_{n,j|i}^{(1)}$ respectivement pour le premier et le second niveau du modèle HMD. Si x_n appartient à la i^{ieme} région alors $z_{n,i}^{(0)} = 1$ et 0, si non. Étant donné la région i, si x_n appartient à la sous-région j alors $z_{n,j|i}^{(1)} = 1$ et 0, si non. La *log*-vraisemblance complète peut être écrite comme suit :

$$\begin{aligned}
\mathcal{L}^c = & \sum_{n=1}^{N}\sum_{i=1}^{K} z_{n,i}^{(0)} \sum_{j=1}^{M_i} z_{n,j|i}^{(1)} \ln\left[\pi_i^{(0)}(x_n)\,\pi_{j|i}^{(1)}(x_n)\,p\left(y_n|x_n,\,\Omega_{ij}^{(2)}\right)\right] \\
= & \sum_{n=1}^{N}\sum_{i=1}^{K} z_{n,i}^{(0)} \ln\left[\pi_i^{(0)}(x_n)\right] + \sum_{n=1}^{N}\sum_{i=1}^{K}\sum_{j=1}^{M_i} z_{n,i}^{(0)} z_{n,j|i}^{(1)} \ln\left[\pi_{j|i}^{(1)}(x_n)\right] \\
& + \sum_{n=1}^{N}\sum_{i=1}^{K}\sum_{j=1}^{M_i} z_{n,i}^{(0)} z_{n,j|i}^{(1)} \ln\left[p\left(y_n|x_n,\,\Omega_{ij}^{(2)}\right)\right]
\end{aligned} \tag{2.12}$$

Lors de la première phase (*E-step*) de l'algorithme EM, nous calculons l'espérance suivante :

$$Q\left(\Omega,\,\Omega_{(t)}\right) = \sum_{n=1}^{N}\sum_{i=1}^{K} h_{n,i}^{(0)} \sum_{j=1}^{M_i} h_{n,j|i}^{(1)} \ln\left[\pi_i^{(0)}(x_n)\,\pi_{j|i}^{(1)}(x_n)\,p\left(y_n|x_n,\,\Omega_{ij}^{(2)}\right)\right], \tag{2.13}$$

où $\Omega_{(t)}$ représente les paramètres à la $t-$ème itération. L'espérance respective de $z_{n,i}^{(0)}$ et de $z_{n,j|i}^{(1)}$ est $h_{n,i}^{(0)}$ et $h_{n,j|i}^{(1)}$:

$$\begin{aligned}
h_{n,i}^{(0)} = & \frac{p\left(z_{n,i}^{(0)} = 1 \mid x_n, \Omega\right) p\left(y_n \mid x_n, \Omega, z_{n,i}^{(0)} = 1\right)}{p\left(y_n \mid x_n, \Omega\right)} \\
= & \frac{\pi_i^{(0)}(x_n) \sum_{j=1}^{M_i} \pi_{j|i}^{(1)}(x_n)\, p\left(y_n|x_n,\,\Omega_{ij}^{(2)}\right)}{\sum_{i=1}^{K} \pi_i^{(0)}(x_n) \sum_{j=1}^{M_i} \pi_{j|i}^{(1)}(x_n)\, p\left(y_n|x_n,\,\Omega_{ij}^{(2)}\right)}
\end{aligned} \tag{2.14}$$

et

$$\begin{aligned} h_{n,j|i}^{(1)} = & \frac{p\left(z_{n,j|i}^{(1)}=1 \mid x_n, \Omega_i^{(1)}\right) p\left(y_n \mid x_n, \Omega_i^{(1)}, z_{n,j|i}^{(1)}=1\right)}{p\left(y_n|x_n,\, \Omega_i^{(1)}\right)} \\ = & \frac{\pi_{j|i}^{(1)}\left(x_n\right) p\left(y_n|x_n,\, \Omega_{ij}^{(2)}\right)}{\sum_{j=1}^{M_i} \pi_{j|i}^{(1)}\left(x_n\right) p\left(y_n|x_n,\, \Omega_{ij}^{(2)}\right)} \end{aligned} \tag{2.15}$$

L'équation 2.13 peut être séparée en trois parties indépendantes. Ceci permet d'estimer séparément les paramètres des fonctions de sélection et des experts Les équations suivantes donnent les opérations d'optimisation à effectuer lors de la seconde phase (*M-step*) de l'algorithme EM :

$$\Omega_{(t+1)}^{(0)} = argmax_{\Omega^{(0)}} \sum_{n=1}^{N}\sum_{i=1}^{K} h_{n,i}^{(0)} \ln\left(\pi_i^{(0)}\left(x_n\right)\right) \ , \tag{2.16}$$

pour $i = 1 \cdots K$,

$$\Omega_{i,(t+1)}^{(1)} = \ argmax_{\Omega_i^{(1)}} \sum_{n=1}^{N} h_{n,i}^{(0)} \sum_{j=1}^{M_i} h_{n,j|i}^{(1)} \ln\left(\pi_{j|i}^{(1)}\left(x_n\right)\right) \ , \tag{2.17}$$

pour $i = 1 \cdots K$ et $j = 1 \cdots M_i$

$$\Omega_{ij,(t+1)}^{(2)} = argmax_{\Omega_{ij}^{(2)}} \sum_{n=1}^{N} h_{n,i}^{(0)} h_{n,j|i}^{(1)} \sum_{c=1}^{C} v_{n,c} \ln\left(p\left(y_n = c|x_n,\, \Omega_{ij}^{(2)}\right)\right) \ , \tag{2.18}$$

où $\Omega_{(t+1)} = \{\Omega_{(t+1)}^{(l)}\}_{l=0}^{2}$ sont les paramètres à la $(t+1) - ème$ itération. La variable binaire notée $v_{n,c}$ indique si l'instance x_n appartient ou non à la classe c. Dépendemment des expressions de $\pi_i^{(0)}\left(x_n\right)$, $\pi_{j|i}^{(1)}\left(x_n\right)$ et $p\left(y_n = c|x_n,\, \Omega_{ij}^{(2)}\right)$, ces opérations d'optimisation peuvent être non linéaires.

2.2.1 Estimation des paramètres pour le modèle HMD1

Dans le cas de HMD1, $\Omega^{(0)} = \{\alpha_i, \mu_i, \Sigma_i\}_{i=1}^{K}$ et $\Omega_i^{(1)} = \{\delta_{ij}\}_{j=1}^{M_i}$. En substituant l'expression de $\pi_i^{(0)}\left(x_n\right)$ (Eq. 2.6) dans l'équation 2.16, nous serons amenés à résoudre un problème d'optimisation non linéaire.

Pour alléger cette charge de calcul, nous appliquons l'algorithme EM sur la proba-

bilité jointe $p(y_n|\, x_n, \Omega)p(x_n|\Omega)$ où $p(x_n|\Omega) = \sum_{l=1}^{K} \alpha_l g_l(x_n, \mu_l, \Sigma_l)$ [135, 94]. Ceci revient à remplacer l'expression de $\pi_i^{(0)}(x_n)$ dans les équations 2.11 à 2.16 par $\alpha_i g_i(x_n, \mu_i, \Sigma_i)$. Une fois cette substitution faite, nous pouvons estimer $\Omega^{(0)}$ avec les équations suivantes : Pour $i \in \{1 \cdots K\}$

$$\begin{aligned} {\alpha_i}^{(t+1)} &= \frac{1}{N}\sum_{n=1}^{N} h_{n,i}^{(0)}; \quad {\mu_i}^{(t+1)} = \frac{\sum_{n=1}^{N} h_{n,i}^{(0)} x_n}{\sum_{n=1}^{N} h_{n,i}^{(0)}} \\ {\Sigma_i}^{(t+1)} &= \frac{\sum_{n=1}^{N} h_{n,i}^{(0)} \left(x_n - {\mu_i}^{(t)}\right)^T \left(x_n - {\mu_i}^{(t)}\right)}{\sum_{n=1}^{N} h_{n,i}^{(0)}}. \end{aligned} \tag{2.19}$$

En substituant les expressions définies aux équations 2.4 et 2.9 dans les équations 2.17 et 2.18, nous pouvons estimer les paramètres δ_{ij} et θ_{ijc}. Ceci peut se faire par un algorithme standard d'optimisation non linéaire et les gradients sont donnés par :

$$\nabla_{\delta_{ij}} Q\left(\Omega, \Omega_{(t)}\right) = \sum_{n=1}^{N} h_{n,i}^{(0)} \left[h_{n,j|i}^{(1)} - \pi_{j|i}^{(1)}(x_n)\right] \tilde{x}_n, \; i \in \{1 \cdots K\} \tag{2.20}$$

$$\begin{gathered} \nabla_{\theta_{ijc}} Q\left(\Omega, \Omega_{(t)}\right) = \sum_{n=1}^{N} h_{n,i}^{(0)} h_{n,j|i}^{(1)} \left[v_{cn} - p\left(y_n = c|x_n; \theta_{ij}\right)\right] \tilde{x}_n, \\ i \in \{1 \cdots K\}, \; j \in \{1 \cdots M_i\} \end{gathered} \tag{2.21}$$

où $\tilde{x}_n = (1, x_n)^T$. L'algorithme (2.1) résume la procédure d'apprentissage du modèle HMD1.

2.2.2 Estimation des paramètres pour le modèle HMD2

Pour le modèle HMD2, $\Omega^{(0)} = \{\alpha_i, \mu_i, \Sigma_i\}_{i=1}^{K}$ et $\Omega^{(1)} = \{\beta_{ij}, \xi_{ij}, \sigma_{ij}\}_{ij=11}^{KM_i}$. Les expressions de la fonction $\pi^{(0)}(x_n)$ et du gradient par rapport à θ_{ijc} sont identiques à celles des équations (2.19) et (2.21). Les fonctions $\pi_{|i}^{(1)}(x_n)$ tels que définis dans l'équation (2.10) peut être re-écrites sous la forme :

Algorithme 2.1 Pseudo-code pour l'apprentissage du modèle HMD1

Données : $\mathbf{X} = \{\mathtt{x}_1, \cdots, \mathtt{x}_N\} \in \mathbb{R}^{D\times N}$, $\mathbf{Y} = (y_1, \cdots, y_N)^T \in \mathbb{R}^N$, K, M_i et C.
Résultats : Retourne $argmax_{c\in\{1,\cdots,C\}}\, p(y = c|\Omega, x)$ pour une instance x.
1. Initialisation :
   (a) Estimer les paramètres $\{\mu_i, \Sigma_i\}_{i=1}^{K}$ avec l'algorithme K-means.
   (b) Choisir aléatoirement les paramètres $\{\alpha_i\}_{i=1}^{K}$ et les normaliser.
   (c) Choisir aléatoirement les paramètres $\{\delta_{ij}\}_{ij=11}^{KM_i}$ et $\{\theta_{ijc}\}_{ijc=111}^{KM_iC}$.
2. E-step : Calculer $Q\left(\Omega,\, \Omega_{(t)}\right)$ via les équations 2.13 à 2.15.
3. M-step : Estimer $\Omega_{(t+1)}$ via les équations 2.17 à 2.21.
4. Répéter les étapes 2 et 3 jusqu'à convergence.

$$\pi_{j|i}^{(1)}\left(x_n\right) = \frac{\exp\left(\eta_{ij}\right)}{\sum\limits_{l=1}^{M_i}\exp\left(\eta_{il}\right)},$$

où $\eta_{ij} = \ln\left(\beta_{ij}\right) - \frac{1}{2}\Big[\left(x_n - \xi_{ij}\right)^T \sigma_{ij}^{-1}\left(x_n - \xi_{ij}\right) + \ln|\sigma_{ij}| + D\ln\left(2\pi\right)\Big]$. Pour s'assurer que β_{ij} et σ_{ijd} sont positifs, nous effectuons les changements de variables $\beta_{ij} = \exp\left(\beta_{ij}^{'}\right)$ et $\sigma_{ijd} = \exp\left(\sigma_{ijd}^{'}\right)$. En remplaçant ces expressions dans l'équation (2.17), les paramètres de $\pi_{j|i}^{(1)}\left(x_n\right)$ peuvent être estimés avec un algorithme d'optimisation non linéaire et les gradients par rapport à $\beta_{ij}^{'}$, ξ_{ijd} et $\sigma_{ijd}^{'}$ sont donnés par :

Pour $i = 1\cdots K$, $j = 1\cdots M_i$ et $d = 1\cdots D$

$$\nabla\beta_{ij}^{'} = \sum_{n=1}^{N} h_{n,i}^{(0)}\left[h_{n,j|i}^{(1)} - \pi_{j|i}^{(1)}\left(x_n\right)\right], \tag{2.22}$$

$$\nabla\xi_{ijd} = \sum_{n=1}^{N} h_{n,i}^{(0)}\left[h_{n,j|i}^{(1)} - \pi_{j|i}^{(1)}\left(x_n\right)\right]\frac{\left(x_{n,d} - \xi_{ij,d}\right)}{\sigma_{ijd}} \tag{2.23}$$

et

$$\nabla\sigma_{ijd}^{'} = \sum_{n=1}^{N} h_{n,i}^{(0)}\left[h_{n,j|i}^{(1)} - \pi_{j|i}^{(1)}\left(x_n\right)\right]\left[\frac{\left(x_{n,d} - \xi_{ij,d}\right)^2}{\sigma_{ijd}} - 1\right]. \tag{2.24}$$

L'indice d désigne la $d-$ème dimension des données. L'algorithme (2.2) résume la procédure d'apprentissage du modèle HMD2.

Algorithme 2.2 Pseudo-code pour l'apprentissage du modèle HMD2

Données : $\mathbf{X} = \{\mathtt{x}_1, \cdots, \mathtt{x}_N\} \in \mathbb{R}^{D\times N}$, $\mathbf{Y} = (y_1, \cdots, y_N)^T \in \mathbb{R}^N$, K, M_i et C.
Résultats : Retourne $argmax_{c\in\{1,\cdots,C\}}\, p(y = c|\Omega, x)$ pour une instance x.
1. Initialisation :
  (a) Appliquer les étapes d'initialisation 1.a et 1.b de l'algorithme (2.1).
  (b) Pour chaque i, choisir aléatoirement les paramètres $\{\beta_{ij}\}_{j=1}^{M_i}$ et les normaliser.
  (c) Avec les données de la $i^{ème}$ région, faire un k-means pour définir les paramètres $\{\xi_{ij}\}_{j=1}^{M_i}$ et $\{\sigma_{ij}\}_{j=1}^{M_i}$
  (c) Choisir aléatoirement les paramètres $\{\theta_{ijc}\}_{ijc=111}^{KM_iC}$
2. E-step : Calculer $Q\left(\Omega, \Omega_{(t)}\right)$ via les équations 2.13 à 2.15
3. M-step : Estimer $\Omega_{(t+1)}$ via les équations 2.17 à 2.19 et 2.21 à 2.24.
4. Répéter les étapes 2 et 3 jusqu'à convergence.

2.2.3 Comment choisir la structure de HMD ?

Bien que nous pensons qu'un choix approprié des fonctions de sélection peut permettre de réduire le nombre d'experts, le problème concernant le choix des valeurs K et M_i demeure. Ceci peut être résolu soit par un choix *a priori*, soit en utilisant un critère comme l'AIC (*Akaike information criterion*) ou par validation croisée. Les deux dernières options deviennent coûteuses en temps de calcul au fur et à mesure que le nombre d'experts augmente. En effet, nous devons évaluer toutes les combinaisons possibles de K et M_i. Une autre approche simple consiste à définir une valeur élevée pour K et M_i et à supprimer de façon itérative les branches qui ont une faible contribution [27, 25]. À partir des probabilités *a posteriori* (Eq. 2.14 et 2.15) nous définissons $S_i = \sum_{n=1}^{N} h_{n,i}^{(0)}/N$ et $S_{ij} = \sum_{n=1}^{N} h_{n,i}^{(0)} h_{n,j|i}^{(1)}/N$ comme étant un indicateur renseignant sur la contribution au score global des fonctions $\pi_{|i}^{(1)}(x_n)$ et de l'expert $p\left(y|x, \Omega_{ij}^{(2)}\right)$. Ce dernier (resp. $\pi_{|i}^{(1)}$) peut être supprimé si S_{ij} (resp. S_i) est en dessous d'un certain seuil. Ce dernier peut soit être unique pour tout les niveaux du modèle HMD soit spécifique à chaque niveau. Le seuil peut aussi varier d'une itération à l'autre. Le choix de la structure du modèle HMD est résumé comme suit :

1. Définir $K = k_{max}$, $M_i = m_{max}$ et choisir un seuil.
2. Exécuter l'algorithme d'apprentissage du modèle HMD1 ou HMD2.
3. Supprimer les branches selon les valeurs de S_i et S_{ij}.

4. Mettre à jour le seuil si nécessaire.
5. Répéter (2)-(4) jusqu'à la stabilité de la structure (faible variation des valeurs K et M_i).

2.3 Résultats Expérimentaux

Pour évaluer les performances de nos modèles (HMD1 et HMD2), nous avons considéré douze collections de données provenant du répertoire *UCI* et *IDA* [84]. Le choix de ces collections est basé sur le fait que les classifieurs linéaires ne peuvent pas distinguer correctement les classes. Le tableau 2.1 résume les principales caractéristiques de ces collections.

Collections	#instances	dimension (D)	#classes
banana	5300	2	2
Thyroid	7200	21	3
Forensic Glass	214	9	6
Pen-digit	10 992	16	10
Vowel	990	13	11
Cancer (Breast)	277	9	2
Heart	270	13	2
Magic	19,020	10	2
Balance scale	625	4	3
Waveform	5000	21	3
Vehicle	946	18	4
Satimage	6 435	4	6

tableau 2.1 – Collections utilisées dans nos études expérimentales.

En général, la méthode d'optimisation non linéaire utilisée pour estimer les paramètres du modèle HME est la méthode *iteratively re-weighted least squares* (IRLS). Cependant, dans le contexte de la classification avec HME, il a été rapporté que cette méthode a des performances médiocres [38]. Par conséquent, nous choisissons d'utiliser la méthode *L-BFGS* [100] pour toutes les opérations d'optimisations non linéaires. Il convient de rappeler que chaque expert est défini comme étant une régression logistique multinomiale. Plusieurs études ont montré l'efficacité de la méthode L-BFGS dans la résolution de ce genre de problème [86]. En particulier dans l'apprentissage du

mélange d'experts, *L-BFGS* surpasse *IRLS* [38]. De plus, L-BFGS nous permet d'éviter de lourds calculs en approximant l'inverse de la matrice Hessienne. Nous avons utilisé l'implémentation *minFunc* [116] de *L-BFGS* en fixant la tolérance de convergence à 10^{-3} avec un nombre d'itérations fixé à 200 au maximum. Nous avons fixé la tolérance de convergence de l'algorithme EM à 10^{-4} avec un nombre d'itérations fixé à 50 au maximum. De plus, vu que l'initialisation est aléatoire, toutes les expériences sont exécutées deux fois afin d'éviter de mauvais maxima locaux. Les mesures de performance sont la justesse (*accuracy* en anglais) et le temps d'apprentissage. La justesse est définie par :

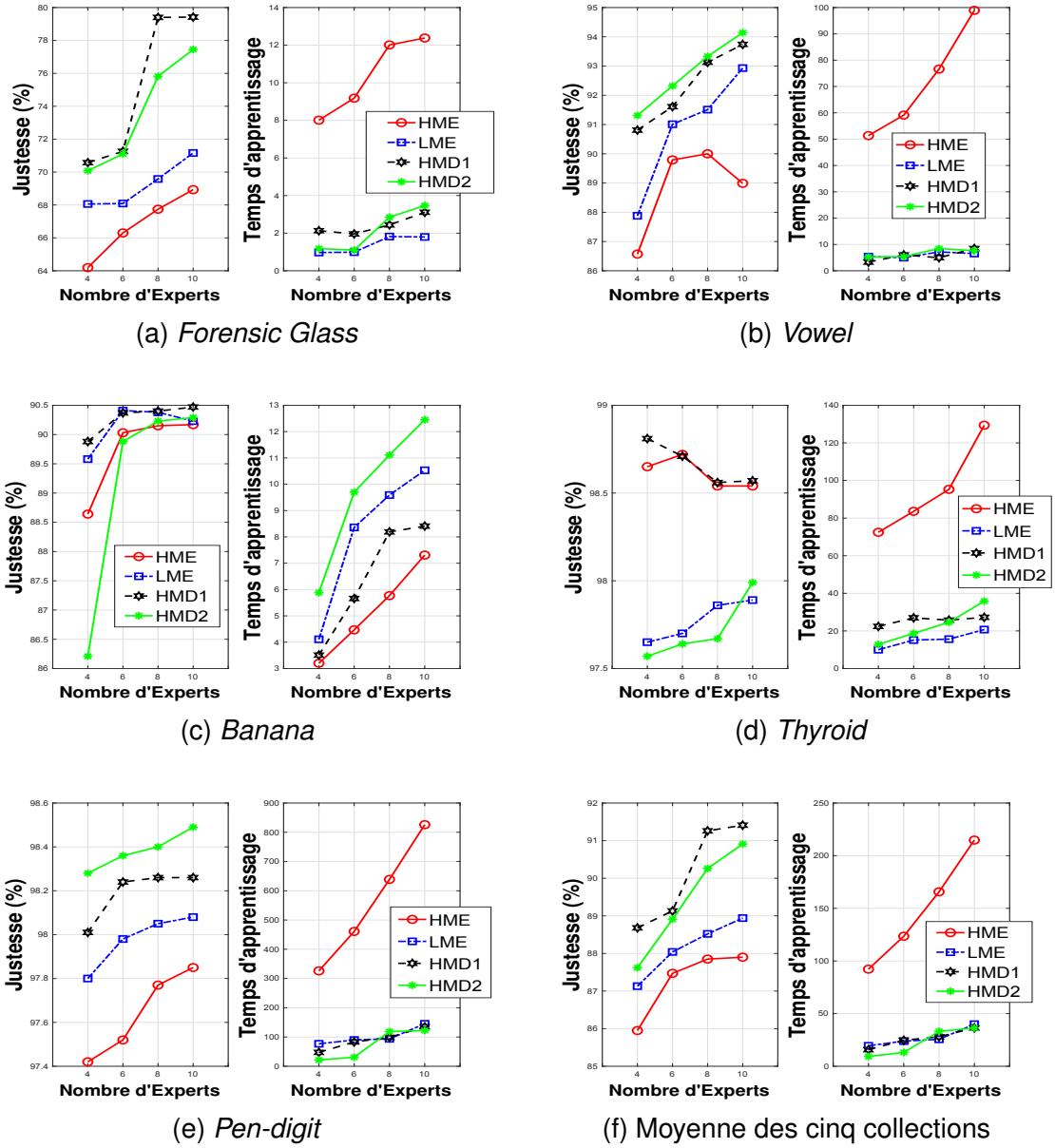
$$\textbf{justesse} = \frac{Nombre\ d'instances\ correctement\ prédites}{Nombre\ total\ d'instances\ \text{à}\ prédire} \times 100. \tag{2.25}$$

Les performances rapportées sont les moyennes arithmétiques des résultats (justesse et temps d'apprentissage) issus de la validation croisée. Le temps d'apprentissage est en secondes. La justesse rapportée est en pourcentage et le chiffre suivant '±' est l'écart-type.

2.3.1 Comparaison entre les modèles HME, LME, HMD1 et HMD2

Dans cette section, nous comparons expérimentalement les modèles HME, LME, HMD1 et HMD2. Ces quatre modèles sont implémentés en utilisant MATLAB et le même type d'experts est utilisé pour chacun d'eux (Eq. 2.4). Le but est de montrer que HMD1 et HMD2 peuvent réduire le nombre d'experts utilisé comparé aux modèles HME et LME. Nous avons évalué ces modèles sur les cinq premières collections décrites dans le tableau (2.1). Une validation croisée à 10 plis a été utilisée pour cette évaluation. Dans une première expérience, nous avons utilisé des experts allant de quatre à dix avec $M_i = 2$. La figure (2.6) représente les justesses et le temps d'apprentissage en fonction du nombre d'experts.

Dans une seconde expérience, nous avons utilisé la procédure de sélection de modèles décrite dans la section (2.2.3). Les valeurs k_{max} et m_{max} sont fixées à 10 et le



(a) *Forensic Glass* (b) *Vowel*

(c) *Banana* (d) *Thyroid*

(e) *Pen-digit* (f) Moyenne des cinq collections

figure 2.6 – Justesse moyenne et temps d'apprentissage évaluée sur les collections *Banana*, *Thyroid*, *Forensic Glass*, *Pen-digit* et *Vowel*.

seuil fixé à 0.01. La justesse de même que le nombre d'experts sélectionné sont reportés dans le tableau (2.2). La meilleure justesse pour chaque nombre d'experts est mis

en gras.

	HME		LME		HMD1		HMD2	
Collections	#Exp.	Justesse	#Exp.	Justesse	#Exp.	Justesse	#Exp.	Justesse
Banana	16	90.17 ± 1.13	34	90.3 ± 0.81	23	**90.36** ± 1.19	**11**	90.13 ± 0.88
Thyroid	40	**99.01** ± 0.32	13	97.57 ± 0.39	11	98.32 ± 0.36	**6**	97.78 ± 0.48
Glass	21	66.85 ± 13.41	22	68.6 ± 10.3	15	71.51 ± 7.46	**10**	**73.46** ± 6.47
Pen-digit	37	98.17 ± 0.27	29	**98.56** ± 0.24	23	98.16 ± 0.28	**18**	98.53 ± 0.34
Vowel	26	92.22 ± 3.09	30	**97.37** ± 0.98	19	94.95 ± 1.3	**14**	96.46 ± 1.19

tableau 2.2 – Justesse et nombre d'experts sélectionné en moyenne pour les collections *Banana*, *Thyroid*, *Forensic Glass*, *Pen-digit* et *Vowel*.

Justesse

Nous discuterons en premier, des résultats de la première expérience (fig. 2.6). En moyenne (fig. 2.6f), même avec dix experts, la justesse obtenue par HME est inférieure à celle obtenue par HMD1 avec quatre experts. De plus, il faut plus de dix experts pour que HME atteigne la justesse obtenue par HMD2 avec six experts. Par ailleurs, LME utilise environ dix experts pour obtenir la justesse obtenue par HMD1 et HMD2 utilisant six experts. Ces résultats varient néanmoins d'une collection à une autre. Par exemple, pour la collection *Forensic Glass* (fig. 2.6a), lorsque huit experts sont utilisés, nous réalisons une justesse de 67.74% avec le modèle HME. Cette justesse chute de 3% lorsque le nombre d'experts est réduit à 4. Au contraire, avec 4 experts, HMD1, HMD2 et LME réalisent une justesse d'au moins 68%. Cependant, il faut au moins dix experts au modèle LME pour obtenir une justesse similaire à celle obtenue par HMD1 et HMD2 avec six experts. Par contre, même avec dix experts, la justesse obtenue par HME est inférieure à celle que les modèles HMD1 et HMD2 ont obtenue avec quatre experts. Des résultats similaires sont obtenus avec la collection *Vowel* (fig. 2.6b) où en général, LME utilise au moins deux experts de plus que HMD1 et

HMD2 pour obtenir une justesse similaire. En ce qui concerne la collection *Banana*, LME et HMD1 réalisent les meilleures performances. La figure 2.6d illustre le fait que sur la collection *thyroid*, HME et HMD1 obtiennent les meilleures justesses. En ce qui concerne la collection *Pen-digit* (fig. 2.6e), même avec dix experts, HME n'obtient pas la justesse obtenue par HMD1 et HMD2 utilisant quatre experts. À l'exception des collections *Vowel* et *Pen-digit*, HMD1 obtient de meilleures justesses comparées à HMD2.

Pour la seconde expérience (tab. 2.2), en termes de justesse, HMD2 réalise des performances similaires à ceux de LME sur les collections *Banana*, *Thyroid* et *Pen-digit*. Cependant, HMD2 et HMD1 ont été surpassés par HME et LME respectivement sur la collection *Thyroid* et *Vowel*. Nous notons que pour toutes les cinq collections (à l'exception de *banana* dans le cas de HMD1), HMD1 et HMD2 ont sélectionné le plus petit nombre d'experts. Cependant, nous notons que le nombre d'experts sélectionné par HMD1 est supérieur à celui de HMD2. Cette différence est due au fait que le nombre M_i sélectionné par HMD1 est toujours supérieur à celui de HMD2 bien que le contraire soit observé pour le nombre K. En effet, contrairement au modèle HMD1, la plupart des valeurs de S_{ij} obtenues pour le modèle HMD2 sont faibles. Par conséquent, nous suggérons l'utilisation de deux seuils : l'un appliqué à S_i et l'autre à S_{ij}. Plus précisément, dans le cas de HMD1, le seuil pour S_{ij} peut augmenter d'une itération à une autre afin de supprimer plus d'experts. Cependant, une valeur maximale doit être imposée au seuil de S_{ij} afin d'éviter la suppression de tous les experts au sein du modèle HMD1.

À partir des résultats de ces deux expériences, nous suggérons que les performances de HMD1 sont dues à l'utilisation de différents type de fonctions de sélection au sein du mélange hiérarchique. De plus, ces résultats confirment que l'utilisation du modèle de mélange hiérarchique peut être utile dans le cas de certaines collections.

Complexité algorithmique

Dans le cas où $M_i = M$, soit N_e le nombre d'experts ($N_e = MK$). Nous avons le nombre suivant d'optimisations non linéaires : $[N_e + K + 1]T$ pour HME, N_eT pour LME, $[N_e+K]T$ pour HMD1 et HMD2 où T est le nombre d'itérations de l'algorithme EM. Bien que LME résolve le plus petit nombre d'optimisations non linéaires, il n'est

pas toujours le plus rapide à entraîner. En effet, nous notons dans nos expériences que pour tous les modèles, le temps d'apprentissage est régie par T et le nombre d'itérations de l'algorithme L-BFGS. Généralement, HME converge après un grand nombre d'itérations de l'algorithme EM ce qui explique son temps d'apprentissage élevée. À partir de la figure (2.6), nous pouvons remarquer qu'en moyenne, HMD2 est plus rapide que les autres algorithmes. Ceci s'explique par le fait que HMD2 converge après un petit nombre d'itérations de l'algorithme EM comparé aux trois autres modèles. Cependant, quand le nombre d'experts est élevée (ex. $N_e = 100$), le nombre d'optimisations non linéaires devient le facteur majeur dans l'évaluation du temps d'apprentissage. Par conséquent, avec un grand nombre d'experts N_e, les modèles les plus rapides à entraîner sont dans l'ordre croissant : HME, HMD2, HMD1 et LME. En outre, pour les quatre modèles, le temps d'apprentissage croît presque linéairement avec le nombre d'instances N. Ces résultats nous suggèrent que l'on peut utiliser HMD2 pour réduire le temps d'apprentissage lorsque nous sommes en présence d'un petit nombre d'experts. Par ailleurs, lorsque nous sommes en présence d'un grand nombre d'experts, HMD1 apparaît comme un compromis entre une justesse élevée et un temps d'apprentissage raisonnable.

2.3.2 Comparaison de HMD1 et HMD2 avec d'autres méthodes de l'état de l'art

Dans cette section, nous comparons nos modèles (HMD1 et HMD2) avec quelques méthodes pertinentes de l'état de l'art relative à la classification non linéaire. Nous avons mené nos expériences sur les douze collections mentionnées plus haut (tab. 2.1). Ces expériences ont été réalisées à l'aide d'une validation croisée à cinq plis.

Comparaison avec les méthodes classiques de l'état de l'art

Les méthodes classiques de l'état de l'art utilisées dans notre étude sont : la régression logistique à noyaux (RLN) appliquée avec un noyau gaussien, le SVM, le perceptron multicouche (MLP), l'arbre de décision C4.5 [106], le réseau RBF (RBFN), *random subspace* (RS) [65] et *adaboost* [55]. Le SVM a été exécuté avec la librairie *libsvm* [37]. Nous avons implémenté notre propre code MATLAB pour la méthode

RLN. Pour le reste des méthodes, nous avons adopté l'implémentation fournie par Weka 3.9 [64]. Pour réduire le surapprentissage, nous avons ajouté une régularisation de type L2 aux experts de nos deux modèles HMD1 et HMD2. Nous avons aussi défini le seuil pour S_i à 0.1 et celui pour S_{ij} à 0.01. Les valeurs des seuils S_i et S_{ij} ont été sélectionnées respectivement dans les intervalles $[0, 1]$ (pas de 0.1) et $[0, 0.1]$ (pas de 0.01). Dans le cas de HMD1, après chaque itération de la procédure définie dans la section 2.2.3, nous augmentons la valeur du seuil pour S_{ij} en le multipliant par 1.6 avec 0.1 comme valeur maximale. Nous avons testé plusieurs facteur multiplicatif dans l'intervalle $[1, 2]$ avec un pas de 0.1.

Collections / Modèles	Glass	Cancer	Banana	Balance	Pen.	Magic	Veh.	Wav.	Sat.	Heart	Thy.	Vow.	Justesse Moyenne
HMD1	75.25 ± 3.40 (5)	**76.19** ± 3.29 (3)	**90.36** ± 0.29 (5)	95.18 ± 2.1 (8)	98.91 ± 0.12 (4)	**86.81** ± 0.96 (5)	**83.57** ± 0.8 (4)	**86.8** ± 1.38 (4)	85.24 ± 0.69 (2)	83.7 ± 6.2 (2)	97.36 ± 0.42 (2)	94.75 ± 1.27 (3)	**87.71**
HMD2	75.71 ± 3.00 (5)	**76.19** ± 3.03 (3)	90 ± 0.32 (5)	94.07 ± 1.26 (8)	98.92 ± 0.28 (4)	85.08 ± 0.7 (4)	83.45 ± 1.13 (4)	**86.7** ± 1.39 (4)	85.24 ± 0.69 (2)	**84.44** ± 5.94 (4)	97.36 ± 0.42 (2)	94.65 ± 0.77 (4)	87.65
Adaboost	**76.97** ± 6.03 (60)	68.37 ± 5.15 (100)	88.26 ± 0.93 (20)	78.72 ± 3.58 (4)	98.91 ± 0.23 (10)	86.2 ± 3.15 (10)	77.49 ± 2.68 (100)	84.64 ± 1.20 (10)	83.96 ± 0.86 (2)	81.41 ± 4.22 (80)	99.63 ± 0.22 (4)	96.21 ± 1.39 (100)	85.06
RS	75.48 ± 5.61 (100)	74.66 ± 3.86 (100)	70.85 ± 1.41 (60)	84.06 ± 2.46 (90)	97.67 ± 1.26 (60)	85.51 ± 2.63 (100)	75 ± 2.35 (90)	84.83 ± 1.06 (10)	84.5 ± 0.76 (100)	83.26 ± 3.94 (80)	97.67 ± 1.26 (60)	94.19 ± 2.02 (100)	83.97
MLP	69.87 ± 5.88 (18)	74.36 ± 5.16 (2)	89.79 ± 0.46 (17)	**96.24** ± 2.03 (18)	95.02 ± 0.26 (20)	86.52 ± 0.47 (16)	82.57 ± 2.76 (15)	85.44 ± 0.88 (5)	**85.99** ± 0.7 (14)	81.30 ± 4.84 (10)	97.36 ± 0.42 (14)	89.9 ± 2.5 (18)	86.19
RBFN	54.87 ± 8.55 (14)	72.83 ± 5.28 (3)	89.96 ± 0.87 (14)	85.14 ± 2.79 (14)	98.32 ± 0.32 (14)	83.39 ± 0.61 (14)	70.78 ± 2.54 (14)	85.78 ± 1.1 (5)	85.5 ± 2.54 (5)	78.52 ± 2.54 (14)	94.03 ± 0.55 (14)	90.83 ± 3.39 (14)	82.49
RLN	67.29 ± 5.86	69.32 ± 2.41	90.13 ± 0.21	91.68 ± 2.3	**99.40** ± 0.15	79.86 ± 0.28	67.96 ± 1.98	81.41 ± 1.04	83.01 ± 0.57	56.66 ± 1.01	94 ± 0.5	**99.09** ± 0.75	81.65
SVM	65.23 ± 5.54 (gaussien)	74.38 ± 3.93 (gaussien)	90.31 ± 0.73 (gaussien)	91.68 ± 2.07 (lin.)	97.77 ± 0.3 (lin)	65.74 ± 0.13 (gaussien)	79.49 ± 2.67 (lin)	86.15 ± 1.14 (gaussien)	84.79 ± 0.82 (lin)	83.59 ± 4.98 (lin)	92.58 ± 0.07	88.1 ± 3.26	83.31
C4.5	66.82 ± 7.26	74.74 ± 5.17	89.04 ± 0.98	78.24 ± 2.59	96.35 ± 0.37	85.07 ± 0.51	72.80 ± 2.64	76.89 ± 1.2	85.08 ± 0.74	81.30 ± 6.13	**99.64** ± 0.21	83.14 ± 4.21	82.42

tableau 2.3 – Comparaison de HMD1, HMD2, *Adaboost*, *Random Subspace*, MLP, RBFNN, RLN, SVM et C4.5.

Dans nos expériences avec *adaboost* et *random subspace* (RS), nous avons premièrement choisi une régression logistique comme classifieur de base mais les résultats obtenus étaient médiocres. Par conséquent, des arbres C4.5 non réduits ont été choisis comme classifieur de base. Nous avons évalué un ensemble de deux à cent arbres C4.5 pour chacune des méthodes *adaboost* et *random rubspace*. Il est important de rappeler que l'arbre C4.5 peut produire des frontières non linéaires entre les classes ce qui n'est pas le cas de la régression logistique utilisée comme expert dans nos modèles. Notre principal but concernant ces deux méthodes (RS et *adaboost*) est de les comparer avec nos modèles au regard du nombre de classifieurs utilisés de même que de la justesse obtenue. Pour les autres méthodes de l'état de l'art, les performances sont évaluées en termes de justesse. Les classifieurs MLP et RBFN sont entraînés avec un ensemble d'unités cachées allant de deux à vingt. La configuration qui donne la plus grande justesse a été retenue.

Nous résumons les résultats obtenus dans le tableau 2.3. Le contenu des parenthèses représente respectivement le nombre d'arbres utilisés pour Adaboost et *Random Subspace*, le nombre d'unités cachées pour les méthodes de type RNA, le type de noyau utilisé pour le SVM et le nombre d'experts moyen utilisé dans nos modèles. La dernière colonne du tableau 2.3 représente la justesse moyenne obtenue par chaque méthode (la meilleur est mis en gras). La meilleure justesse par collection est mise également en gras. Les résultats montrent que HMD1 et HMD2 surpassent *Adaboost* sur sept collections. Plus précisément, sur les collections *Cancer* et *Vehicle*, même avec 100 arbres C4.5, la justesse obtenue avec *Adaboost* reste inférieure à celle de nos modèles (environ quatre experts ou moins ont été utilisés). Un résultat similaire est noté lorsque nous comparons RS à nos modèles. En effet, HMD1 et HMD2 ont une justesse supérieure à celle de RS sur huit collections. En général, l'écart-type obtenu par HMD1 et HMD2 est inférieure à celle obtenue par les méthodes *Adaboost* et RS. Nous pouvons donc utiliser peu d'experts linéaires au sein de nos modèles, tout en obtenant une faible erreur de classification comparée à *Adaboost* et RS. Cependant, le temps d'apprentissage de nos modèles peut être long étant donné que nous devons apprendre les paramètres des fonctions de sélection [1]. En ce qui concerne MLP et RBFN,

1. Nous n'avons pas effectué une comparaison en termes de temps d'apprentissage. En effet, le temps d'exécution d'un code matlab non optimisé ne peut pas être directement comparée à celui

HMD1 et HMD2 ont une meilleure justesse qu'eux sur au moins huit collections. La méthode RLN a obtenu une justesse supérieure à celles obtenues par nos modèles sur uniquement les collections *Vowel* et *Pen-digit* (moins de 1% sur cette dernière). Par contre, dans nos expériences, SVM n'a pas surpassé nos modèles. Les résultats obtenus sur les collections *Banana*, *Satimage* et *Thyroid* montrent qu'un arbre C4.5 réduit peut être meilleur qu'un ensemble d'arbres C4.5 non réduits. L'arbre C4.5 réduit a obtenu une meilleure justesse que nos modèles uniquement sur la collection *Thyroid*. À partir du tableau (2.3), nous notons qu'en moyenne, HMD1 et HMD2 obtiennent les plus grandes justesses avec un meilleur écart-type sur au moins la moitié des collections utilisées.

Comparaison avec diverses méthodes d'ensemble

Nous avons également comparé nos modèles avec de récentes méthodes d'ensemble [76, 91, 83, 109, 75, 136, 42, 53, 1, 2]. La méthode DCE-CC sélectionne dynamiquement les classifieurs de base (les plus proches voisins et l'arbre C4.5) en fonction de leur compétence [83]. Une approche similaire est appliquée par la méthode META-DES [42] où cinq méta-caractéristiques et méta-classifieurs ont été utilisés pour dynamiquement sélectionner le classifieur de base le plus compétent. La méthode MFSE [75] est construit en trois étapes. À la première, un sous-ensemble optimal de caractéristiques est sélectionné par apprentissage de plusieurs classifieurs MLP. À la seconde étape, un classifieur MLP est entrainé sur chaque sous-ensemble sélectionné. Finalement, un mélange d'experts est entrainé en utilisant le classifieur MLP comme fonction de sélection et comme experts. Les paramètres estimés lors de la deuxième étape sont utilisés pour initialiser chaque expert. La méthode MARK-ELM [53] combine par *boosting*, différent classifieurs se basant sur la méthode du noyau. La méthode *NULCOEC* (*Non-Uniform Layered Cluster Oriented Ensemble Classifier*) [109] partitionne les données en plusieurs groupes (*clusters*). Un ensemble de classifieurs (SVM avec le noyau gaussien) est alors entrainé sur chaque groupe et le vote de la majorité est utilisé pour la décision finale. Pour une autre méthode d'ensemble nommée BABORO R [76], les auteurs combinent les méthodes *bagging*, *boosting*, *rotation forest* et

d'un code Java hautement optimisé.

random subspace pour classifier les données. Les deux plus récentes méthodes d'ensemble *RTQRT-ME* et *R-RTQRT-ME* [1, 2] sont des mélanges utilisant à la fois un modèle MLP comme fonction de sélection et comme experts. Un terme de régularisation est aussi introduit afin d'accroître la diversité des experts. Dans notre expérience, nous avons utilisé le même nombre d'experts (cinq) que les auteurs dans [1, 2]. La structure de nos modèles est déterminé par une validation croisée à cinq plis. Nous reportons dans le tableau (2.4), la justesse obtenue pour chaque collection.

Le modèle HMD1 obtient la meilleure justesse sur sept collections tandis que R-RTQRT-ME est le meilleur sur deux collections. Le modèle HMD1 et R-RTQRT-ME ont la même justesse sur la collection *Magic*. Quant au modèle HMD2, il obtient la meilleure justesse sur la collection *heart*. L'amélioration faite par nos modèles varie de 1% à 30%. Par exemple, sur la collection *vehicle*, HMD1 améliore la justesse de près de 16.4% comparé à NULCOEC [109], 10.8% comparé à MARK-ELM [53], 10.32% comparé à LODE [91], 4.6% comparé à META-DES [42], 3.6% comparé à MFSE [75] et moins de 0.5% comparé à RTQRT-ME et R-RTQRT-ME. Cependant, l'amélioration faite par HMD1 lorsque nous le comparons à la plus efficace des méthodes (R-RTQRT-ME) ne dépasse pas 2%. En effet, nous avons utilisé cinq régressions logistique dans nos modèles tandis que R-RTQRT-ME utilise cinq classifieurs MLP (ayant au moins cinq unités cachées). Il est donc intéressant de remarquer qu'avec un choix approprié des fonctions de sélection et un ensemble de régressions logistiques, nous pouvons fair mieux qu'une combinaison de classifieurs non linéaires (MLP). Comme noté dans les sections précédentes, HMD1 est meilleur que HMD2 en termes de justesse. Les résultats obtenus par HMD1 sont probablement dûs au fait que les fonctions de sélection utilisées, «collaborent» afin d'améliorer les performences de chaque expert.

Coll.	Méthodes	Just. (%)
Glass	META-DES [42]	66.87
	MFSE [75]	72.22
	DCE-CC [83]	72.48
	NULCOEC [109]	**91.82**
	RTQRT-ME [1]	72.87
	R-RTQRT-ME [2]	74.76
	HDM1	75.25
	HMD2	75.71
Cancer	BABOROR[76]	73.16
	RTQRT-ME [1]	75.74
	R-RTQRT-ME [2]	**81.02**
	HDM1	**81.24**
	HMD2	79.80
Banana	R-RTQRT-ME [2]	89.81
	HDM1	**90.36**
	HMD2	90
Balance	DCE-CC [83]	75.18
	RTQRT-ME [1]	92.96
	ISCG-Ranking[136]	78.88
	HDM1	**94.08**
	HMD2	93.44
Pen-digit	MFSE [75]	98.27
	RTQRT-ME [1]	97.13
	R-RTQRT-ME [2]	97.84
	HDM1	**98.91**
	HMD2	98.25
Magic	ISCG-Ranking[136]	82.33
	R-RTQRT-ME [2]	**86.82**
	HDM1	**86.81**
	HMD2	85.82
Vehicle	META-DES [42]	82.75
	MARK-ELM [53]	76.54
	MFSE [75]	83.73
	NULCOEC [109]	70.87
	LODE [91]	77.03

suite

Coll.	Méthodes	Just. (%)
	RTQRT-ME [1]	87.11
	R-RTQRT-ME [2]	**87.29**
	HDM1	**87.35**
	HMD2	86.17
Waveform	BABOROR[76]	84.29
	RTQRT-ME [1]	87.96
	R-RTQRT-ME [2]	**89.72**
	HDM1	87.26
	HMD2	87
Satimage	MFSE [75]	85.33
	RTQRT-ME [1]	87.44
	R-RTQRT-ME [2]	**89.61**
	HDM1	86.39
	HMD2	86.2
Heart	META-DES [42]	84.80
	DCE-CC [83]	80
	BABOROR[76]	81.74
	RTQRT-ME [1]	84.41
	R-RTQRT-ME [2]	87.4
	HDM1	88.15
	HMD2	**88.52**
Thyroid	ISCG-Ranking[136]	95.44
	HDM1	**98.58**
	HMD2	97.57
Vowel	BABOROR[76]	94.6
	HDM1	**97.37**
	HMD2	96.76

tableau 2.4 – Comparaison avec d'autres méthodes d'ensemble.

2.4 Conclusion

Dans ce chapitre, nous avons proposé un modèle général pour le mélange hiérarchique de classifieurs (HMD). L'idée principale derrière ce modèle est de diviser une tâche complexe de classification en plusieurs sous-tâches à travers un ensemble de fonctions de sélection. Chaque sous-tâche est alors assignée à un classifieur qui joue le rôle d'expert. Afin de répondre à la question de savoir comment réduire le nombre d'experts utilisé, nous avons tiré du modèle général, deux exemples nommés HMD1 et HMD2. Ces deux exemples sont basés sur deux différentes stratégies de modélisation des fonctions de sélection. La première modélise les fonctions de sélection en utilisant à la fois un modèle non linéaire et un modèle linéaire (HMD1) tandis que la deuxième utilise uniquement un modèle non linéaire (HMD2). Ce faisant, HMD1 et HMD2 permettent une meilleure organisation des experts réduisant ainsi leur nombre. En outre, l'utilisation d'experts linéaires au sein de HMD1 et HMD2 permet une meilleure explication de leurs prédictions. Plusieurs expériences ont été menées afin de montrer la capacité de ces deux modèles à réduire le nombre d'experts utilisés. De plus, HMD1 et HMD2 surpassent (en termes de justesse) plusieurs méthodes de l'état de l'art. Nous avons également noté qu'avec HMD1 et HMD2, nous pouvons utiliser peu de d'experts comparé à d'autres méthodes d'ensemble qui utilisent des experts non linéaires. Dans le chapitre suivant, nous nous intéressons à la classification des données proportionnelles.

Chapitre 3

Mélange hiérarchique de classifieurs discriminatifs : cas des données proportionnelles

Les données proportionnelles sont présentes dans plusieurs domaines tels que l'écologie, la géologie, l'économie et le traitement d'images (ex : molarité, pourcentage de revenus, histogramme). Elles représentent des portions d'un ensemble. Ces données sont bornées, positives et multidimensionnelles. De plus, les données proportionnelles sont soumises à une contrainte : la somme des éléments d'un vecteur proportionnel est égale à une constante. Géométriquement, les données proportionnelles sont à l'intérieur d'un simplexe (généralisation du triangle). Cela étant, l'analyse de ces données ne peut se faire avec des méthodes développées pour des données non restreintes à un simplexe [6]. Plusieurs modèles génératifs de mélange ont été exploités pour la classification non supervisée (*clustering*) des données proportionnelles [28, 23, 52, 51]. En ce qui concerne l'utilisation des modèles discriminatifs pour la classification des données proportionnelles, deux approches ont été proposées dans la littérature. La première consiste à appliquer un prétraitement afin d'obtenir des données non restreintes au simplexe [6, 129, 61]. Suite à ce prétraitement, un modèle discriminatif standard est appliqué sur les données transformées. La deuxième approche consiste à construire un noyau adéquat [20–22, 29] suivi de l'utilisation du SVM pour classifier les données.

Cependant, les modèles résultants de ces deux approches sont difficiles à interpréter. De plus, l'extraction d'un noyau pour le traitement des données proportionnelles est très couteuse en temps de calcul [29]. En effet, une étape préliminaire à la formation de ces noyaux consiste à décrire chaque instance de données à l'aide d'un modèle de mélange. Contrairement aux travaux précédents, nous proposons d'utiliser une approche purement discriminative en modélisant directement la distribution *a posteriori* des classes. Cette distribution est construite à l'aide de la Dirichlet généralisée (DG). Ce choix est justifié par le fait que la Dirichlet généralisée est assez flexible et bien adaptée pour la modélisation des données proportionnelles [52, 51]. Nous appelons ce modèle *Discriminative Generalized Dirichlet* (DGD). Au mieux de nos connaissances, cette distribution *a posteriori* n'a jamais été utilisée comme classifieur discriminatif [1]. Cela peut être dû au fait que le terme du mélange de DG qui apparaît dans la distribution *a posteriori* donne une vraisemblance difficile à utiliser. Dans ce chapitre, nous établirons une borne supérieure pour le mélange de Dirichlet généralisée afin d'aller au-delà de cette limitation (voir section 3.2.1).

En présence d'une tâche complexe de classification, il est possible de combiner plusieurs classifieurs. L'objectif ici est de permettre à chaque classifieur de se focaliser sur une sous-tâche spécifique. Dans ce cas, il est possible d'utiliser des méthodes d'ensemble comme le *bagging* [31] ou le *random subspace* [65]. Nous pouvons nous baser également sur le modèle de mélange décrit au chapitre 2. En effet, le modèle DGD peut être hiérarchiquement combiné selon le paradigme du modèle HMD en utilisant comme fonction de sélection un modèle DGD. Nous avons évalué le modèle DGD, de même que son mélange hiérarchique à travers trois expériences. La première a été réalisée sur plusieurs collections du répertoire UCI [84]. Les deux autres expériences sont respectivement, la détection de courriers indésirables et l'identification de l'espace de couleur d'une image. Ce chapitre résume trois articles dont deux sont soumis dans des journaux et une présentée et publiée dans une conférence. Le premier article intitulé *Classification using mixture of discriminative learners : The case of compositional data* a été publié dans *International Conference Image Analysis and Recognition* en

1. En général, la probabilité *a posteriori* est calculée après que les paramètres du mélange de DG soient estimés. Mais dans ce travail, les paramètres sont estimés directement à partir de la distribution *a posteriori*.

2017. Les deux autres articles intitulés respectivement *Identifying color space for improved image display* et *Hierarchical mixture of discriminative Generalized Dirichlet classifiers* ont été soumis dans les journaux *Pattern Recognition Letters* et *Pattern Recognition* en février 2021.

Le reste de ce chapitre est structuré comme suit. La section 3.1 présente le modèle DGD de même que son mélange hiérarchique. La section 3.2 décrit l'estimation des paramètres de même que l'établissement d'une borne supérieure du mélange de Dirichlet généralisée. La section 3.3 présente l'évaluation des modèles à travers les résultats expérimentaux.

3.1 Modèle de classification basé sur la Dirichlet généralisée

3.1.1 La distribution de Dirichlet généralisée

Soit $x = \{(x_1, \cdots, x_{D+1})^T \in \mathbb{R}^{D+1} \mid \sum_{d=1}^{D+1} x_d = A;\ A \geq 1\}$ un vecteur aléatoire suivant une distribution de Dirichlet généralisée (DG) ayant pour densité :

$$\begin{aligned} &GD(x|\mu) = \prod_{d=1}^{D} \frac{x_d^{a_d-1}[A - \sum_{l=1}^{d} x_l]^{\gamma_d}}{A^{(a_d+b_d-1)}\mathtt{B}(a_d, b_d)}; \\ &0 < x_d < A,\ a_d,\ b_d > 0; \\ &\gamma_d = b_d - (a_{d+1} + b_{d+1})\ pour\ d = 1 \cdots D-1; \\ &\gamma_D = b_D - 1 \\ &\mu = (\mu_1, \cdots, \mu_D)^T;\ \mu_d = (a_d, b_d) \end{aligned}, \tag{3.1}$$

où $(.)^T$ est l'opération de transposition, $\mathtt{B}(a_d, b_d)$ est la fonction Beta. La distribution de Dirichlet généralisée étant définie sur le simplexe $S_D = \{(x_1, \cdots, x_D), \sum_{d=1}^{D} x_d < A\}$, elle est un choix naturel pour la modélisation des données proportionnelles. Notons que la distribution de Dirichlet généralisée est équivalent à la distribution de Dirichlet lorsque $b_{d-1} = (a_d + b_d)$. Cependant, cette dernière a une covariance négative contrairement à la Dirichlet généralisée qui à une covariance plus générale. La dis-

tribution de Dirichlet généralisée a été largement utilisée dans les modèles génératifs pour la classification des données proportionnelles [26, 52]. Elle a été aussi utilisée pour générer des noyaux [21, 22, 29]. À travers une propriété mathématique de la distribution de Dirichlet généralisée, l'équation 3.1 peut être exprimée en termes de variables indépendantes [30, 24, 52] :

$$\begin{aligned} GD(x|\mu) = & \prod_{d=1}^{D} \texttt{Beta}(\texttt{v}_d|a_d, b_d) \\ = & \prod_{d=1}^{D} \frac{\texttt{v}_d^{a_d-1}(A-\texttt{v}_d)^{b_d-1}}{A^{(a_d+b_d-1)}\texttt{B}(a_d, b_d)} \end{aligned}, \tag{3.2}$$

où $\texttt{Beta}(\texttt{v}_d|a_d, b_d)$ est la distribution Beta avec pour paramètres $\{a_d, b_d\}$. Le changement de variable effectué est $\texttt{v}_1 = x_1$ et $\texttt{v}_d = \dfrac{x_d}{A - \sum_{l=1}^{d-1} x_l}$.

3.1.2 Classifieur discriminatif basé sur la Dirichlet généralisée

Soit le problème de classification illustré par la figure 3.1 où les données proportionnelles sont réparties en deux classes («o» en rouge et «∗»en bleu). Ces données ont été générées en considérant que chaque classe provient d'un mélange de Dirichlet.

En général dans la littérature, les données proportionnelles sont prétraitées avant l'utilisation d'une méthode de classification. Le prétraitement consiste à appliquer une transformation aux données. Le but de cette dernière est d'obtenir des données qui ne se trouvent pas dans un simplexe. Tsagris *et al.* [129] ont proposé une transformation appelée alpha-transformation. Cette dernière est une généralisation de précédentes transformations utilisées dans la littérature et est décrite par :

$$\mathbf{z}_\alpha(x) = \mathbf{H} \cdot \left(\frac{(D+1)\mathbf{u}_\alpha(x) - \mathbf{1}_{(D+1)}}{\alpha} \right), \tag{3.3}$$

où $\alpha > 0$, $\mathbf{1}_{(D+1)}$ est le vecteur unitaire de dimensions $D+1$, la matrice $\mathbf{H}$ est obtenue en supprimant la première ligne de la matrice de Helmert [81] et

$$\mathbf{u}_\alpha(x) = \left(\frac{x_1^\alpha}{\sum_{d=1}^{D+1} x_d^\alpha}, \cdots, \frac{x_{D+1}^\alpha}{\sum_{d=1}^{D+1} x_d^\alpha} \right)^T \tag{3.4}$$

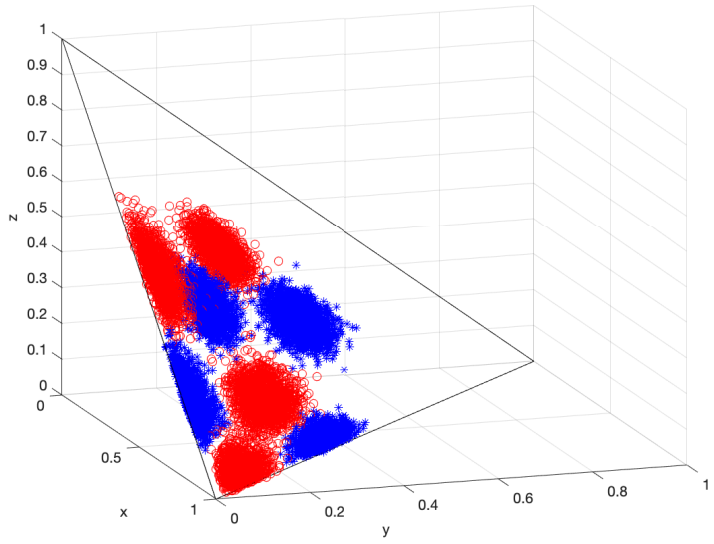


figure 3.1 – Illustration de données proportionnelles.

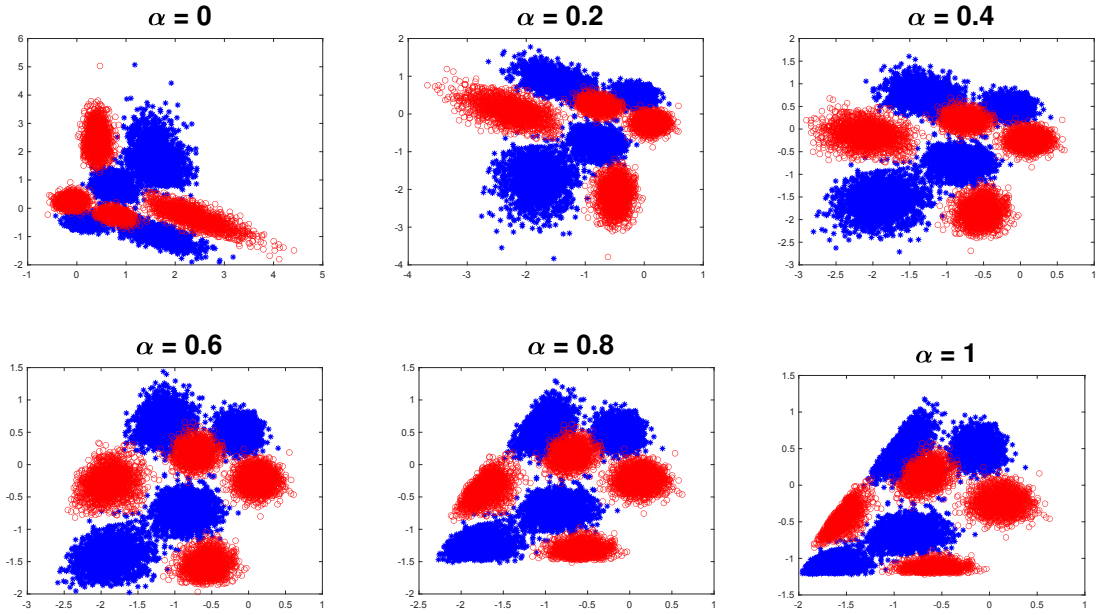


figure 3.2 – Illustration de l'alpha-transformation pour différentes valeurs de α.

est la transformation de puissance [6]. Dans le cas où $\alpha \to 0$, Tsagris *et al.* ont

démontré que :

$$\begin{aligned} \mathbf{z}_0(x) = & \lim_{\alpha \to 0} \mathbf{z}_\alpha(x) \\ = & \mathbf{ilr}(x) \\ = & \mathbf{H} \cdot \mathbf{clr}(x) \end{aligned} \tag{3.5}$$

où

$$\begin{aligned} \mathbf{clr}(x) = & \left(\ln \left\{ \frac{x_1}{g(x)} \right\}, \ldots, \ln \left\{ \frac{x_{(D+1)}}{g(x)} \right\} \right)^T \\ g(x) = & \prod_{i=1}^{D+1} x_i^{1/(D+1)} \end{aligned} \tag{3.6}$$

et $\mathbf{ilr}(x)$ est la transformation isométrique du log-ratio [48] et $\mathbf{clr}(x)$ est la transformation centrée du log-ratio [6]. La figure 3.2 illustre l'effet de la valeur α sur l'alpha-transformation.

Dans ce chapitre, nous proposons un modèle de classification des données proportionnelles. Ce modèle est appliqué directement sur des données définie dans le simplexe. Soit α_c la probabilité *a priori* que le vecteur x appartienne la classe c ($p(y = c)$). Supposons que chaque classe est décrite par une distribution de Dirichlet généralisée : $p(x|\mu_c) = GD(x|\mu_c)$. La probabilité *a posteriori* peut être utilisée pour modéliser un classifieur pour les données proportionnelles :

$$p(y = c|x, \Omega) = \frac{\alpha_c GD(x|\mu_c)}{\sum_{k=1}^{C} \alpha_k GD(x|\mu_k)} = \pi_c \tag{3.7}$$

où $\alpha_k \geq 0$, $\sum_{k=1}^{C} \alpha_k = 1$, $\Omega = \{\alpha_k, \mu_k\}_{k=1}^{C}$ représente les paramètres de ce classifieur et C est le nombre de classes. Nous appelons ce modèle (eq. 3.7), *Discriminative Generalized Dirichlet* (DGD). En optimisant $p(y = c|x, \Omega)$, nous nous intéressons aux paramètres qui maximisent la discrimination des classes et non leurs descriptions. Les paramètres estimés ne permettent pas nécessairement une description de chaque classe par une Dirichlet généralisée. Ainsi, nous obtenons une estimation moins sensible lorsque les classes ne peuvent être décrites par la Dirichlet généralisée.

3.1.3 Mélange Hiérarchique du classifieur DGD

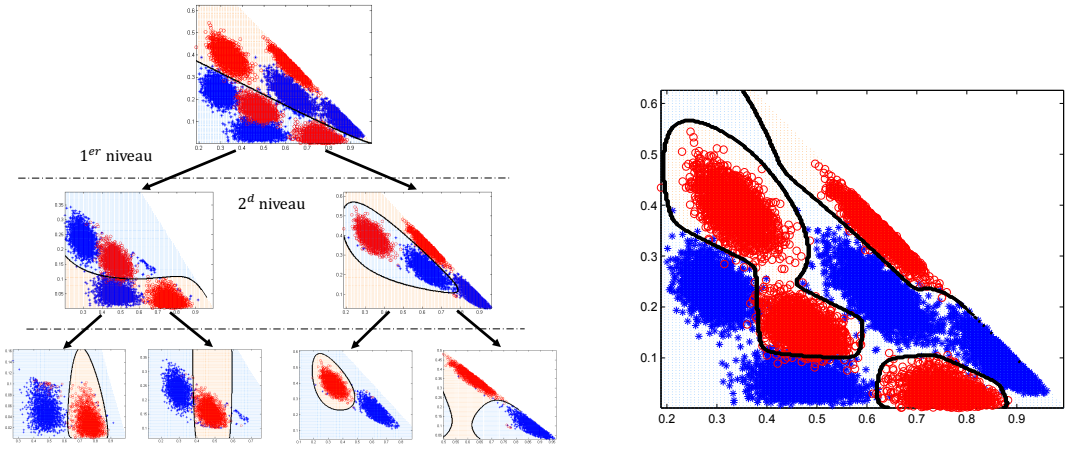
Comme décrit dans le chapitre 2 et dans [128], lorsque nous sommes en face d'une tâche de classification difficile, nous pouvons hiérarchiquement combiner plusieurs classifieurs (experts) afin de former un nouveau classifieur. Il est bien de rappeler que selon ce paradigme, une tâche de classification peut être décomposée de façon récursive en plusieurs sous-tâches. Ceci revient à diviser l'ensemble d'apprentissage en plusieurs sous-ensembles. Chaque expert apprend alors à partir d'un sous-ensemble spécifique. La probabilité globale est obtenue en combinant à l'aide des fonctions de sélection, les probabilités obtenues par chacun des experts. À partir du modèle HMD décrit dans le chapitre 2, nous remarquons que les fonctions de sélection effectuent une tâche de classification. Dans le cas des données proportionnelles nous pouvons utiliser le modèle DGD pour modéliser aussi bien les fonctions de sélection que les experts. Nous avons appelé cette hiérarchie de modèles DGD, *Hierarchical mixture of Discriminative Generalized Dirichlet* (HMGD). En se basant sur l'équation (2.3), nous définissons les expressions suivantes pour le modèle *HMGD* à deux niveaux :

$$\begin{array}{ll} \pi_i^{(0)}(x) = \dfrac{\alpha_i^{(0)} GD(x|\mu_i^{(0)})}{\sum_{k=1}^{K} \alpha_k^{(0)} GD(x|\mu_k^{(0)})}; & \pi_{j|i}^{(1)}(x) = \dfrac{\alpha_{ij}^{(1)} GD(x|\mu_{ij}^{(1)})}{\sum_{k=1}^{M_i} \alpha_{ik}^{(1)} GD(x|\mu_{ik}^{(1)})}; \\ s.\text{à } \alpha_k^{(0)} \geq 0; \sum_{k=1}^{K} \alpha_k^{(0)} = 1 & s.\text{à } \alpha_{ik}^{(1)} \geq 0; \sum_{k=1}^{M_i} \alpha_{ik}^{(1)} = 1 \end{array} \tag{3.8}$$

$$p\left(y = c|x; \Omega_{ij}^{(2)}\right) = \frac{\alpha_{ijc}^{(2)} GD(x|\mu_{ijc}^{(2)})}{\sum_{c=1}^{C} \alpha_{ijc}^{(2)} GD(x|\mu_{ijc}^{(2)})}; \quad s.\text{à } \alpha_{ijc}^{(0)} \geq 0; \sum_{c=1}^{C} \alpha_{ijc}^{(2)} = 1$$

où $\Omega^{(0)} = \left\{\alpha_i^{(0)}, \mu_i^{(0)}\right\}_{i=1}^{K}$ est l'ensemble des paramètres de $\left\{\pi_i^{(0)}\right\}_{i=1}^{K}$.
Pour $i = 1 \cdots K$, $\Omega_i^{(1)} = \left\{\alpha_{ij}^{(1)}, \mu_{ij}^{(1)}\right\}_{j=1}^{M_i}$ est l'ensemble des paramètres de $\left\{\pi_{j|i}^{(1)}\right\}_{j=1}^{M_i}$.
Pour $i = 1 \cdots K$ et $j = 1 \cdots M_i$, $\Omega_{ij}^{(2)} = \left\{\alpha_{ijc}^{(2)}, \mu_{ijc}^{(2)}\right\}_{c=1}^{C}$ est l'ensemble des paramètres de $\left\{p\left(y = c|x; \Omega_{ij}^{(2)}\right)\right\}_{c=1}^{C}$. Les expressions K et $\{M_i\}_{i=1}^{K}$ représentent respectivement

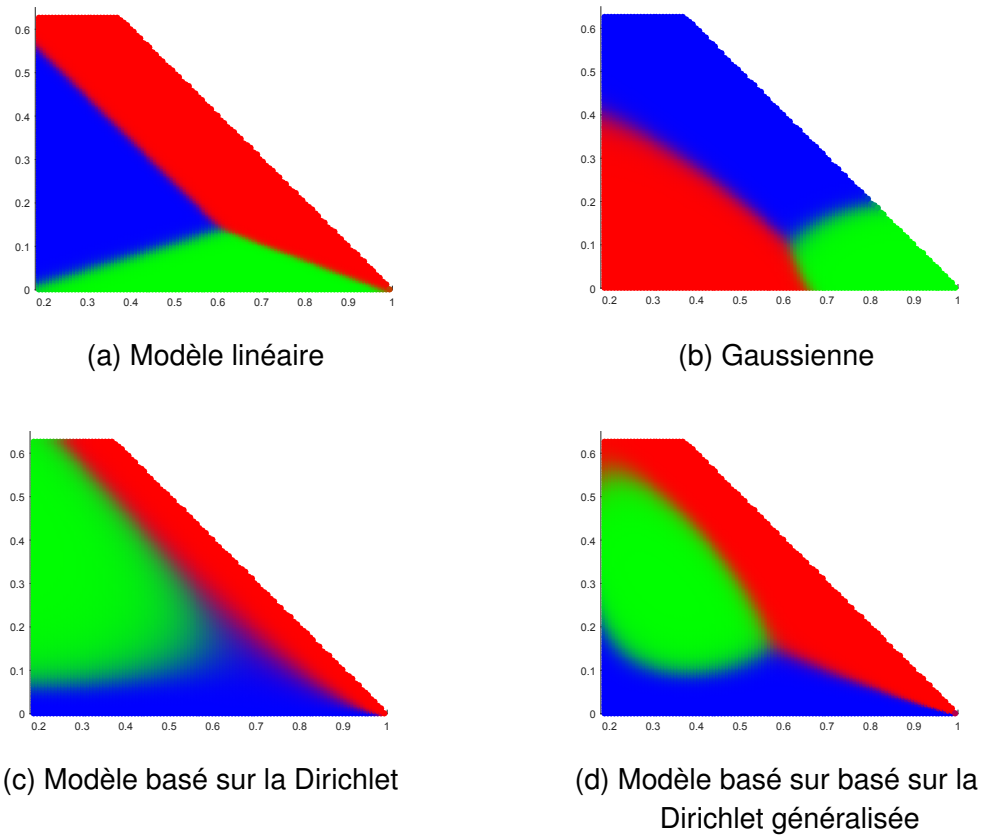
le nombre de composantes au premier et au second niveau du mélange hiérarchique. Le nombre de classes est C. Rappelons que $\left\{\pi_i^{(0)}\right\}_{i=1}^{K}$ et $\left\{\pi_{j|i}^{(1)}\right\}_{ij=1}^{K,M_i}$ sont respectivement les fonctions de sélection au premier et au second niveau du mélange hiérarchique. La figure 3.3 illustre le fonctionnement du modèle HMGD où les courbes en noires délimitent soit les régions, les sous-régions ou les classes. Chaque sous-figure de la figure 3.3b représente une délimitation réalisée par un modèle DGD. Les données sont réparties en deux classes («o» en rouge et «∗» en bleu) et sont les mêmes que celles illustrées à la figure 3.1.



(a) Découpage hiérarchique. (b) Frontière finale.

figure 3.3 – Illustration du fonctionnement du modèle HMGD.

Le modèle *HMGD* est une amélioration du modèle *DME* que nous avons proposé dans [127]. Pour le modèle *DME*, il n'y a pas d'hiérarchie et les fonctions de sélection sont plutôt basées sur la distribution de Dirichlet. De plus, le modèle *DME* utilise des régressions logistiques comme experts. La figure 3.4 illustre le découpage effectué par quelques fonctions de sélection dans le cas du problème de classification présenté à la figure 3.1. Ces illustrations sont réalisées en se basant sur le modèle général *HMD* (avec un seul niveau) et en utilisant des régressions logistiques comme experts. Dans ce cas précis, nous utilisons trois experts ($K = 3$ et $M_i = 1, \forall i \in \{1 \cdots K\}$). Pour chaque point x, nous obtenons un triplet $\left(\pi_1^{(0)}(x), \pi_2^{(0)}(x), \pi_3^{(0)}(x)\right)$. Ce dernier représente les probabilités que x appartienne à l'une des trois régions. Ensuite le triplet est représenté par une couleur de l'espace RVB. Une forte probabilité d'appartenance



(a) Modèle linéaire

(b) Gaussienne

(c) Modèle basé sur la Dirichlet

(d) Modèle basé sur basé sur la Dirichlet généralisée

figure 3.4 – Illustration de différentes fonctions de sélection.

à une région donne soit la couleur rouge, vert ou bleu. Nous obtenons ainsi les limites de chaque région (figure 3.4). Ces limites sont différentes d'une fonction de sélection à l'autre.

3.2 Estimation des Paramètres

À partir du cadre défini dans la section (2.2) nous pouvons déduire l'estimation des paramètres des modèles DGD et HMGD. Nous avons à optimiser au sein du modèle *HMGD* plusieurs modèles DGD pondérés. En général, à la phase *M-step* de l'algorithme EM, nous avons des problèmes d'optimisation sous la forme :

$$\begin{aligned}\Omega^{(l)}_{i,(t+1)} &= argmax_{\Omega^{(l)}_i}\Phi^{(l)} \\ \Phi^{(l)} &= \sum_n H^{(l-1)}_{n,i}\sum_j h^{(l)}_{n,j|i}\ln\left(\pi^{(l)}_{j|i}\left(x_n\right)\right)\end{aligned} \quad , \tag{3.9}$$

où $h^{(l)}_{n,j|i}$ agit comme une étiquette, $H^{(l-1)}_{n,i}$ agit comme un poids. L'exposant (l) fait référence au niveau de la hiérarchie et dans notre cas, $l \in \{0,1,2\}$. Pour les paramètres du premier niveau, d'après les équations 2.16 et 3.9, $H^{(l)}_{n,i} = 1$, $h^{(l)}_{n,j|i} \equiv h^{(0)}_{n,i}$ et $\pi^{(l)}_{j|i} \equiv \pi^{(0)}_i$. Pour les paramètres du deuxième niveau, d'après les équations 2.17 et 3.9, $H^{(l)}_{n,i} \equiv h^{(0)}_{n,i}$, $h^{(l)}_{n,j|i} \equiv h^{(1)}_{n,j|i}$ et $\pi^{(l)}_{j|i} \equiv \pi^{(1)}_{j|i}$. Pour les paramètres des experts, on a d'après les équations 2.18 et 3.9, $H^{(l)}_{n,i} \equiv h^{(0)}_{n,i}h^{(1)}_{n,j|i}$, $h^{(l)}_{n,j|i} \equiv v_{n,c}$ et $\pi^{(l)}_{j|i} \equiv p\left(y = c|x_n;\, \Omega^{(2)}_{ij}\right)$. Les expressions de $h^{(0)}_{n,i}$ et $h^{(1)}_{n,j|i}$ sont définies respectivement dans les équations 2.14 et 2.15. Rappelons que $v_{n,c}$ est une variable binaire indiquant si l'instance x_n appartient ou non à la classe c.

3.2.1 Borne supérieure pour le mélange de Dirichlet généralisée

Étant donné l'expression du modèle DGD, les opérations de maximisation dans le M-Step conduisent à des problèmes d'optimisations hautement non linéaires. Ceci est dû au terme $\ln\left(\frac{\alpha^{(l)}_{ij}GD(x|\mu^{(l)}_{ij})}{\sum_{k=1}\alpha^{(l)}_{ik}GD(x|\mu^{(l)}_{ik})}\right)$ apparaissant dans l'équation (3.9). Il est possible de contourner ce problème en utilisant une approximation variationnelle. Cette dernière permet d'optimiser une borne inférieure de la fonction objective $\Phi^{(l)}$ décrite dans l'équation (3.9). Ceci revient à déterminer une borne inférieure au terme $\ln\left(\frac{\alpha^{(l)}_{ij}GD(x|\mu^{(l)}_{ij})}{\sum_{k=1}\alpha^{(l)}_{ik}GD(x|\mu^{(l)}_{ik})}\right)$. Plus précisément, nous avons à trouver une borne supérieure à l'expression $\ln\left(\sum_k \alpha^{(l)}_{ik}GD(x|\mu^{(l)}_{ik})\right)$. En exploitant l'inverse de l'inégalité de Jensen pour un mélange de distributions appartenant à la famille exponentielle [73], nous déterminons la borne supérieure suivante :

$$\ln\left(\sum_k \alpha^{(l)}_{ik}GD(x_n|\mu^{(l)}_{ik})\right) \leq -\sum_k W^{(l)}_{n,ik}\left[\ddot{\mathbf{x}}^{(l)T}_{n,ik}\bar{\Omega}^{(l)}_{ik} - \mathcal{K}(\bar{\Omega}^{(l)}_{ik})\right] + cst_{n,i} \quad , \tag{3.10}$$

où

$$
\begin{aligned}
cst_{n,i} &= \ln\left(\sum_k \breve{\alpha}_{ik}^{(l)} GD(x_n|\breve{\mu}_{ik}^{(l)})\right) + \sum_k W_{n,ik}^{(l)}\left[\ddot{\mathbf{x}}_{n,ik}^{(l)T}\breve{\Omega}_{ik}^{(l)} - \mathcal{K}(\breve{\Omega}_{ik}^{(l)})\right],\\
\ddot{\mathbf{x}}_{n,ik}^{(l)} &= \frac{\tilde{\pi}_{k|i}^{(l)}(x_n)}{W_{n,ik}^{(l)}}\left[\mathcal{K}'(\breve{\Omega}_{ik}^{(l)}) - \bar{\mathbf{x}}_{n,ik}^{(l)}\right] + \mathcal{K}'(\breve{\Omega}_{ik}^{(l)}),\\
w_{n,ik}^{(l)} &= min\ w_{n,ik}^{(l)}\ tel\ que\ \left[\frac{\tilde{\pi}_{k|i}^{(l)}(x_n)}{w_{n,ik}^{(l)}}\left[\mathcal{K}'(\breve{\Omega}_{ik}^{(l)}) - \bar{\mathbf{x}}_{n,ik}^{(l)}\right] + \mathcal{K}'(\breve{\Omega}_{ik}^{(l)})\right] \in \mathcal{K}'(\bar{\Omega}_{ik}^{(l)}),\\
W_{n,ik}^{(l)} &= 4G\left(\breve{\pi}_{k|i}^{(l)}(x_n)/2\right)\left[(\mathcal{Z}_{n,ik}^{(l)})^T\mathcal{Z}_{n,ik}^{(l)}\right] + w_{n,ik}^{(l)},\\
\mathcal{Z}_{n,ik}^{(l)} &= \mathcal{K}''(\breve{\Omega}_{ik}^{(l)})^{-1/2}\left[\bar{\mathbf{x}}_{n,ik}^{(l)} - \mathcal{K}'(\breve{\Omega}_{ik}^{(l)})\right].
\end{aligned}
\tag{3.11}
$$

Plus de détails sur les expressions apparaissant dans cette borne supérieure sont donnés en annexe A. Le paramètre $\bar{\Omega}_{ik}^{(l)}$ est un re-paramétrage de $\Omega_{ik}^{(l)}$. La valeur $\breve{\pi}_{k|i}^{(l)}(x_n)$ est calculée au point de contact tangentiel $\breve{\Omega}_{ik}^{(l)} = \left\{\breve{\alpha}_{ik}^{(l)}, \breve{\mu}_{ik}^{(l)}\right\}$ (valeur de $\Omega_{ik}^{(l)}$ à l'itération précédente). Le scalaire $cst_{n,i}$ est une constante garantissant un point de contact tangentiel entre la borne supérieure et le logarithme du mélange de DG. Les expressions $\ddot{\mathbf{x}}_{n,ik}^{(l)}$ et $W_{n,ik}^{(l)}$ sont des paramètres variationnels. La fonction $\mathcal{K}(\Omega_{ik}^{(l)})$ est la fonction cumulant de la dirichlet généralisée. Les expressions $\mathcal{K}'(\Omega_{ik}^{(l)})$ et $\mathcal{K}''(\Omega_{ik}^{(l)})$ sont respectivement le gradient et l'Hessien de $\mathcal{K}(\Omega_{ik}^{(l)})$. La fonction G est linéaire et ses paramètres sont définis par une table de référence (voir tableau A.1).

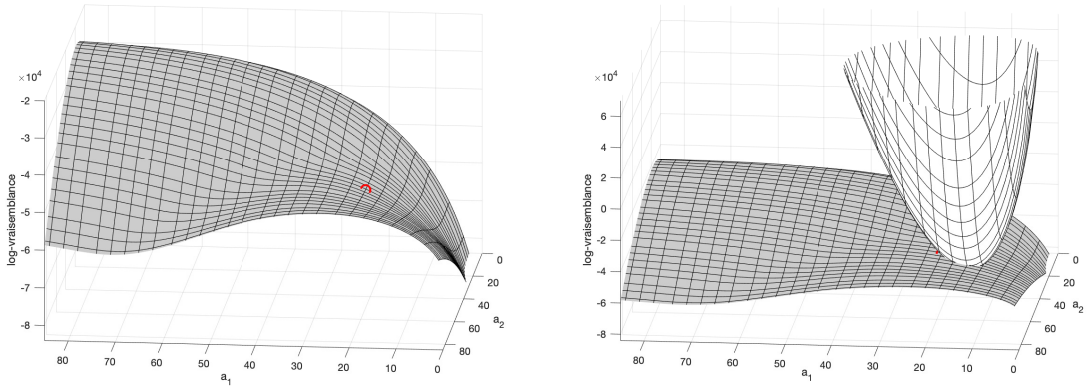
Le tableau 3.1 résume les principaux symboles utilisés pour exprimer la borne supérieure du mélange de DG. Afin de visualiser cette borne supérieure, nous avons tracé la log-vraisemblance du mélange de deux distributions Beta (Fig. 3.5a) de même que sa borne supérieure (Fig. 3.5b). Le mélange utilisé est :

$$0.3\ \texttt{Beta}(x|a_1, 50) + 0.7\ \texttt{Beta}(x|a_2, 100).$$

La log-vraisemblance est tracé en fonction des paramètres a_1 et a_2. La courbe en gris représente la log-vraisemblance tandis que celle plus claire est la borne supérieure générée au point de contact (cercle rouge). Ce point est fixé à $a_1 = 20$ et $a_2 = 50$. Rappelons que la distribution de Beta est un cas particulier de la Dirichlet généralisée.

Symboles	Signification
$\alpha_{ij}^{(l)}$	coefficients
$\mu_{ij}^{(l)} = \left\{a_{ij}^{(l)}, b_{ij}^{(l)}\right\}$	Paramètres de la distribution DG
$\mu_{ij,d}^{(l)} = \left(a_{ij,d}^{(l)}, b_{ij,d}^{(l)}\right)$	Vecteur contenant le $d-$ème élément de $a_{ij}^{(l)}$ et $b_{ij}^{(l)}$
$\Omega_{ij}^{(l)} = \left\{\mu_{ij}^{(l)}, \alpha_{ij}^{(l)}\right\}$	les paramètres de $\pi_{j\|i}^{(l)}$
$\mathcal{K}(.)$	Fonction cumulant de la distribution DG
$W_{n,ij}^{(l)}, \ddot{\mathbf{x}}_{n,ij}^{(l)}$	Paramètres variationnels pour la bonne supérieur du mélange de DG
$\ddot{\mathbf{x}}_{n,ij,d}^{(l)}$	d^{th} élément de $\ddot{\mathbf{x}}_{n,ij}^{(l)}$

tableau 3.1 – Signification des symboles utilisés dans l'expression de la borne supérieure du mélange de DG.



(a) Log-vraisemblance à borner.

(b) Borne supérieure de la log-vraisemblance.

figure 3.5 – Illustration de la borne supérieure du mélange de Beta.

À partir de la borne supérieure du mélange de Dirichlet généralisée, nous pouvons obtenir la borne inférieure pour l'expression $\ln\left(\frac{\alpha_{ij}^{(l)} GD(x|\mu_{ij}^{(l)})}{\sum_{k=1}\alpha_{ik}^{(l)} GD(x|\mu_{ik}^{(l)})}\right)$:

$$\ln\left(\frac{\alpha_{ij}^{(l)} GD(x|\mu_{ij}^{(l)})}{\sum_{k=1}\alpha_{ik}^{(l)} GD(x|\mu_{ik}^{(l)})}\right) \geq \ln\left(\alpha_{ij}^{(l)} GD(x|\mu_{ij}^{(l)})\right) + \sum_{k} W_{n,ik}^{(l)}\left[\ddot{\mathbf{x}}_{n,ik}^{(l)T}\bar{\Omega}_{ik}^{(l)} - \mathcal{K}(\bar{\Omega}_{ik}^{(l)})\right] + cst_{n,i}. \tag{3.12}$$

3.2.2 Estimation des Paramètres des modèles DGD et HMGD

À partir de l'expression de la fonction objective $\Phi^{(l)}$ (Eq. 3.9) et de la borne inférieure décrite dans l'équation 3.12, nous pouvons effectuer les opérations de maximisations sur la borne inférieure suivante :

$$\begin{gathered}\Phi_1^{(l)} = \sum_{n,j} H_{n,i}^{(l-1)}\left[h_{n,j|i}^{(l)} \ln\left(\alpha_{ij}^{(l)} GD(x_n|\mu_{ij}^{(l)})\right) + W_{n,ij}^{(l)}\left[\ddot{\mathbf{x}}_{n,ij}^{(l)t}\bar{\Omega}_{ij}^{(l)} - \mathcal{K}(\bar{\Omega}_{ij}^{(l)})\right] + cst_{n,i}\right] \\ \Phi_1^{(l)} \leq \Phi^{(l)}\end{gathered} \tag{3.13}$$

Étant donné que la Dirichlet généralisée peut être exprimée comme le produit de Beta, les paramètres $\mu_{ij,d}^{(l)}$ peuvent être estimatés indépendamment pour chaque dimension $d \in \{1 \cdots D\}$. Nous utilisons la méthode de Newton-Raphson pour estimer ces paramètres. Vu que $\mu_{ij,d}^{(l)}$ doit être strictement positif, nous réalisons le changement de variable $\mu_{ij,d}^{(l)} = \exp\left(\xi_{ij,d}^{(l)}\right)$. Une fois ce changement de variable effectué, la dérivée partielle de $\Phi_1^{(l)}$ (Eq. 3.13) par rapport à $\xi_{ij,d}^{(l)}$ est donnée par :

$$\begin{aligned}\frac{\partial\, \Phi_1^{(l)}}{\partial \xi_{ij,d}^{(l)}} = \; & \mu_{ij,d}^{(l)}[\Psi(|\mu_{ij,d}^{(l)}|) - \Psi(\mu_{ij,d}^{(l)}) - \ln(A)]\sum_{n} H_{n,i}^{(l-1)} \\ & \left[h_{n,j|i}^{(l)} + W_{n,ij}^{(l)}\right] + \mu_{ij,d}^{(l)}\sum_{n} H_{n,i}^{(l-1)}\left[h_{n,j|i}^{(l)}\tilde{\mathbf{v}}_{n,d} + W_{n,ij}^{(l)}\chi_{n,ij}^{(l)}\right]\end{aligned}, \tag{3.14}$$

où Ψ est la fonction Digamma, $\tilde{\mathbf{v}}_{n,d} = (\ln(\mathbf{v}_{n,d}), \ln(A - \mathbf{v}_{n,d}))^T$ et $\chi_{n,ij}^{(l)} = (\ddot{\mathbf{x}}_{n,ij,2d-1}^{(l)}, \ddot{\mathbf{x}}_{n,ij,2d}^{(l)})^T$. Les deux éléments de $\frac{\partial\, \Phi_1^{(l)}}{\partial \xi_{ij,d}^{(l)}}$ sont notés $\left[\frac{\partial\, \Phi_1^{(l)}}{\partial \xi_{ij,d}^{(l)}}\right]_1$ et $\left[\frac{\partial\, \Phi_1^{(l)}}{\partial \xi_{ij,d}^{(l)}}\right]_2$.

L'Hessien $\mathcal{H}_{ij,d}^{(l)}$ est une matrice symétrique de taille 2×2 où les éléments de la diagonale sont donnés par :

$$\begin{pmatrix} \left[a_{ij,d}^{(l)}\right]^2 \left[\Psi'(a_{ij,d}^{(l)} + b_{ij,d}^{(l)}) - \Psi'(a_{ij,d}^{(l)})\right] \sum_n H_{n,i}^{(l-1)} \left[h_{n,j|i}^{(l)} + W_{n,ij}^{(l)}\right] + \left[\frac{\partial\, \Phi_1^{(l)}}{\partial \xi_{ij,d}^{(l)}}\right]_1 \\ \\ \left[b_{ij,d}^{(l)}\right]^2 \left[\Psi'(a_{ij,d}^{(l)} + b_{ij,d}^{(l)}) - \Psi'(b_{ij,d}^{(l)})\right] \sum_n H_{n,i}^{(l-1)} \left[h_{n,j|i}^{(l)} + W_{n,ij}^{(l)}\right] + \left[\frac{\partial\, \Phi_1^{(l)}}{\partial \xi_{ij,d}^{(l)}}\right]_2 \end{pmatrix} \tag{3.15}$$

et les éléments anti-diagonaux sont donnés par

$$a_{ij,d}^{(l)} b_{ij,d}^{(l)} \Psi'(a_{ij,d}^{(l)} + b_{ij,d}^{(l)}) \sum_{n=1} \left[h_{n,j|i}^{(l)} + W_{n,ij}^{(l)}\right] \ , \tag{3.16}$$

où Ψ' est la fonction Trigamma. Avec une valeur initiale $[\xi_{ij,d}^{(l)}]^{old}$, nous avons l'équation suivante :

$$[\xi_{ij,d}^{(l)}]^{new} = [\xi_{id}^{(l)}]^{old} - [\mathcal{H}_{ij,d}^{(l)}]^{-1} \frac{\partial\, \Phi_1^{(l)}}{\partial \xi_{ij,d}^{(l)}}. \tag{3.17}$$

Nous donnons plus de détails dans l'annexe A.3 pour un calcul moins coûteux de l'inverse de l'Hessien $\mathcal{H}_{ij,d}^{(l)}$. En dérivant $\Phi_1^{(l)}$ (Eq. 3.13) par rapport à $\alpha_{ij}^{(l)}$ et en prenant en compte la contrainte $\sum_j \alpha_{ij}^{(l)} = 1$, nous avons les équations suivantes, pour $j = 1 \cdots M_i - 1$:

$$\alpha_{ij}^{(l)} = \frac{\sum_n H_{n,i}^{(l-1)} h_{n,j|i}^{(l)} + \sum_{n,k} H_{n,i}^{(l-1)} W_{n,ik}^{(l)} \ddot{\mathbf{x}}_{n,ik,2D+j}^{(l)}}{\sum_n H_{n,i}^{(l-1)} + \sum_{n,k} H_{n,i}^{(l-1)} W_{n,ik}^{(l)}} \quad et \quad \alpha_{iM_i}^{(l)} = 1 - \sum_{j=1}^{M_i - 1} \alpha_{ij}^{(l)} \ . \tag{3.18}$$

L'apprentissage du modèle DGD est résumé par l'algorithme 3.1.

Algorithme 3.1 Pseudo-code pour l'apprentissage du modèle *DGD*

Données : $\mathbf{X} = \{\mathtt{x}_1, \cdots, \mathtt{x}_N\} \in \mathbb{R}^{D\times N}$, $\mathbf{Y} = (y_1, \cdots, y_N)^T \in \mathbb{R}^N$, K, M_i et C.
Résultats : Retourne $argmax_{c\in\{1,\cdots,C\}}\, p(y = c|\Omega, x)$ pour une instance x.
1. Initialisation :
Nous utilisons l'algorithme B.1 (voir annexe B.1)
2. E-step : Calculer les paramètres variationnels $W^{(l)}_{n,ij}$ et $\ddot{\mathbf{x}}^{(l)}_{n,ij}$ via l'équation 3.11 et la *log-vraisemblance* via l'équation (3.9). Ici, $H^{(l-1)}_{n,i} = 1$, $h^{(l)}_{n,j|i} \equiv v_{n,c}$ et $\pi^{(l)}_{j|i} \equiv p\,(y = c|x_n;\, \Omega)$. L'expression de $p\,(y = c|x_n;\, \Omega)$ est donné par l'équation 3.7.

3. M-step : Maximiser $\Phi^{(l)}$ (eq. 3.9) étant donné la valeur actuelle de Ω et estimer Ω via les équations 3.14 à 3.18. Ici, $H^{(l-1)}_{n,i} = 1$, $h^{(l)}_{n,j|i} \equiv v_{n,c}$ et $\pi^{(l)}_{j|i} \equiv p\,(y = c|x_n;\, \Omega)$. L'expression de $p\,(y = c|x_n;\, \Omega)$ est donné par l'équation 3.7.

4. Répéter les étapes 2 et 3 jusqu'à convergence.

Étant donné que le modèle *HMGD* est une combinaison de plusieurs modèles *DGD* (pondéré), nous pouvons utiliser le pseudo-code résumé dans l'algorithme 3.2 pour son apprentissage.

Algorithme 3.2 Pseudo-code pour l'apprentissage du modèle HMGD

Données : $\mathbf{X} = \{\mathtt{x}_1, \cdots, \mathtt{x}_N\} \in \mathbb{R}^{D\times N}$, $\mathbf{Y} = (y_1, \cdots, y_N)^T \in \mathbb{R}^N$, K, M_i et C.
Résultats : Retourne $argmax_{c\in\{1,\cdots,C\}}\, p(y = c|\Omega, x)$ pour une instance x.
1. Initialisation :
Nous utilisons l'algorithme B.2 (voir annexe B.2).
2. E-step : Calculer $Q\,(\Omega,\, \Omega^t)$ via les équations 2.13 à 2.15.

3. M-step : Les nouveaux paramètres sont obtenus à partir des anciens pour
(a) $\{\pi^{(0)}_i\}^K_{i=1}$, en appliquant le modèle DGD avec $\{h^{(0)}_{n,i}\}^K_{i=1}$ comme étiquettes,
(b)$\{\pi^{(1)}_{j|i}\}^{M_i}_{j=1}$, en appliquant un modèle DGD pondéré avec $h^{(1)}_{n,j|i}$ comme étiquette et $h^{(0)}_{n,i}$ comme poids,
(c) l'expert $p\left(y = c|x_n;\, \Omega^{(2)}_{ij}\right)$, en appliquant un modèle DGD pondéré avec les étiquettes des données et $h^{(0)}_{n,i}h^{(1)}_{n,j|i}$ comme poids.

4. Répéter les étapes 2 et 3 jusqu'à convergence.

3.3 Résultats Expérimentaux

Afin d'évaluer les performances des modèles *DGD* et *HMGD*, nous avons effectué trois expériences. La première a été réalisée sur plusieurs collections du répertoire UCI [84]. Les autres expériences conduites sont la détection de courriers indésirables et l'identification de l'espace de couleur d'une image. Dans toutes les expériences, le nombre d'itérations maximales de l'algorithme EM pour le modèle *DGD* est fixé à 50 et la tolérance de convergence à 10^{-4}. Pour le modèle *HMGD*, nous avons à effectuer un algorithme EM pour le mélange hiérarchique, ensuite pour chaque modèle *DGD*, un algorithme EM également. Dans ce cas, afin de réduire le surapprentissage, nous avons estimé les paramètres de $\left\{\pi_i^{(0)}(x)\right\}_{i=1}^{K}$ et de $\left\{\pi_{j|i}^{(1)}(x)\right\}_{ij=11}^{KM_i}$ avec cinq itérations au maximum. De plus, le nombre d'itérations en ce qui concerne les experts est fixé à 30 au maximum. Par ailleurs, nous fixons le nombre d'itérations maximales de l'algorithme EM pour le mélange hiérarchique à 10. En général, nous avons observé qu'un nombre maximal d'itérations supérieur à ceux indiqués plus haut conduit à du surapprentissage. Puisque que la Dirichlet généralisée ne prends pas en compte des valeurs nulles, nous remplaçons ces dernières par une valeur faible [88].

3.3.1 Évaluation des modèles DGD et HMGD sur des collections du répertoire UCI

Dans cette section, nous avons mené notre étude sur six collections de référence issues du répertoire UCI [84]. Le but de cette étude est de comparer les modèles DGD, HMGD, HMD1, HMD2, DME, la régression logistique (LR) de même que le mélange de Dirichlet généralisée (MDG). Pour ce dernier, chaque classe est d'abord décrite par une distribution de Dirichlet généralisée. Ensuite, les probabilités *a posteriori* sont utilisées pour donner l'appartenance à une classe. Estimer les paramètres du mélange de Dirichlet généralisée revient à estimer les paramètres du modèle DGD en fixant les paramètres variationnels à zéro. La comparaison entre le modèle MDG et DGD nous permettra d'évaluer la pertinence de la borne supérieure décrite plus haut. Rappelons que HMD1, HMD2 et DME sont des modèles qui combinent plusieurs régressions logistiques à l'aide de différents types de fonctions de sélection. Afin d'obtenir des données proportionnelles, nous avons : 1) supprimé les caractéristiques binaires, 2)

standardisé et ramené dans l'intervalle $]0, 1[$, 3) normalisé les données afin qu'elles soient dans le simplex. Le tableau (3.2) présente les principales caractéristiques des collections utilisées.

Collections	#instances	dimension	#classes
Appendicitis	106	7	2
Vowel	990	10	11
Cancer (Breast)	277	9	2
Magic	19,020	10	2
Vehicle	946	18	4
Satimage	6 435	4	6

tableau 3.2 – Collections utilisées pour l'évaluation des modèles *DGD* et *HMGD*.

Les modèles sont évalués en fonction de la justesse (eq. 2.25) sur la base d'une validation croisée à cinq plis. Nous avons réalisés deux expériences. Dans la première, le nombre d'experts utilisé est celui sélectionné par le modèle HMD1 tandis que dans la deuxième, nous faisons varier le nombre d'experts entre deux et huit ($M_i = 2$). Dans la deuxième expérience, les résultats sont également évalués en termes de temps d'apprentissage.

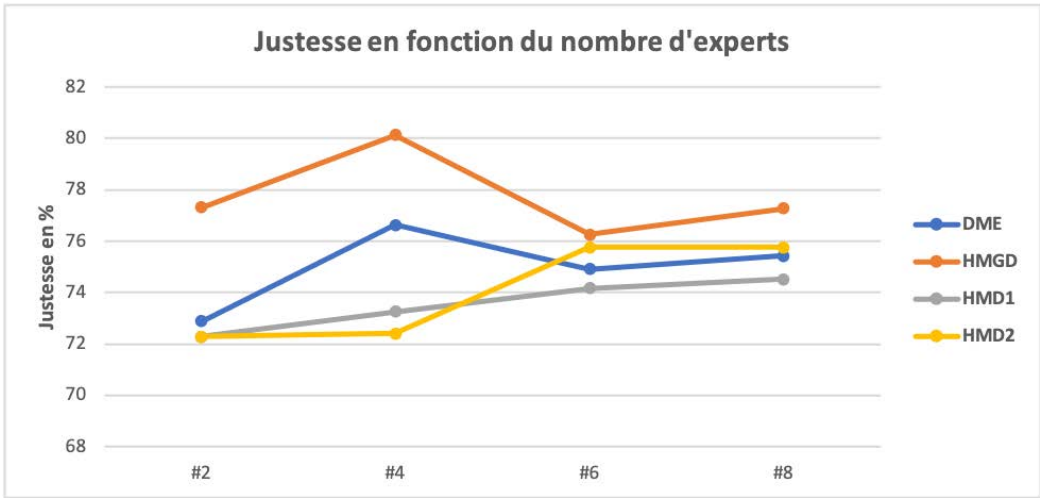
Les résultats de la première expérience sont reportés dans le tableau (3.3) et le nombre entre parenthèses désigne le nombre d'experts utilisé au sein des modèles HMGD, HMD1, HMD2, DME. Pour cette expérience, les résultats varient d'une collection à une autre. Sur la collection *Magic*, DGD et HMGD améliorent de plus de 4%, la justesse obtenue par la régression logistique et MDG. Cependant les écart-types obtenus par DGD et HMGD sont un peu plus élevés que celui de la régression logistique. Sur cette même collection, HMD1 et DME obtiennent des justesses légèrement supérieures à celle de HMGD (moins de 1%) et un meilleur écart-type. Sur la collection *Vehicle*, DGD obtient une justesse supérieure d'au moins 7% comparé à MDG, HMD1, HMD2 et DME. Par contre ces derniers ont un écart-type légèrement faible (environ 1% de moins que DGD). Sur toutes les collections, le modèle DGD a obtenu une justesse supérieure à celle de MDG. Dans le seul cas (*Vehicle*) où l'écart-type obtenue par DGD est supérieure (moins de 2%) à celle de MDG, la justesse de DGD est de 9% supérieure à celle de MDG. Ceci montre l'avantage lié à l'utilisation de la borne supérieure du mélange de DG. Le modèle DGD obtient une meilleure

Collections / Modèles	**Magic** (5)	**Veh.**(4)	**Vow.** (4)	**Appe.** (2)	**Sat.** (4)	**Cancer** (2)	**Moyenne**
LR	78.5 ± 0.39	40.8 ± 12.44	64.04 ± 3.34	84.89 ± 4.03	74.33 ± 0.58	72.56 ± 2.76	69.25
MDG	77.25 ± 1.43	52.96 ± 5.6	66.36 ± 4.47	83.07 ± 5.11	77.53 ± 0.69	71.49 ± 4.6	71.44
HMD1	83.65 ± 0.74	44.09 ± 6.56	86.16 ± 4.53	86.80 ± 2.08	75.34 ± 0.62	72.57 ± 6.93	74.76
HMD2	83.31 ± 1.17	45.99 ± 7.22	82.02 ± 3.91	86.80 ± 2.08	75.34 ± 0.62	72.57 ± 6.93	74.33
DME [127]	**83.84** ± 0.53	55.44 ± 2.82	84.04 ± 3.81	84.89 ± 2.22	75.31 ± 0.74	70.4 ± 7.09	75.61
DGD	82.23 ± 0.8	62.17 ± 7.12	79.49 ± 3.08	86.8 ± 3.96	78.15 ± 0.61	72.22 ± 4.6	76.52
HMGD	83.22 ± 0.83	**68.91** ± 8.37	**88.79** ± 0.9	**87.75** ± 2.5	**78.91** ± 0.52	**72.22** ± 3.09	**79.97**

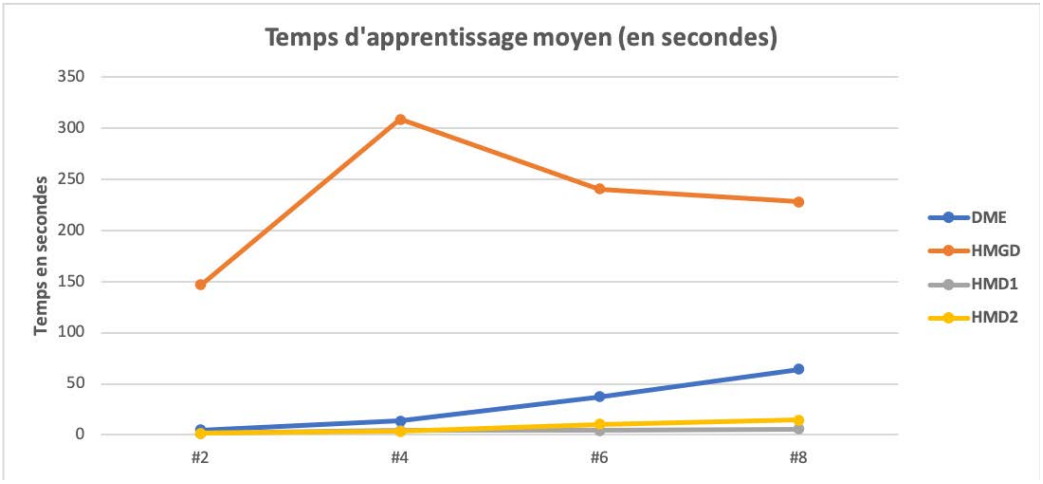
tableau 3.3 – Comparaison de DGD, MDG, HMGD, DME, HMD1, HMD2 et LR.

justesse comparé aux modèles HMD1 et HMD2 sur les collections *Vehicle* et *Satellite*. Sur ces deux collections, l'écart-type de ces trois modèles (DGD, HMD1 et HMD2) sont similaires. En ce qui concerne HMGD, sur les collections *Vowel*, *Appendicitis* et *Satimage*, il réalise de meilleures justesses comparé à HMD1 et HMD2 (entre 1% et 6% de plus). Sur ces trois collections, HMGD obtient des écart-types inférieures ou similaires à celles de HMD1 et HMD2. Sur la collection *Cancer*, comparé à HMD1 et HMD2, HMGD obtient un meilleur écart-type (3% de moins) tandis que les justesses de ses trois modèles sont similaires (moins de 0.4% de différence). En général, le modèle HMGD améliore la justesse et l'écart-type obtenus par DGD. Dans les cas où l'écart-type obtenu par HMGD est supérieure à celle de DGD (*Magic* et *Vehicle*), la différence est faible comparée au gain en termes de justesse.

Les résultats de la deuxième expérience sont illustrés par les figures 3.6a et 3.6b. En moyenne, en utilisant deux experts, HMGD obtient une justesse supérieure à celle obtenue par DME, HMD1 et HMD2. À l'exception des collections *Magic* et *Vehicle* (pour huit experts), HMGD obtient une meilleure justesse comparée à HMD1. Le modèle HMD2 obtient une meilleure justesse comparée à HMGD uniquement sur les



(a)



(b)

figure 3.6 – (a) Justesse en fonction du nombre d'experts. (b) Temps d'apprentissage en fonction du nombre d'experts.

collections *Vehicle* (pour six et huit experts) et *Magic* (pour huit experts). En ce qui concerne le temps d'apprentissage, HMGD est plus lent que HMD1, HMD2 et DME. Ceci est dû au calcul des paramètres variationnels et à l'algorithme itérative utilisé pour l'estimation des paramètres des distributions de Beta [2]. Rappelons aussi

2. Grâce à l'équation 3.2, estimer les paramètres de la Dirichlet généralisée revient à estimer les paramètres de D distribution de Beta

qu'au sein de l'algorithme du modèle HMGD, nous avons à optimiser plusieurs fois le modèle DGD. En général, le temps d'apprentissage au sein du mélange hiérarchique est dominé par l'estimation des paramètres des experts. Dans le cas de HMD1 et HMD2, les experts étant des modèles linéaires, l'apprentissage est plus rapide comparé à HMGD. La lenteur du modèle DME comparé aux modèles HMD1 et HMD2 est dûe au fait que l'optimisation des paramètres de la fonction de sélection nécessite un algorithme itérative. Rappelons que la fonction de sélection utilisé au niveau du modèle DME est basé sur la distribution de Dirichlet.

3.3.2 Détection des courriers indésirables

Les spams ou courriers indésirables sont des messages envoyés à un ensemble de destinataires et qui sont sans intérêt pour eux. Ces courriers peuvent contenir des logiciels malveillants ou des programmes d'hameçonnage. Selon Kaspersky, 56.51 % des emails reçus en 2019 étaient des spams [87]. Il est important de pouvoir détecter ces courriers afin de ne pas ennuyer les destinataires, leur permettre de se concentrer sur l'essentiel ou de leur éviter d'être victimes d'actes criminels. À cet effet, plusieurs méthodes ont été utilisées notamment, celles liées à l'apprentissage automatique [71, 104, 8, 44]. Dans cette section, nous utilisons nos modèles DGD et HMGD pour aborder le problème de la détection de courriers indésirables. Nos modèles sont comparés aux modèles HMD1 et LR (régression logistique) de même qu'au mélange de Dirichlet généralisée (MDG). L'objectif ici est de comparer la stratégie consistant à appliquer l'alpha-transformation (eq. 3.3) aux données et nos modèles qui utilisent des données définies dans un simplexe. Ainsi, nous effectuons l'alpha-transformation sur les données avant d'utiliser les modèles HMD1 et LR pour la classification. Dans nos expériences, avec l'alpha-transformation, nous avons fait varier α entre 0 et 1 avec un pas de 0.2. Les modèles sont évalués à l'aide de la justesse (eq. 2.25). Les résultats reportés dans le tableau (3.4) sont basés sur une validation croisée à cinq plis. Le chiffre entre parenthèses désigne le nombre d'experts utilisés. Nos expériences ont été réalisées sur deux collections précédemment utilisées dans la littérature pour la détection de spams [78, 8, 113]. Les modèles HMD1 et HMGD ont été entrainés en utilisant deux experts car l'augmentation du nombre d'Experts n'améliorent pas les

résultats.

i) **HP Spambase** est issue du répertoire UCI [84] et a été créée par *Hewlett-Packard Labs*. Cette collection contient 4601 instances ayant 57 attributs continus et un attribut binaire qui représente les étiquettes. Cette dernière indique si un courrier est indésirable ou légitime. Les courriers indésirables représentent 39.4% des instances de la collection. Étant donné que nos modèles sont définis uniquement pour des données proportionnelles, nous utilisons les 20 attributs représentant la fréquence des mots les plus utilisés dans la collection. Un détecteur de courriers indésirables peut répartir les courriers en trois catégories : légitime, indésirable et nécessitant un examen supplémentaire. Afin de créer cette troisième catégorie, nous avons choisi aléatoirement 977 courriers parmi les courriers légitimes. Ainsi, la nouvelle collection a une même proportion de courriers indésirables et de courriers légitimes et 21.2% de courriers nécessitant un examen supplémentaire. Les données sont par la suite normalisées afin de respecter les contraintes sur les données proportionnelles. Les résultats obtenus par la régression logistique et HMD1 sont similaires à celui du modèle DGD. Cependant, le modèle HMGD améliore les résultats obtenus par le modèle DGD et surpasse le modèle HMD1. Quoique l'écart-tye obtenu par HMGD soit légèrement supérieur à celui de HMD1 (environ 0.06% de plus), le gain en justesse est à près de 1%. Rappelons que le modèle HMD1 utilisé est un mélange de deux régressions et que le modèle HMGD utilisé est un mélange de deux modèles DGD. Contrairement au HMD1 qui n'améliore pas la justesse obtenue par la régression logistique, HMGD obtient une meilleure justesse comparée à DGD.

ii) **Ling-spam** [3] est une collection de courriers dont 481 (16.6%) sont indésirables et 2412 (82.4%) légitimes. Nous avons effectué un traitement similaire à celui appliqué à la collection *HP Spambase* . Nous avons utilisé 1443 courriels afin d'obtenir des données équilibrées (soit 481 courriers par catégorie). La catégorie des courriers nécessitant un examen supplémentaire est créée en sélectionnant aléatroirement 481 courriers légitimes. Afin de construire les caractéristiques à utiliser, nous supprimons d'abord les mots non significatifs ou mots vides et nous appliquons une lemmatisation [4] [8]. Nous construisons par la suite, un dictionnaire contenant les 30 mots les

3. https://www.kaggle.com/mandygu/lingspam-dataset
4. Une lemmatisation sert à retrouver le lemme d'un mot (forme canonique du mot).

plus courants. Les caractéristiques sont obtenues en créant un vecteur contenant la fréquence d'apparition des mots du dictionnaire. Les modèles DGD et HMGD ont obtenu de meilleures justesses comparées aux modèles LR, HMD1 et MDG. Nous observons que le modèle HMGD améliore la justesse obtenue par DGD d'un peu plus de 1% de même que l'écart-type. Les modèles DGD et HMGD obtiennent une justesse de 2% à 3% supérieure aux deux modèles utilisant l'α-transformation (LR et HMD1). Comme noté dans les expériences de la section précédente, l'approximation variationnelle utilisée au sein du modèle DGD nous a permis d'obtenir de meilleurs résultats comparés au modèle MDG.

	LR	MDG	HMD1 (2)	DGD	HMGD (2)
HP Spambase	73.51 ± 0.9 ($\alpha^* = 0$)	71.30 ± 0.83	73.49 ± 0.66 ($\alpha^* = 0$)	73.60 ± 0.85	**74.47** ± 0.72
Ling-spam	62.81 ± 2.69 ($\alpha^* = 0.6$)	63.93 ± 2.92	62.67 ± 2.56 ($\alpha^* = 0.6$)	64.81 ± 3.01	**66.07** ± 2.10

tableau 3.4 – Comparaison de DGD, HMGD, HMD1, LR et MDG sur les collections *HP Spambase* et *Ling-spam*.

3.3.3 Identification de l'espace de couleur d'une image

L'espace de couleur est une représentation d'un système de synthèse de la couleur et se base en général sur un ensemble de triplets. Une couleur est donc associée à chaque triplet et peut varier d'un espace à un autre. L'un des espaces de couleur les plus utilisés pour la visualisation est le RVB et est composé des canaux Rouge, Vert et Bleu. Dans le souci d'une représentation plus fidèle des couleurs nécessaire à différentes activités humaines (ex : photographie, affichage, industrie), plusieurs variantes de l'espace RVB ont vu le jour. Nous pouvons citer Adobe RVB, Apple RVB, sRVB et ProPhoto RVB. Il est important de correctement choisir l'espace de couleur d'une image afin de garantir une reproduction fidèle de cette dernière. Certaines applications choisissent par défaut l'espace de couleur pour l'affichage des images.

Par exemple, l'espace de couleur utilisé par défaut sur le Web est le sRVB tandis que l'éditeur d'images Adobe Lightroom utilise Adobe RVB. Après avoir traité une image, il est possible d'intégrer à cette dernière, la donnée relative à son espace de couleur. Une fois cette donnée connue, l'image peut être fidèlement représentée avec n'importe quel autre dispositif ayant ou pas le même espace de couleur que l'image. Mais pour diverses raisons, cette donnée peut être absente ou peu fiable ce qui entraînera une mauvaise représentation de l'image. Ce fait est illustré par la figure 3.7 où les trois premières lignes sont des images venant de cinq différents espaces de couleur. Néanmoins, la donnée relative à l'espace de couleur n'est pas disponible dans les métadonnées de ces images. Sans cette information, l'affichage de ces images varie d'un espace à l'autre. La quatrième ligne de la figure 3.7 représente des images qui incorporent la donnée relative à leur espace de couleur. Nous remarquons dans ce cas que l'affichage varie très peu de l'original.

figure 3.7 – Images affichées avec le fureteur Firefox.

Le but de notre expérience est d'identifier les espaces de couleur de la famille

RVB à partir d'une image. Pour cette expérience, nous avons considéré les espaces Apple RVB, ColorMatch RVB, Adobe RVB, sRVB et ProPhoto RVB. Le problème de l'identification des espaces de couleur a été abordé pour la première fois par Vezina *et al.* [133]. Dans ce travail, les auteurs se sont limités aux espaces sRVB, TSV, TSL et Lab et se sont basés sur l'estimation du Gamut. Malheureusement, pour l'identification des espaces de couleur entre membres de la famille RVB (ex : Adobe RVB, Apple RVB et sRVB), il faut une estimation plus précise de la forme et du volume du Gamut. Cependant, il est possible d'exploiter de potentiels correlations entre les pixels d'une image. On entend ici par corrélation, le fait qu'un pixel est formé à partir des pixels de son voisinage. Supposons que ces corrélations sont linéaires. Soit I^{rvb} une image dans un espace RVB, la corrélation entre pixels peut être décrite à l'aide du modèle suivant [105] :

$$I^{rvb}_{k_1}(m,n) = \sum_{(i,j)\in V(m,n)} \gamma_{i,j,k_1} I^{rvb}_{k_2}(m+i, n+j) + r(m,n) \quad , \tag{3.19}$$

où $(k_1, K_2) \in \{R,V,B\} \times \{R,V,B\}$ fait référence aux canaux de l'espace RGB et $V(m,n)$ est le voisinage du pixel situé à l'emplacement (m,n). Lorsque $k_1 = K_2$, $\gamma_{0,0,k_1} = 0$. L'expression $r(m,n)$ représente un bruit blanc gaussien ($r(m,n) \sim \mathcal{N}(0,\sigma^2)$) et γ_{i,j,k_1} sont les coefficients de corrélation. Nous avons deux corrélations possibles à savoir la corrélation intra-canal ($k_1 = k_2$) et la corrélation inter-canal ($k_1 \neq k_2$). Les paramètres du modèle décrit à l'équation 3.19 sont donnés par le vecteur γ_{k_1} ayant pour éléments $= \{\gamma_{i,j,k_1} \mid (i,j) \in V(m,n)\}$ et l'écart-type σ. La figure 3.8 illustre quelques exemples de voisinages.

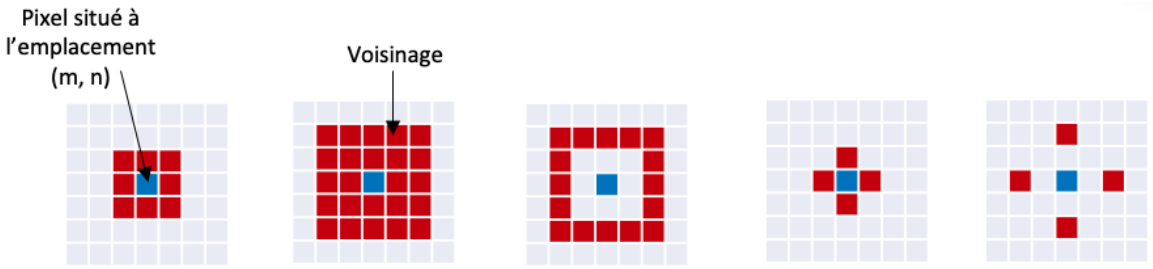


figure 3.8 – Exemple de voisinages.

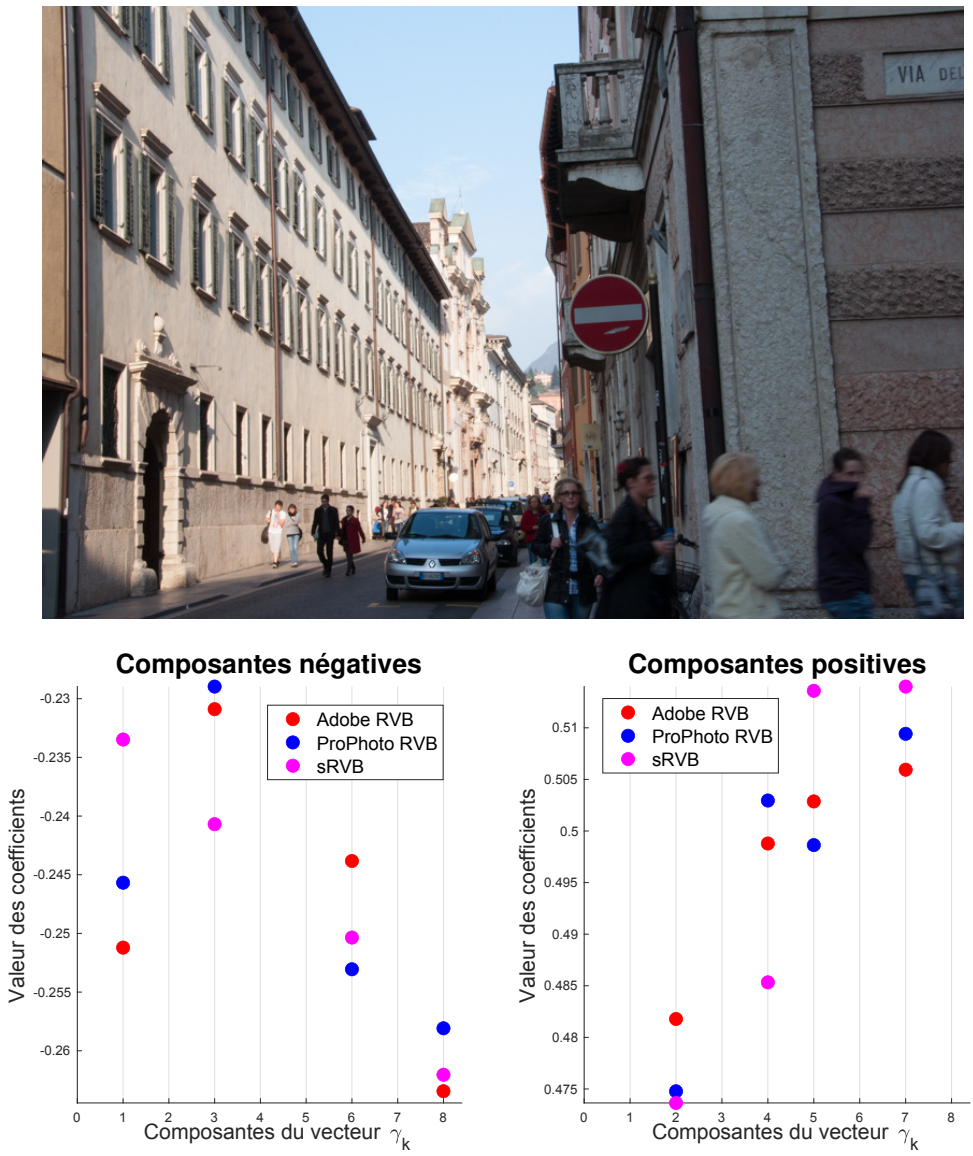


figure 3.9 – Coefficients de corrélation pour chaque espace de couleur.

Soit l'image de la figure 3.9 représentée dans les espaces sRVB, Adobe RVB et ProPhoto RVB. Pour chacune des trois images obtenues, les composantes du vecteur γ_{k_1} sont estimées à l'aide de l'algorithme décrit en annexe C.1 ($k_1 = k_2$). Le voisinage utilisé est celui de la première illustration de la figure 3.8 en partant de la gauche. La figure 3.9 illustre les coefficients estimés à partir du canal Rouge de chacune de ces images. Les axes x et y représentent respectivement l'indice et la valeur de chaque

élément du vecteur γ_{k_1}. Pour une meilleure visualisation, nous avons représenté les composentes négatives et positives du vecteur γ_k sur deux graphiques distincts. Nous remarquons que ces coefficients varient d'un espace de couleur à un autre. À partir de cette observation, nous supposons que ces coefficients peuvent être utilisés pour l'identification de l'espace couleur.

À partir d'une collection d'images étiquetées, nous pouvons estimer les coefficients de corrélation pour chaque image. Les données obtenues peuvent être utlisées pour entraîner un classifieur. Pour le test, étant donnée une image, les coefficients γ_{i,j,k_1} sont estimés (alg. C.1) et l'espace de couleur est identifié en utilisant le classifieur entraîné.

Nous avons collecté 1000 images de chacune des collections DRESDEN [59] et RAISE [45]. Ensuite, Adobe Lightroom a été utilisé afin de générer les images dans chacun des cinq espaces de couleur. Ceci donne un total de 10K images. Nous nous limitons au deuxième exemple de voisinage (deuxième illustration de la figure 3.8 en partant de la gauche). Dans ce cas, les éléments du vecteur γ_{k_1} sont $\{\gamma_{i,j,k_1} \mid -2 \leq i,j \leq 2\}$. L'analyse discriminante généralisée [11] a été utilisé pour réduire la dimension des données. Nos modèles (DGD et HMGD) sont comparés aux modèles HMD1 et LR (régression logistique) de même qu'au mélange de Dirichlet généralisée (MDG). Les modèles LR et HMD1 sont utilisés en combinaison avec l'alpha-transformation (Eq. 3.4). Les performances ont été évaluées à l'aide de la justesse et sont en pourcentage(%). Les modèles HMD1 et HMGD ont été entrainés en utilisant deux experts car l'augmentation du nombre d'experts n'améliorent pas les résultats.

Tous les résultats reportés dans les tableaux 3.5 et 3.6 sont sur la base d'une validation croisée à cinq plis. En utilisant les coefficients de corrélation intra-canal, les modèles HMGD et DGD obtiennent de meilleures justesses comparés aux autres. Cependant, la justesse obtenue par le modèle DGD est légèrement supérieure à celle obtenue par HMD1 (moins d'1 %). Le modèle HMGD a amélioré la justesse obtenue par le modèle DGD tout en réduisant l'écart-type. Nous notons que les modèles HMD1 et HMGD ont ontenus un écart-type inférieur à celui obtenu par les autres modèles. En général, HMD1 est meilleur pour l'identification des espaces ColorMatch RGB et Prophoto RGB. Les modèles MGD et HMGD sont respectivement meilleurs pour l'identification des espaces Adobe RGB et Apple RGB. Il est bien de noter qu'en

	α^*	**Espace de couleur**	**Justesse par espace (%)**	**Justesse**
LR	0	Adobe	63.25	66.88 ± 1.17
		Apple	63.85	
		ColorMatch	78.5	
		ProPhoto	67.2	
		sRVB	61.6	
HMD1	1	Adobe	62.25	67.71 ± 0.91
		Apple	62.59	
		ColorMatch	**82.4**	
		ProPhoto	**70.3**	
		sRVB	61	
MDG		Adobe	**82.9**	66.24 ± 1.40
		Apple	61.54	
		ColorMatch	63.65	
		ProPhoto	62.7	
		sRVB	60.4	
DGD		Adobe	72	68.05 ± 1.19
		Apple	60.14	
		ColorMatch	78.05	
		ProPhoto	66.55	
		sRVB	63.5	
HMGD		Adobe	70.55	**69.58** ± 0.75
		Apple	**67.75**	
		ColorMatch	78.05	
		ProPhoto	68	
		sRVB	63	

tableau 3.5 – Comparaison des résultats obtenus par les modèles DGD, HMGD, HMD1, LR et MDG en utilisant les coefficients de corrélation intra-canal.

utilisant la méthode proposée par Vezina *et al.* [133], la justesse obtenue est de 20.61 %. Ceci confirme le fait que pour l'utilisation du gamut pour l'identification des espaces RVB, il faut une estimation plus précise du gamut.

En utilisant les coefficients de corrélation inter-canal (tab. 3.6), tous les résultats ont été amélioré de près de 10%. Le modèle HMGD améliore les justesses obtenues par les autres modèles. Cependant la différence en termes de justesse est faible lorsque

	α^*	**Espace de couleur**	**Justesse par espace (%)**	**Justesse**
LR	0.8	Adobe	71.05	77.65 ± 1.06
		Apple	76.8	
		ColorMatch	67.25	
		ProPhoto	83.55	
		sRVB	89.6	
HMD1	1	Adobe	67.55	78.56 ± 0.90
		Apple	**78.5**	
		ColorMatch	**91.65**	
		ProPhoto	**85.3**	
		sRVB	69.95	
MDG		Adobe	70.6	77 ± 1.17
		Apple	71.95	
		ColorMatch	66.55	
		ProPhoto	83.45	
		sRVB	**92.45**	
DGD		Adobe	72.65	78.28 ± 0.82
		Apple	72.75	
		ColorMatch	81.2	
		ProPhoto	83.8	
		sRVB	81	
HMGD		Adobe	**75**	**79.18** ± 0.89
		Apple	77	
		ColorMatch	76.7	
		ProPhoto	85.15	
		sRVB	82.05	

tableau 3.6 – Comparaison des résultats obtenus par les modèles DGD, HMGD, HMD1, LR et MDG en utilisant les coefficients de corrélation inter-canal.

HMGD est comparé à HMD1 et DGD (moins d'1%).

3.4 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la classification des données proportionnelles. À cet effet, nous avons proposé l'utilisation de la probabilité *a posteriori*

basée sur la Dirichlet généralisée comme modèle (DGD). De plus, à partir du modèle HMD présenté dans le chapitre 2, nous avons proposé un mélange hiérarchique du classifieur DGD (HMGD). À travers une approximation variationnelle, nous avons estimé les paramètres du modèle DGD. Afin d'effectuer cette approximation, nous avons proposé une borne supérieure pour mélange de Dirichlet généralisée. À notre connaissance, c'est la première qu'une telle borne est établie pour le mélange de Dirichlet généralisée. Les expériences menées sur des collections de référence ont montré que les modèles DGD et HMGD peuvent surpasser les modèles HMD1 et HMD2 (voir chapitre 2). Les expériences concernant la détection de courriers indésirables, nous suggère que les modèles DGD et HMGD peuvent être des alternatifs à l'utilisation de l'alpha-transformation. Pour finir, nous avons noté que dans certains cas, le modèle HMGD n'obtient pas de meilleures prédictions comparées au modèle DGD. Ceci est dû soit à du surapprentissage, soit au fait que les données sont presque linéairement séparables. Dans le premier cas, on pourrait ajouter des contraintes au modèle HMGD lors de l'estimation des paramètres.

Conclusion

Dans cette thèse, nous nous sommes intéressés à la classification de données non linéairement séparables. Notre motivation première est basée sur le fait que plus les données deviennent complexes et plus les algorithmes d'apprentissage tendent à être des boîtes noires. Ainsi en se basant sur un principe naturel qui consiste à décomposer un problème trop complexe en une série de problèmes plus faciles à aborder, nous avons proposé plusieurs modèles discriminatifs pour la classification non linéaire des données.

Dans le premier chapitre, nous avons effectué une revue de littérature des principales méthodes utilisées pour la classification non linéaire des données. Dans le deuxième chapitre, nous avons proposé un modèle général du mélange hiérarchique de classifieurs discriminatifs (*HMD*). L'idée principale derrière notre modèle est de diviser une tâche complexe de classification en plusieurs sous-tâches à travers un ensemble de fonctions de sélection. Chaque sous-tâche est traitée par un classifieur discriminatif (expert). Les fonctions de sélection servent à la fois à répartir les tâches entre les experts, mais aussi à leur attribuer un degré d'expertise. À travers ce modèle général, nous avons montré que la plupart des modèles de mélange discriminatifs peuvent en être déduits. De plus, nous avons extrait deux nouveaux exemples (*HMD1* et *HMD2*) à partir de ce modèle général. À travers ces deux exemples, nous avons démontré qu'il est possible d'utiliser moins d'experts tout en gardant des scores de classification élevés. L'exemple *HMD1* modélise les fonctions de sélection du premier et du second niveau, respectivement avec un modèle non linéaire et un modèle linéaire. Quant à l'exemple *HMD2*, les fonctions de sélection sont modélisées par un modèle non linéaire à travers toute la hiérarchie. En agissant de la sorte, ces deux exemples permettent une répartition efficace des tâches entre les experts et conduit de

ce fait à une utilisation plus judicieuse du nombre d'experts. Des expériences ont été réalisées afin de montrer l'habileté de nos exemples à réduire le nombre d'experts. En outre, *HMD1* et *HMD2* surpassent quelques méthodes pertinentes de l'état de l'art. Cependant, *HMD1* est en général plus efficace en termes de justesse que *HMD2*. Par contre, le temps d'apprentissage de ce dernier est plus court en présence d'un petit nombre d'experts. Ce travail a fait l'objet d'une publication dans un articule intitulé **Classification using hierarchical mixture of discriminative learners : How to achieve high scores with few resources ?**, dans la revue **Expert Systems with Applications** en 2018.

Dans le troisième chapitre, nous nous sommes intéressés à la classification des données proportionnelles par une approche discriminative. Pour ce fait, le modèle proposé (*DGD*) est basé sur la Dirichlet généralisée. Afin d'estimer les paramètres de ce modèle, nous avons établi une borne supérieure au mélange de Dirichlet généralisée. À notre connaissance, c'est la première fois qu'une telle borne est proposée. À partir du modèle HMD, nous avons aussi proposé un modèle de mélange hiérarchique où les fonctions de sélection et les experts sont modélisés avec le modèle *DGD*. L'idée est de réduire l'erreur de classification du modèle *DGD*. Les expériences ont montré que les modèles *DGD* et *HMGD* sont en moyenne, meilleurs que les modèles *HMD1* et *HMD2* en ce qui concerne la classification de données proportionnelles. De plus, le modèle HMGD peut utiliser peu d'experts comparativement au modèle HMD1 et HMD2. Cependant, nous avons noté que dans le cas où le modèle *DGD* est sujet à du surapprentissage, sa combinaison via un mélange hiérarchique ne garantit pas l'amélioration des scores. Il serait donc utile d'établir des stratégies afin de permettre une meilleure généralisation. Le temps d'apprentissage du modèle HMGD est également élevé.

Les travaux effectués durant cette thèse mènent à des extensions intéressantes. En effet, le modèle *HMD* pourrait être utilisé pour classifier des données provenant de plusieurs sources. Dans ce cas, chaque expert reçoit les données provenant d'un certain nombre de sources. Les fonctions de sélection indiqueraient dans ce cas, la contribution de chaque source à la décision finale. Par ailleurs, l'établissement de la borne supérieure du mélange de Dirichlet généralisée pourrait faciliter l'analyse bayésienne des modèles *DGD* et *HMGD* ce qui pourrait réduire le surapprentissage.

Il serait également intéressant de se pencher sur la classification d'autres types de données (ex. spatio-temporelles) à l'aide d'exemples spécifiques issus du modèle *HMD*.

Annexe A

Borne supérieure pour le mélange de Dirichlet généralisée

La Dirichlet généralisée appartient à la famille des distributions exponentielles. Une distribution de la famille exponentielle peut s'écrire sous la forme :

$$\exp\left(\mathcal{A}(X) + X^T\Omega - \mathcal{K}(\Omega)\right),$$

où $\mathcal{K}(\Omega)$ est la fonction cumulant qui dépend uniquement des paramètres et est convexe. La fonction $\mathcal{A}(X)$ dépend uniquement des données. Une des propriétés de la famille exponentielle est que le vecteur X appartient à l'espace du gradient de la fonction cumulant $\mathcal{K}(\Omega)$ ($X \in \frac{\partial}{\partial\Omega}\mathcal{K}(\Omega)$ ou $X \in \mathcal{K}'(\Omega)$).

Dans les lignes suivantes, nous appliquons les résultats obtenus dans [73], au cas spécifique du mélange de Dirichlet généralisée. En effet, Jebara *et al.* [73] ont proposé une bonne supérieure pour le mélange de distributions appartenant à la famille exponentielle. Ils ont exploité entre autre la convexité de la fonction cumulant $\mathcal{K}(\Omega)$. Pour plus de détails sur les étapes concernant l'élaboration de cette borne, les lecteurs peuvent se référer à la thèse de Jebara *et al.* [73].

A.1 Re-paramétrage du mélange de Dirichlet généralisée

Nous utilisons la transformation de la distribution de Dirichlet généralisée décrite dans l'équation 3.2. Le mélange de Dirichlet généralisée peut être re-paramétré sous forme d'une somme de distributions de la famille exponentielle :

$$\sum_{k=1}^{M_i} \alpha_{ik}^{(l)} GD(x_n|\mu_{ik}^{(l)}) = \sum_{k=1}^{M_i} \exp\left[\bar{\mathbf{x}}_{n,ik}^{(l)T} \bar{\Omega}_{ik}^{(l)} - \mathcal{K}(\bar{\Omega}_{ik}^{(l)})\right], \tag{A.1}$$

où

$$\bar{\mathbf{x}}_{n,ik}^{(l)} = \begin{pmatrix} \ln(\mathtt{v}_{n,1}) \\ \ln(A - \mathtt{v}_{n,1}) \\ \vdots \\ \ln(\mathtt{v}_{n,D}) \\ \ln(A - \mathtt{v}_{n,D}) \\ \dot{\mathbf{x}}_k \end{pmatrix}; \quad \bar{\Omega}_{ik}^{(l)} = \begin{pmatrix} \mu_{ik}^{(l)} \\ \\ \eta_i^{(l)} \end{pmatrix}; \quad \mu_{ik}^{(l)} = \begin{pmatrix} a_{ik,1}^{(l)} \\ b_{ik,1}^{(l)} \\ \vdots \\ a_{ik,D}^{(l)} \\ b_{ik,D}^{(l)} \end{pmatrix}; \quad \eta_i^{(l)} = \begin{pmatrix} \eta_{i,1}^{(l)} \\ \vdots \\ \eta_{i,M_i-1}^{(l)} \end{pmatrix},$$

$$\mathcal{K}(\bar{\Omega}_{ik}^{(l)}) = \ln\left(1 + \sum_{k=1}^{M_i-1} \exp\left(\eta_{i,k}^{(l)}\right)\right) + \sum_{d=1}^{D}\left[\ln\left(\Gamma\left(a_{ik,d}^{(l)}\right)\right) + \left(a_{ik,d}^{(l)} + b_{ik,d}^{(l)} - 1\right)\ln(A) + \ln\left(\Gamma\left(b_{ik,d}^{(l)}\right)\right) + \ln\left(\Gamma\left(a_{ik,d}^{(l)} + b_{ik,d}^{(l)}\right)\right)\right]; \; \eta_{i,k}^{(l)} = \ln\left(\alpha_{i,k}^{(l)}/\alpha_{i,M_i}^{(l)}\right),$$

avec $\Gamma(.)$ désignant la fonction Gamma et $\dot{\mathbf{x}}_k$ est un vecteur de dimension $(M_i - 1)$. Le $k-$ème élément de ce vecteur est égal à 1 et les autres sont nuls.

À partir de l'expression décrite dans l'équation (A.1), nous pouvons déduire la bonne supérieure du mélange de Dirichlet généralisée (Eq. 3.10) comme démontré dans [73] pour la famille exponentielle en général. Deux paramètres variationnelles sont introduites à savoir $\ddot{\mathbf{x}}_{n,ik}^{(l)}$ et $W_{n,ik}^{(l)}$. Le paramètre $W_{n,ik}^{(l)}$ doit être généré de sorte à ce que $\ddot{\mathbf{x}}_{n,ik}^{(l)}$ appartient à l'espace du gradient de la fonction cumulant $\mathcal{K}(\bar{\Omega}_{ik}^{(l)})$. Rappelons ici l'expression de la borne supérieure décrite dans l'équation 3.10 :

$$\ln\left(\sum_k \alpha_{ik}^{(l)} GD(x_n|\mu_{ik}^{(l)})\right) \leq -\sum_k W_{n,ik}^{(l)}\left[\ddot{\mathbf{x}}_{n,ik}^{(l)T}\bar{\Omega}_{ik}^{(l)} - \mathcal{K}(\bar{\Omega}_{ik}^{(l)})\right] + cst_{n,i} \ , \tag{A.2}$$

où

$$\begin{aligned}
cst_{n,i} &= \ln\left(\sum_k \breve{\alpha}_{ik}^{(l)} GD(x_n|\breve{\mu}_{ik}^{(l)})\right) + \sum_k W_{n,ik}^{(l)}\left[\ddot{\mathbf{x}}_{n,ik}^{(l)T}\breve{\Omega}_{ik}^{(l)} - \mathcal{K}(\breve{\Omega}_{ik}^{(l)})\right], \\
\ddot{\mathbf{x}}_{n,ik}^{(l)} &= \frac{\tilde{\pi}_{k|i}^{(l)}(x_n)}{W_{n,ik}^{(l)}}\left[\mathcal{K}'(\breve{\Omega}_{ik}^{(l)}) - \bar{\mathbf{x}}_{n,ik}^{(l)}\right] + \mathcal{K}'(\breve{\Omega}_{ik}^{(l)}), \\
w_{n,ik}^{(l)} &= min\ w_{n,ik}^{(l)}\ tel\ que\ \frac{\tilde{\pi}_{k|i}^{(l)}(x_n)}{w_{n,ik}^{(l)}}\left[\mathcal{K}'(\breve{\Omega}_{ik}^{(l)}) - \bar{\mathbf{x}}_{n,ik}^{(l)}\right] + \mathcal{K}'(\breve{\Omega}_{ik}^{(l)}) \in \mathcal{K}'(\Omega_{ik}^{(l)}),
\end{aligned} \tag{A.3}$$

$$W_{n,ik}^{(l)} = 4G\left(\breve{\pi}_{k|i}^{(l)}(x_n)/2\right)\left[(\mathcal{Z}_{n,ik}^{(l)})^T\mathcal{Z}_{n,ik}^{(l)}\right] + w_{n,ik}^{(l)},$$

$$\mathcal{Z}_{n,ik}^{(l)} = \mathcal{K}''(\breve{\Omega}_{ik}^{(l)})^{-1/2}\left[\bar{\mathbf{x}}_{n,ik}^{(l)} - \mathcal{K}'(\breve{\Omega}_{ik}^{(l)})\right].$$

Le gradient de la fonction cumulant $\mathcal{K}(\bar{\Omega}_{ik}^{(l)})$ est donné par le vecteur de dimension $2D + M_i - 1$ suivant :

$$\mathcal{K}'(\bar{\Omega}_{ik}^{(l)}) = \begin{pmatrix}
\Psi(a_{ik,1}^{(l)}) - \Psi(a_{ik,1}^{(l)} + b_{ik,1}^{(l)}) + \ln(A) \\
\Psi(b_{ik,1}^{(l)}) - \Psi(a_{ik,1}^{(l)} + b_{ik,1}^{(l)}) + \ln(A) \\
\vdots \\
\Psi(a_{ik,D}^{(l)}) - \Psi(a_{ik,D}^{(l)} + b_{ik,D}^{(l)}) + \ln(A) \\
\Psi(b_{ik,D}^{(l)}) - \Psi(a_{ik,D}^{(l)} + b_{ik,D}^{(l)}) + \ln(A) \\
\left(\exp\left(\eta_{i,1}^{(l)}\right)\right)/\sum_{j=1}^{M_i}\exp\left(\eta_{i,j}^{(l)}\right) \\
\vdots \\
\left(\exp\left(\eta_{i,M_i-1}^{(l)}\right)\right)/\sum_{j=1}^{M_i}\exp\left(\eta_{i,j}^{(l)}\right)
\end{pmatrix}. \tag{A.4}$$

L'Hessien de la fonction cumulant $\mathcal{K}(\bar{\Omega}_{ik}^{(l)})$ est une matrice diagonale par bloc :

$$\mathcal{K}''(\bar{\Omega}_{ik}^{(l)}) = \begin{pmatrix} \mathcal{K}_1''(\bar{\Omega}_{ik}^{(l)}) & \mathbf{0} \\ \mathbf{0} & \mathcal{K}_2''(\bar{\Omega}_{ik}^{(l)}) \end{pmatrix}, \tag{A.5}$$

où $\mathcal{K}_1''(\bar{\Omega}_{ik}^{(l)})$ est également une matrice diagonale par bloc composée de D blocs

$$\mathcal{K}_{1,d}''(\bar{\Omega}_{ik}^{(l)}) = \begin{pmatrix} \Psi'(a_{ik,d}^{(l)}) - \Psi'(a_{ik,d}^{(l)} + b_{ik,d}^{(l)}) & -\Psi'(a_{ik,d}^{(l)} + b_{ik,d}^{(l)}) \\ -\Psi'(a_{ik,d}^{(l)} + b_{ik,d}^{(l)}) & \Psi'(b_{ik,d}^{(l)}) - \Psi'(a_{ik,d}^{(l)} + b_{ik,d}^{(l)}) \end{pmatrix}. \tag{A.6}$$

Les éléments de la matrice $\mathcal{K}_2''(\bar{\Omega}_{ik}^{(l)})$ sont donnés pour d_1 et $d_2 \in \{1, \ldots, M_i - 1\}$ par :

$$\mathcal{K}_{2,d_1 d_2}''(\bar{\Omega}_{ik}^{(l)}) = \begin{cases} \mathcal{K}'_{(2D+d_1)}(\bar{\Omega}_{ik}^{(l)}) \left[1 - \mathcal{K}'_{(2D+d_1)}(\bar{\Omega}_{ik}^{(l)})\right], & \text{si } d_1 = d_2 \\ -\mathcal{K}'_{(2D+d_1)}(\bar{\Omega}_{ik}^{(l)}) \mathcal{K}'_{(2D+d_2)}(\bar{\Omega}_{ik}^{(l)}), & \text{si } d_1 \neq d_2 \end{cases}. \tag{A.7}$$

A.2 Contraintes sur le point $\ddot{\mathbf{x}}_{n,ik}^{(l)}$

Le point variationnel $\ddot{\mathbf{x}}_{n,ik}^{(l)}$ doit appartenir à l'espace du gradient de la fonction cumulant $\mathcal{K}(\bar{\Omega}_{ik}^{(l)})$. Pour cela, nous devons calculer $w_{n,ik}^{(l)}$ comme étant la valeur minimale de $W_{n,ik}^{(l)}$ qui garantit cette contrainte sur $\ddot{\mathbf{x}}_{n,ij}^{(l)}$. Rappelons que $\bar{\mathbf{x}}_{n,ik}^{(l)}$ appartient à l'espace du gradient de la fonction cumulant $\mathcal{K}(\bar{\Omega}_{ik}^{(l)})$. Il suffit donc d'appliquer les contraintes du point $\bar{\mathbf{x}}_{n,ik}^{(l)}$ au point $\ddot{\mathbf{x}}_{n,ik}^{(l)}$. Trois conditions doivent être satisfaites :

- les $2D$ premiers éléments de $\ddot{\mathbf{x}}_{n,ik}^{(l)}$ doivent être inférieur à $\ln(A)$
- les éléments restants doivent être dans l'intervalle $]0, 1[$ et leur somme est égale à 1.

Nous pouvons déduire que $w_{n,ik}^{(l)}$ doit satisfaire la contrainte suivante :

$$w_{n,ik}^{(l)} > \hat{B}_{n,ik}^{(l)} = \max\left\{B_{n,ik,1}^{(l)}; \cdots; B_{n,ik,(2D+2M_i)}^{(l)}\right\}, \tag{A.8}$$

où

$$B_{n,ik,d}^{(l)} = \begin{cases} \text{pour } d = 1 \cdots 2D \\ \tilde{\pi}_{k|i}^{(l)}(x_n) \dfrac{\left[\bar{\mathbf{x}}_{n,ik,d}^{(l)} - \mathcal{K}_d'(\breve{\Omega}_{ik}^{(l)})\right]}{\mathcal{K}_d'(\breve{\Omega}_{ik}^{(l)}) - \ln(A)}; \\ \\ \text{pour } d = 2D+1 \cdots 2D+M_i-1 \\ \tilde{\pi}_{k|i}^{(l)}(x_n) \dfrac{\left[\bar{\mathbf{x}}_{n,ik,d}^{(l)} - \mathcal{K}_d'(\breve{\Omega}_{ik}^{(l)})\right]}{\mathcal{K}_d'(\breve{\Omega}_{ik}^{(l)})}; \\ \\ \text{pour } d = 2D+M_i \cdots 2D+2M_i-1 \\ \tilde{\pi}_{k|i}^{(l)}(x_n) \dfrac{\left[\bar{\mathbf{x}}_{n,ik,(d-M_i+1)}^{(l)} - \mathcal{K}_{(d-M_i+1)}'(\breve{\Omega}_{ik}^{(l)})\right]}{\mathcal{K}_{(d-M_i+1)}'(\breve{\Omega}_{ik}^{(l)}) - 1} \\ \\ \text{pour } d = 2D+2M_i \\ \tilde{\pi}_{k|i}^{(l)}(x_n) \dfrac{\sum_{j=2D+1}^{2D+M_i-1}\left[\bar{\mathbf{x}}_{n,ik,j}^{(l)} - \mathcal{K}_j'(\breve{\Omega}_{ik}^{(l)})\right]}{\sum_{j=2D+1}^{2D+M_i-1}\mathcal{K}_j'(\breve{\Omega}_{ik}^{(l)}) - 1} \end{cases}.$$

A.3 Inversion des matrices $\mathcal{K}''(\breve{\Omega}_{ik}^{(l)})$ et $\mathcal{H}_{ik,d}^{(l)}$

Nous avons besoin d'inverser les matrices $\mathcal{K}''(\breve{\Omega}_{ik}^{(l)})$ et $\mathcal{H}_{ik,d}^{(l)}$ plusieurs fois dans notre modèle. Ceci peut conduire à une charge calculatoire excessive et éviter cela serait bénéfique. Rappelons que $\mathcal{K}''(\breve{\Omega}_{ik}^{(l)})$ et $\mathcal{H}_{ik,d}^{(l)}$ interviennent respectivement dans l'expression de $W_{n,ik}^{(l)}$ et dans l'estimation de $\mu_{ik}^{(l)}$. Inverser $\mathcal{K}''(\breve{\Omega}_{ik}^{(l)})$ revient à l'inversion de chacune des matrices $\mathcal{K}_{1,d}''(\breve{\Omega}_{ik}^{(l)})$ et $\mathcal{K}_2''(\breve{\Omega}_{ik}^{(l)})$. Nous pouvons ré-écrire les matrices $\mathcal{K}_{1,d}''(\breve{\Omega}_{ik}^{(l)})$, $\mathcal{K}_2''(\breve{\Omega}_{ik}^{(l)})$ (équations A.6 et A.7) et $\mathcal{H}_{ik,d}^{(l)}$ (équations 3.15 et 3.16) comme suit :

A.3. Inversion des matrices $\mathcal{K}''(\breve{\Omega}_{ik}^{(l)})$ et $\mathcal{H}_{ik,d}^{(l)}$

$$\mathcal{K}''_{1,d}(\breve{\Omega}_{ik}^{(l)}) = Q_1 + e_1 \mathbf{1}\mathbf{1}^T \ \mid e_1 = -\Psi'(a_{ik,d}^{(l)} + b_{ik,d}^{(l)}),$$

$$\mathcal{K}''_{2}(\breve{\Omega}_{ik}^{(l)}) = Q_2 + e_2 \mathtt{f}\mathtt{f}^T \ \mid e_2 = -1,$$

$$\mathcal{H}_{ik,d}^{(l)} = R + e_3 \mathtt{g}\mathtt{g}^T \ \mid \mathtt{g}^T = (a_{ik,d}^{(l)}, b_{ik,d}^{(l)}),$$

$$e_3 = \Psi'(a_{ik,d}^{(l)} + b_{ik,d}^{(l)}) \sum_n H_{n,i}^{(l-1)} \left[h_{n,k|i}^{(l)} + W_{n,ik}^{(l)} \right],$$

avec

$$Q_1 = \begin{pmatrix} \Psi'(a_{ik,d}^{(l)}) & 0 \\ 0 & \Psi'(b_{ik,d}^{(l)}) \end{pmatrix},$$

$\mathbf{1}$ est un vecteur colonne rempli de 1. Les éléments $\left\{ \mathcal{K}'_d(\bar{\Omega}_{ik}^{(l)}) \right\}_{d=2D+1}^{2D+M_i-1}$ sont ceux de Q_2 et $\mathtt{f}$ à l'exception que Q_2 est une matrice diagonal et $\mathtt{f}$ est un vecteur colonne. La matrice R est une matrice diagonale de taille 2×2 ayant pour entrées :

$$\begin{pmatrix} R_1 & \\ & \\ & R_2 \end{pmatrix} = \begin{pmatrix} -[a_{ij,d}^{(l)}]^2 \Psi'(a_{ij,d}^{(l)}) \sum_n H_{n,i}^{(l-1)} \left(h_{n,j|i}^{(l)} + W_{n,ij}^{(l)} \right) + \left[\dfrac{\partial\, \Phi_1^{(l)}}{\partial \xi_{ij,d}^{(l)}} \right]_1 & \\ & \\ -[b_{ij,d}^{(l)}]^2 \Psi'(b_{ij,d}^{(l)}) \sum_n H_{n,i}^{(l-1)} \left(h_{n,j|i}^{(l)} + W_{n,ij}^{(l)} \right) + \left[\dfrac{\partial\, \Phi_1^{(l)}}{\partial \xi_{ij,d}^{(l)}} \right]_2 \end{pmatrix},$$

où $\left[\dfrac{\partial\, \Phi_1^{(l)}}{\partial \xi_{ij,d}^{(l)}} \right]_1$ et $\left[\dfrac{\partial\, \Phi_1^{(l)}}{\partial \xi_{ij,d}^{(l)}} \right]_2$ sont respectivement la première et la seconde composante de $\dfrac{\partial\, \Phi_1^{(l)}}{\partial \xi_{ij,d}^{(l)}}$. En utilisant la formule de Sherman-Morrison, les inverses peuvent s'écrire comme suit :

$$\mathcal{K}''_{1,d}(\breve{\Omega}_{ik}^{(l)})^{-1} = Q_1^{-1} - e_1^* u u^T \ \mid u^{-1} = \begin{pmatrix} \Psi'(a_{ik,d}^{(l)}) \\ \\ \Psi'(b_{ik,d}^{(l)}) \end{pmatrix},$$

$$\mathcal{K}''_{2}(\breve{\Omega}_{ik}^{(l)})^{-1} = Q_2^{-1} + e_2^* \mathbf{1}\mathbf{1}^T \ \mid \frac{1}{e_2^*} = 1 - \sum_{d=2D+1}^{2D+M_i-1} \mathcal{K}'_d(\bar{\Omega}_{ik}^{(l)}),$$

$$[\mathcal{H}_{ik,d}^{(l)}]^{-1} = R^{-1} - e_3^* \mathsf{g}^* \mathsf{g}^{*T} \mid \mathsf{g}^* = \begin{pmatrix} \frac{a_{ik,d}^{(l)}}{R_1} \\ \\ \frac{b_{ik,d}^{(l)}}{R_2} \end{pmatrix},$$

$$e_3 = e_3^* \Big[1 + e_3 \Big[\frac{[a_{ij,d}^{(l)}]^2}{R_1} + \frac{[b_{ij,d}^{(l)}]^2}{R_2} \Big] \Big] ,$$

$$e_1 = e_1^* \Big[1 + e_1 \Big[\frac{1}{\Psi'(a_{ik,d}^{(l)})} + \frac{1}{\Psi'(b_{ik,d}^{(l)})} \Big] \Big] .$$

A.4 Code Matlab pour générer la table de référence

Afin de générer une borne supérieure adéquate pour le mélange de Dirichlet généralisée, nous avons besoin des paramètres de la fonction G (voir équation 3.11). Cette fonction est sous la forme : $G(\gamma) = a\gamma + b$ [73]. Une liste de des paramètres a et b de même que le code Matlab pour les générer sont donnés dans le tableau A.1. N'importe quel couple (a, b) généré par ce code nous donne une borne supérieure valide. Dans nos travaux, nous avons utilisé $a = 0.9811$ et $b = 0.0450$.

a	b	% Code Matlab pour générer
0.8108	0.1892	D=1000;
0.8919	0.1081	G=zeros(D,1);
0.9811	0.0450	Z=G;
1.0792	0.0437	W=G;
1.1872	0.0425	OPTI=zeros(15,1);
1.3059	0.0414	OPTI(2)=1e-6;
1.4365	0.0404	OPTI(14)=59000;
1.5801	0.0394	
1.7381	0.0385	
1.9119	0.0377	for j=1:D
2.1031	0.0369	g = (1.2)^(-D+j);
2.3134	0.0361	s = @(x)(-log(g*exp(x)+g*exp(-x)-g*2+1)/(x*x));
2.5448	0.0354	w = fminbnd(s,1e-8,1000.0,OPTI);
2.7992	0.0347	G(j) = g;
3.0792	0.0341	Z(j) = log(g*exp(w)+g*exp(-w)+1-2*g)/(w*w);
3.3871	0.0335	end
3.7258	0.0329	
4.0984	0.0323	A=zeros(D/2,1);
4.5082	0.0318	B=A;
4.9590	0.0312	for j=1:(D/2)
5.4549	0.0307	a = (1.1)^(j-3.2);
6.0004	0.0303	b = max(Z-G*a);
6.6005	0.0298	A(j) = a;
7.2605	0.0293	B(j) = b;
7.9866	0.0289	end
8.7852	0.0285	
9.6638	0.0281	
10.6301	0.0277	
11.6931	0.0273	
12.8625	0.0270	

tableau A.1 – Quelques valeurs de la table de référence utilisée de même que le code Matlab pour les générer.

Annexe B

Initialisation des paramètres pour les modèles DGD et HMGD

B.1 Initialisation des paramètres du modèle DGD

L'algorithme B.1 décrit la procédure utiliser pour initialiser les paramètres du modèle DGD. Nous nous sommes basés sur la méthode des moments pour initialiser les paramètres des différentes distributions de Dirichlet généralisée [92].

Algorithme B.1 Pseudo-code pour l'initialisation des paramètres du modèle DGD

Données : $\mathbf{V} = \{\mathtt{v}_1, \cdots, \mathtt{v}_N\} \in \mathbb{R}^{D \times N}$ (obtenues à partir de l'équation 3.2) et $\mathbf{Y} = (y_1, \cdots, y_N)^T \in \mathbb{R}^N$.
Résultats : $\Omega = \{\mu_c, \alpha_c\}_{c=1}^{C}$ avec $\mu_c = \{a_{cd}, b_{cd}\}_{d=1}^{D}$.
Pour chaque classe $c\{1 \cdots C\}$ et pour $d \in \{1 \cdots D\}$

1. Sélectionner $\mathtt{v}_{cd}$ comme étant les données de la dimension d appartenant à la classe c.
2. Calculer $\ln(\bar{\mathtt{v}}_{cd,1}) = \frac{1}{N}\sum_n \ln(\mathtt{v}_{n,cd})$ et $\ln(\bar{\mathtt{v}}_{cd,2}) = \frac{1}{N}\sum_n \ln(A - \mathtt{v}_{n,cd})$.
3. Calculer $a_{cd} = \frac{1}{2}\frac{A - \bar{\mathtt{v}}_{cd,2}}{A - \bar{\mathtt{v}}_{cd,1} - \bar{\mathtt{v}}_{cd,2}}$.
4. Calculer $b_{cd} = \frac{1}{2}\frac{A - \bar{\mathtt{v}}_{cd,1}}{A - \bar{\mathtt{v}}_{cd,1} - \bar{\mathtt{v}}_{cd,2}}$.
5. Définir $\{\alpha_c\}_{c=1}^{C}$ comme étant le *prior* de chaque classe ($\alpha_c = N_c/N$) où N_c est le nombre d'instances appartenant à la classe c.

B.2 Initialisation des paramètres pour les modèles HMGD

L'algorithme B.2 décrit la procédure utiliser pour initialiser les paramètres du modèle HMGD. Nous nous sommes basés sur la méthode des moments [92] et de la procédure décrite dans [28] pour initialiser les paramètres des différentes distributions de Dirichlet généralisée .

Algorithme B.2 Pseudo-code pour l'initialisation des paramètres du modèle HMGD

Données : $\mathbf{V} = \{\mathtt{v}_1, \cdots, \mathtt{v}_N\} \in \mathbb{R}^{D\times N}$ (obtenues à partir de l'équation 3.2), $\mathbf{Y} = (y_1, \cdots, y_N)^T \in \mathbb{R}^N$, K, M_i et C.
Résultats : $\Omega^{(0)}$, $\Omega^{(1)}$ et $\Omega^{(2)}$.

1. Pour les paramètres $\Omega^{(0)}$
   (a) Appliquer la méthode *fuzzy C-means* (K groupes) aux données (sans les étiquettes $\mathbf{Y}$), les affecter aux groupes et créer une étiquette pour chaque groupe.
   (b) Appliquer l'algorithme B.1 avec en entrées $\mathbf{V}$ et les nouvelles étiquettes générées.
2. Pour les paramètres $\Omega^{(1)}$
   (a) Affecter les données aux K groupes en utilisant la fonction de sélection $\pi^{(0)}$ avec les paramètres $\Omega^{(0)}$ issues de l'étape 1.
   (b) Appliquer l'étape 1 aux données issues de chacun des groupes construit à l'item précédent. Dans ce cas, la méthode *fuzzy C-means* est appliqué pour M_i groupes.
3. Pour les paramètres $\Omega^{(2)}$
   (a) Pour chaque $i, j \in \{1\cdots K\}\times\{1\cdots M_i\}$, affecter les données aux différents experts en utilisant la fonction de sélection $\pi^{(1)}_{j|i}$ avec les paramètres $\Omega^{(1)}_i$ issues de l'étape 2.
   (b) Appliquer l'algorithme B.1 aux données affectés à chaque expert avec leurs étiquettes $\mathbf{Y}$.

Annexe C

Algorithme pour l'estimation des coefficients de corrélation

Dans les deux cas, l'intra-corrélation et l'inter-corrélation, l'équation 3.19 est équivalent à un modèle de régression linéaire. Cependant, il faut nuancer entre les deux cas. Dans le cas de l'inter-corrélation ($K_1 \neq k_2$), $\{I_{k_2}(m+i, n+j)\}_{(i,j) \in V(m,n)}$ sont les variables explicatives et $I(m, n, k_1)$ représente les étiquettes. Dans le cas, de l'intra-corrélation, certains pixels peuvent ne pas être linéairement corrélés à leurs voisins.

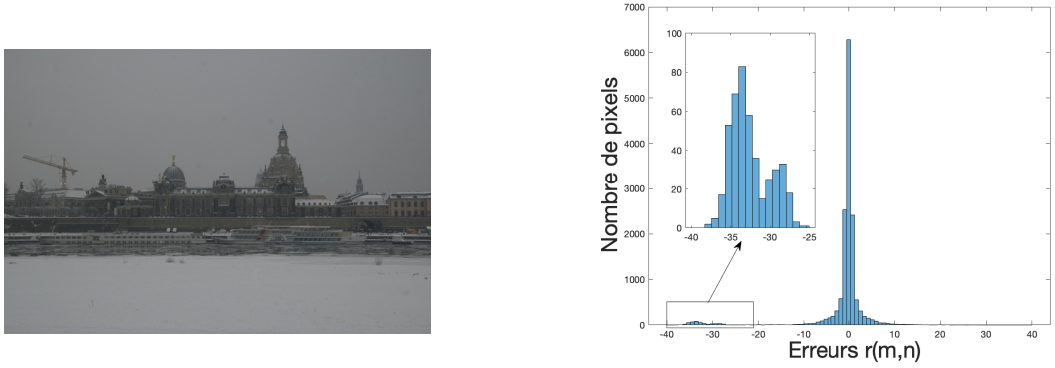


figure C.1 – (a) Une image. (b) Histogramme des erreurs $r(m, n)$ calculée pour le canal rouge de l'image affichée en (a).

Dans ce cas, les erreurs $r(m,n)$ ne sont pas distribuées selon une gaussienne de moyenne nulle. Soit $I_k;\ k \in \{R,V,B\}$ l'un des canaux de l'image I. Pour le canal rouge d'une image RVB (fig. C.1a), nous avons illustré l'histogramme des erreurs $r(m,n)$ (fig. C.1b). Le sous-tracé de la figure C.1b est un grossissement de l'histogramme pour les erreurs comprises entre -40 et -25. Comme nous pouvons remarquer, la distribution des erreurs est composée d'une gaussienne de moyenne nulle et d'une autre distribution.

Soit $I_k;\ k \in \{R,V,B\}$ un canal de l'image I et $s_{m,n}$ une variable aléatoire cachée qui indique si le pixel $I_k(m,n)$ est linéairement corrélé à ses voisins. Lorsque $s_{m,n}=1$, le pixel $I_k(m,n)$ a une distribution gaussienne de moyenne $\sum_{i,j}\gamma_{i,j,k}I_k(m+i,n+j)$ et de variance σ_k^2 :

$$\begin{aligned} &p\left(I_k(m,n) \mid s_{m,n}=1\right) = \\ &\mathcal{N}\left(I_k(m,n) \mid \sum_{i,j}\gamma_{i,j,k}I_k(m+i,n+j), \sigma_k^2\right) \end{aligned} \tag{C.1}$$

où $\mathcal{N}(\mid \mu,\sigma^2)$ est la distribution gaussienne avec μ comme moyenne et σ^2 comme variance. Si $s_{m,n}=0$, le pixel $I_k(m,n)$ peut prendre n'importe quel valeur et nous assumons qu'elle a une distribution uniforme avec :

$$p\left(I_k(m,n) \mid s_{m,n}=0\right) = \frac{1}{\textit{Nombre de valeurs possibles de } I_k(m,n)} \tag{C.2}$$

La log-vraisemblance peut s'écrire comme

$$\mathcal{L} = \sum_{m,n}\sum_{s_{m,n}} s_{m,n} \ln\left[p\left(I_k(m,n) \mid s_{m,n}\right)\right] \tag{C.3}$$

Nous pouvons utiliser l'algorithme EM pour estimer les paramètres $\gamma_{i,j,k}$ et σ. Dans la phase E-step, nous calculons l'espérance suivante :

$$\begin{aligned} \mathcal{Q} = \sum_{m,n} & h_1(m,n)\ln\left[p\left(I_k(m,n) \mid s_{m,n}=1\right)\right] \\ & + h_0(m,n)\ln\left[p\left(I_k(m,n) \mid s_{m,n}=0\right)\right] \end{aligned} \tag{C.4}$$

où

$$\begin{aligned} h_1(m,n) &= p\left(s_{m,n}=1 \mid I_k(m,n)\right) \\ &= \frac{p\big(I_k(m,n)|s_{m,n}=1\big)p\big(s_{m,n}=1\big)}{\sum_{i=0}^{1} p\big(I_k(m,n)|s_{m,n}=i\big)p\big(s_{m,n}=i\big)} \end{aligned} \tag{C.5}$$

et

$$\begin{aligned} h_0(m,n) &= p\left(s_{m,n}=0 \mid I_k(m,n)\right) \\ &= \frac{p\big(I_k(m,n)|s_{m,n}=0\big)p\big(s_{m,n}=0\big)}{\sum_{i=0}^{1} p\big(I_k(m,n)|s_{m,n}=i\big)p\big(s_{m,n}=i\big)} \end{aligned} \tag{C.6}$$

Les probabilités *a priori* $p\left(s_{m,n}=0\right)$ et $p\left(s_{m,n}=1\right)$ ont été choisies comme étant égale à $1/2$. Dans la phase M-Step, nous devons estimer $\gamma_{i,j,k}$ et σ en maximisant $\mathcal{Q}$ (Eq. C.4). L'expression de σ est donné par :

$$\sigma^2 = \frac{\sum\limits_{m,n} h_1(m,n)\left[I_k(m,n) - \sum\limits_{i,j}\gamma_{i,j,k} I_k(m+i,n+j)\right]^2}{\sum\limits_{m,n} h_1(m,n)} \tag{C.7}$$

Estimé $\gamma_{i,j,k}$ est équivalent à minimiser

$$E = \sum_{m,n} h_1(m,n)\left(I_k(m,n) - \sum_{i,j}\gamma_{i,j,k} I_k(m+i,n+j)\right)^2 \tag{C.8}$$

En annulant la dérivée de E par rapport à $\gamma_{i,j,k}$, nous avons à résoudre l'équation suivante :

$$A\gamma_k = Y \tag{C.9}$$

où A est une matrice symétrique avec

$$A = \begin{pmatrix} a_{1,1} & a_{2,1} & \cdots & a_{r,1} \\ a_{2,1} & a_{2,2} & \cdots & a_{r,2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r,1} & a_{r,2} & \cdots & a_{r,r} \end{pmatrix} ; \gamma_k = \begin{pmatrix} \gamma_{-J,-J,k} \\ \gamma_{1-J,-J,k} \\ \vdots \\ \gamma_{J,J,k} \end{pmatrix} \tag{C.10}$$

et

$$a_{1,1} = \sum_{m,n} h_1(m,n) \left[I_k(m-J, n-J) \right]^2 \tag{C.11}$$

$$a_{2,2} = \sum_{m,n} h_1(m,n) \left[I_k(m-J+1, n-J) \right]^2 \tag{C.12}$$

$$a_{r,r} = \sum_{m,n} h_1(m,n) \left[I_k(m+J, n+J) \right]^2 \tag{C.13}$$

$$a_{2,1} = \sum_{m,n} \left[h_1(m,n)\, I_k(m-J+1, n-J)\, I_k(m-J, n-J) \right] \tag{C.14}$$

$$a_{r,1} = \sum_{m,n} \left[h_1(m,n)\, I_k(m+J, n+J)\, I_k(m-J, n-J) \right] \tag{C.15}$$

$$a_{r,2} = \sum_{m,n} \left[h_1(m,n)\, I_k(m+J, n+J)\, I_k(m-J+1, n-J) \right] \tag{C.16}$$

$$Y = \begin{pmatrix} \sum_{m,n} h_1(m,n)\, I_k(m,n)\, I_k(m-J, n-J) \\ \sum_{m,n} h_1(m,n)\, I_k(m,n)\, I_k(m-J+1, n-J) \\ \vdots \\ \sum_{m,n} h_1(m,n)\, I_k(m,n)\, I_k(m+J, n+J) \end{pmatrix} \tag{C.17}$$

L'estimation des paramètres est résumée dans l'algorithme C.1 qui est similaire à celui proposé par Popescu et Farid [105].

Algorithme C.1 Pseudo-code pour l'estimation des coefficients de corrélation.

Données : I_k.
Résultats : $\gamma_{i,j,k}$.
1. Initialisation :
Choisir aléatoirement $\gamma_{i,j,k}$, poser $p\left(I_k(m,n) \mid s_{m,n}=0\right) = 1/256$ et $\sigma_k = 0.28$.
2. E-step : Pour chaque pixel $I_k(m,n)$, calculer $h_1(m,n)$ (Eq. C.5).

3. M-step : Estimer respectivement σ_k et $\gamma_{i,j,k}$ en utilisant l'équation C.7 et en résolvant l'équation C.9.

4. Répéter les étapes 2 et 3 jusqu'à convergence.

Bibliographie

[1] E. Abbasi, M. E. Shiri, and M. Ghatee. Root-quatric mixture of experts for complex classification problems. *Expert Systems with Applications*, 53 :192 – 203, jul 2016.

[2] E. Abbasi, M. E. Shiri, and M. Ghatee. A regularized root–quartic mixture of experts for complex classification problems. *Knowledge-Based Systems*, 110 :98 – 109, oct 2016.

[3] C. C. Aggarwal. Instance-based learning : A survey. *Data classification : algorithms and applications*, 157, 2014.

[4] C. C. Aggarwal. Neural networks and deep learning. *Cham : Springer International Publishing*, 2018.

[5] N. Ahmed and M. Campbell. On estimating simple probabilistic discriminative models with subclasses. *Expert Systems with Applications*, 39(7) :6659–6664, June 2012.

[6] J. Aitchison. The statistical analysis of compositional data. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 139–177, 1982.

[7] M. S. Allili, N. Bouguila, and D. Ziou. Finite general gaussian mixture modeling and application to image and video foreground segmentation. *Journal of Electronic Imaging*, 17(1) :013005, 2008.

[8] D. Ankam and N. Bouguila. Compositional data analysis with pls-da and security applications. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 338–345, 2018.

[9] S. Banitaan, A. B. Nassif, and M. Azzeh. Class decomposition using k-means and hierarchical clustering. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 1263–1267. IEEE, 2015.

[10] D. Barbella, S. Benzaid, J. M. Christensen, B. Jackson, X. V. Qin, and D. R. Musicant. Understanding support vector machine classifications via a recommender system-like approach. In *DMIN*, pages 305–311. Las Vegas, NV, 2009.

[11] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural computation*, 12(10) :2385–2404, 2000.

[12] T. Bdiri, N. Bouguila, and D. Ziou. Object clustering and recognition using multi-finite mixtures for semantic classes and hierarchy modeling. *Expert Systems with Applications*, 41(4, Part 1) :1218–1235, Mar. 2014.

[13] Y. Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1) :1–127, 2009.

[14] J. Bernardo, M. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. Smith, and M. West. Generative or discriminative ? getting the best of both worlds. *Bayesian statistics*, 8(3) :3–24, 2007.

[15] M. Bevilacqua and F. Marini. Local classification : Locally weighted-partial least squares-discriminant analysis (LW-PLS-DA). *Analytica Chimica Acta*, 838 :20–30, Aug. 2014.

[16] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[17] J. Błaszczyński, A. T. de Almeida Filho, A. Matuszyk, M. Szeląg, and R. Słowiński. Auto loan fraud detection using dominance-based rough set approach versus machine learning methods. *Expert Systems with Applications*, 163 : 113740, 2021.

[18] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152. ACM, 1992.

[19] G. Bouchard and B. Triggs. The tradeoff between generative and discriminative classifiers. In *16th IASC International Symposium on Computational Statistics (COMPSTAT'04)*, pages 721–728, 2004.

[20] N. Bouguila. Bayesian hybrid generative discriminative learning based on finite liouville mixture models. *Pattern Recognition*, 44(6) :1183–1200, 2011.

[21] N. Bouguila. Hybrid generative/discriminative approaches for proportional data modeling and classification. *IEEE Transactions on Knowledge and Data Engineering*, 24(12) :2184–2202, 2012.

[22] N. Bouguila. Deriving kernels from generalized dirichlet mixture models and applications. *Information Processing & Management*, 49(1) :123–137, 2013.

[23] N. Bouguila and D. Ziou. A hybrid sem algorithm for high-dimensional unsupervised learning using a finite generalized dirichlet mixture. *IEEE Transactions on Image Processing*, 15(9) :2657–2668, 2006.

[24] N. Bouguila and D. Ziou. Online clustering via finite mixtures of dirichlet and minimum message length. *Engineering Applications of Artificial Intelligence*, 19(4) :371–379, 2006.

[25] N. Bouguila and D. Ziou. A dirichlet process mixture of dirichlet distributions for classification and prediction. In *2008 IEEE workshop on machine learning for signal processing*, pages 297–302. IEEE, 2008.

[26] N. Bouguila and D. Ziou. A dirichlet process mixture of generalized dirichlet distributions for proportional data modeling. *IEEE Transactions on Neural Networks*, 21(1) :107–122, 2010.

[27] N. Bouguila and D. Ziou. A countably infinite mixture model for clustering and feature selection. *Knowledge and information systems*, 33(2) :351–370, 2012.

[28] N. Bouguila, D. Ziou, and J. Vaillancourt. Unsupervised learning of a finite mixture model based on the dirichlet distribution and its application. *IEEE Transactions on Image Processing*, 13(11) :1533–1543, 2004.

[29] S. Bourouis, A. Zaguia, N. Bouguila, and R. Alroobaea. Deriving probabilistic svm kernels from flexible statistical mixture models and its application to retinal images classification. *IEEE Access*, 7 :1107–1117, 2018.

[30] S. Boutemedjet, N. Bouguila, and D. Ziou. A hybrid feature extraction selection approach for high-dimensional non-gaussian data clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(8) :1429–1443, 2008.

[31] L. Breiman. Bagging predictors. *Machine learning*, 24(2) :123–140, 1996.

[32] L. Breiman, J. Friedman, C. J. Stone, and R. . Olshen. *Classification and regression trees*. CRC press, 1984.

[33] T. J. Brinker, A. Hekler, J. S. Utikal, N. Grabe, D. Schadendorf, J. Klode, C. Berking, T. Steeb, A. H. Enk, and C. von Kalle. Skin cancer classification using convolutional neural networks : systematic review. *Journal of medical Internet research*, 20(10) :e11936, 2018.

[34] K. Burse, M. Manoria, and V. P. S. Kirar. Improved back propagation algorithm to avoid local minima in multiplicative neuron model. In V. V. Das, G. Thomas, and F. L. Gaol, editors, *Information Technology and Mobile Communication*, number 147 in Communications in Computer and Information Science, pages 67–73. Springer Berlin Heidelberg, Jan. 2011.

[35] L. Carvalho and A. von Wangenheim. 3d object recognition and classification : a systematic literature review. *Pattern Analysis and Applications*, 22(4) :1243–1292, 2019.

[36] H. Cevikalp and R. Polikar. Local classifier weighting by quadratic programming. *IEEE Transactions on Neural Networks*, 19(10) :1832–1838, Oct. 2008.

[37] C.-C. Chang and C.-J. Lin. LIBSVM : A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2 :27 :1–27 :27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[38] K. Chen, L. Xu, and H. Chi. Improved learning algorithms for mixture of experts in multiclass classification. *Neural networks*, 12(9) :1229–1252, 1999.

[39] K. Chen, J. Hu, and J. He. Detection and classification of transmission line faults based on unsupervised feature learning and convolutional sparse autoencoder. *IEEE Transactions on Smart Grid*, 9(3) :1748–1758, 2018.

[40] G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han. When deep learning meets metric learning : remote sensing image scene classification via learning discriminative cnns. *IEEE transactions on geoscience and remote sensing*, 56(5) : 2811–2821, 2018.

[41] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv :1710.09282*, 2017.

[42] R. M. Cruz, R. Sabourin, G. D. Cavalcanti, and T. I. Ren. Meta-des : A dynamic ensemble selection framework using meta-learning. *Pattern Recognition*, 48(5) : 1925–1935, 2015.

[43] R. M. Cruz, R. Sabourin, and G. D. Cavalcanti. Dynamic classifier selection : Recent advances and perspectives. *Information Fusion*, 41 :195–216, 2018.

[44] E. G. Dada, J. S. Bassi, H. Chiroma, A. O. Adetunmbi, O. E. Ajibuwa, et al. Machine learning for email spam filtering : review, approaches and open research problems. *Heliyon*, 5(6) :e01802, 2019.

[45] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato. Raise : A raw images dataset for digital image forensics. In *Proceedings of the 6th ACM Multimedia Systems Conference*, pages 219–224. ACM, 2015.

[46] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[47] K. Deng. *OMEGA : On-line memory-based general purpose system classifier*. PhD thesis, Georgia Institute of Technology, 1998.

[48] J. J. Egozcue, V. Pawlowsky-Glahn, G. Mateu-Figueras, and C. Barcelo-Vidal. Isometric logratio transformations for compositional data analysis. *Mathematical Geology*, 35(3) :279–300, 2003.

[49] A. El-Zaart and D. Ziou. Statistical modelling of multimodal sar images. *International Journal of Remote Sensing*, 28(10) :2277–2294, 2007.

[50] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639) :115, 2017.

[51] W. Fan, F. R. Al-Osaimi, N. Bouguila, and J. Du. Proportional data modeling via entropy-based variational bayes learning of mixture models. *Applied Intelligence*, 47(2) :473–487, 2017.

[52] W. Fan, H. Sallay, and N. Bouguila. Online learning of hierarchical pitman-yor process mixture of generalized dirichlet distributions with feature selection. *IEEE transactions on neural networks and learning systems*, 2017.

[53] J. M. Fossaceca, T. A. Mazzuchi, and S. Sarkani. Mark-elm : Application of a novel multiple kernel learning framework for improving the robustness of network intrusion detection. *Expert Systems with Applications*, 42(8) :4062–4080, 2015.

[54] E. Frank, M. Hall, and B. Pfahringer. Locally weighted naive bayes. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, UAI'03, pages 249–256. Morgan Kaufmann Publishers Inc., 2003.

[55] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156, 1996.

[56] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression : a statistical view of boosting. *Annals of Statistics*, 28 :2000, 1998.

[57] S. Ghodratnama and H. A. Moghaddam. Content-based image retrieval using feature weighting and c-means clustering in a multi-label classification framework. *Pattern Analysis and Applications*, 24(1) :1–10, 2021.

[58] A. Ghorbani, A. Abid, and J. Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688, 2019.

[59] T. Gloe and R. Böhme. The'dresden image database'for benchmarking digital image forensics. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1584–1590. Acm, 2010.

[60] M. Gönen and E. Alpaydin. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12 :2211–2268, July 2011.

[61] M. Greenacre. *Compositional data analysis in practice*. Chapman and Hall/-CRC, 2018.

[62] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77 :354–377, 2018.

[63] T. K. Gupta and K. Raza. Optimization of ann architecture : A review on nature-inspired techniques. In *Machine Learning in Bio-Signal Analysis and Diagnostic Imaging*, pages 159–182. Elsevier, 2019.

[64] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software : an update. *ACM SIGKDD explorations newsletter*, 11(1) :10–18, 2009.

[65] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8) :832–844, Aug. 1998.

[66] H. Hong, J. Liu, D. T. Bui, B. Pradhan, T. D. Acharya, B. T. Pham, A.-X. Zhu, W. Chen, and B. B. Ahmad. Landslide susceptibility mapping using j48 decision tree with adaboost, bagging and rotation forest ensembles in the guangchang area (china). *Catena*, 163 :399–413, 2018.

[67] G. Huang, G.-B. Huang, S. Song, and K. You. Trends in extreme learning machines : A review. *Neural Networks*, 61 :32–48, 2015.

[68] A. Ibrahim, A. Tharwat, T. Gaber, and A. E. Hassanien. Optimized superpixel and adaboost classifier for human thermal face recognition. *Signal, Image and Video Processing*, 12(4) :711–719, 2018.

[69] O. İrsoy and E. Alpaydın. Dropout regularization in hierarchical mixture of experts. *Neurocomputing*, 419 :148–156, 2021.

[70] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1) :79–87, 1991.

[71] G. Jain, M. Sharma, and B. Agarwal. Optimizing semantic lstm for spam detection. *International Journal of Information Technology*, 11(2) :239–250, 2019.

[72] P. Jeatrakul and K. Wong. Comparing the performance of different neural networks for binary classification problems. In *2009 Eighth International Symposium on Natural Language Processing*, pages 111–115. IEEE, Oct. 2009.

[73] T. Jebara and A. P. Pentland. *Discriminative, generative and imitative learning.* PhD thesis, PhD thesis, Media laboratory, MIT, 2001.

[74] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. In *Proceedings of 1993 International Joint Conference on Neural Networks, 1993. IJCNN '93-Nagoya*, volume 2, pages 1339–1344, Oct. 1993.

[75] S. R. Kheradpisheh, F. Sharifizadeh, A. Nowzari-Dalini, M. Ganjtabesh, and R. Ebrahimpour. Mixture of feature specified experts. *Information Fusion*, 20 : 242–251, 2014.

[76] S. Kotsiantis. Combining bagging, boosting, rotation forest and random subspace methods. *Artificial Intelligence Review*, 35(3) :223–240, 2011.

[77] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[78] R. K. Kumar, G. Poonkuzhali, and P. Sudhakar. Comparative study on email spam classifier using data mining techniques. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, pages 14–16, 2012.

[79] S. Y. Kung. *Kernel Methods and Machine Learning*. Cambridge University Press, Apr. 2014.

[80] S. Lai, L. Xu, K. Liu, and J. Zhao. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[81] H. Lancaster. The helmert matrices. *The American Mathematical Monthly*, 72 (1) :4–12, 1965.

[82] N. R. Langley, B. Dudzik, and A. Cloutier. A decision tree for nonmetric sex assessment from the skull. *Journal of forensic sciences*, 63(1) :31–37, 2018.

[83] L. Li, B. Zou, Q. Hu, X. Wu, and D. Yu. Dynamic classifier ensemble using classification confidence. *Neurocomputing*, 99 :581–591, 2013.

[84] M. Lichman. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2013.

[85] C. Liu and H. Wechsler. Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. *IEEE Transactions on Image Processing*, 11(4) :467–476, Apr. 2002.

[86] R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–7. Association for Computational Linguistics, 2002.

[87] V. Maria, S. Tatyana, S. Tatyana, and K. Tatyana. Spam and phishing in 2019. https://securelist.com/spam-report-2019/96527/, Apr. 2020.

[88] J. A. Martín-Fernández, C. Barceló-Vidal, and V. Pawlowsky-Glahn. Dealing with zeros and missing values in compositional data sets using nonparametric imputation. *Mathematical Geology*, 35(3) :253–278, 2003.

[89] W. Masoudimansour and N. Bouguila. Generalized dirichlet mixture matching projection for supervised linear dimensionality reduction of proportional data. In *Multimedia Signal Processing (MMSP), 2016 IEEE 18th International Workshop on*, pages 1–6. IEEE, 2016.

[90] S. Masoudnia and R. Ebrahimpour. Mixture of experts : a literature survey. *Artificial Intelligence Review*, 42(2) :275–293, 2014. ISSN 1573-7462.

[91] R. Meo, D. Bachar, and D. Ienco. Lode : A distance-based classifier built on ensembles of positive and negative observations. *Pattern Recognition*, 45(4) : 1409–1425, 2012.

[92] T. Minka. Estimating a dirichlet distribution, 2000.

[93] T. Minka. A comparison of numerical optimizers for logistic regression. *Unpublished draft*, pages 1–18, 2003.

[94] P. Moerland. Classification using localized mixture of experts. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 2, pages 838–843, 1999.

[95] P. Moerland. *Mixture models for unsupervised and supervised learning*. PhD thesis, École Polytechnique Fédérale de Lausanne, Computer Science Department, 2000.

[96] Y. Motai. Kernel association for classification and prediction : A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2014.

[97] S. Mou, P. Du, and Z. Cheng. A brain-inspired information processing algorithm and its application in text classification. *Expert Systems with Applications*, page 114828, 2021.

[98] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers : A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.

[99] S. Ng, C. Cheung, S. Leung, and A. Luk. Fast convergence for backpropagation network with magnified gradient function. In *Proceedings of the International Joint Conference on Neural Networks, 2003*, volume 3, pages 1903–1908, July 2003.

[100] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151) :773–782, 1980.

[101] K. Oehler and R. Gray. Combining image compression and classification using vector quantization. *IEEE transactions on pattern analysis and machine intelligence*, 17(5) :461–473, May 1995.

[102] J. B. Oliva, A. Dubey, A. G. Wilson, B. Póczos, J. Schneider, and E. P. Xing. Bayesian nonparametric kernel-learning. In *Artificial Intelligence and Statistics*, pages 1078–1086, 2016.

[103] R. Pascanu. *On Recurrent and Deep Neural Networks*. PhD thesis, Université de Montréal, 2015.

[104] W. Peng, L. Huang, J. Jia, and E. Ingram. Enhancing the naive bayes spam filter through intelligent text modification detection. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 849–854. IEEE, 2018.

[105] A. C. Popescu and H. Farid. Exposing digital forgeries in color filter array interpolated images. *IEEE Transactions on Signal Processing*, 53(10) :3948–3959, 2005.

[106] J. R. Quinlan. *C4.5 : programs for machine learning*, volume 1. Morgan kaufmann, 1993.

[107] R. Jenssen. Mean vector component analysis for visualization and clustering of nonnegative data. *IEEE Transactions on Neural Networks and Learning Systems*, 24(10) :1553–1564, 2013.

[108] A. Rahman and B. Verma. Cluster-based ensemble of classifiers. *Expert Systems*, 30(3) :270–282, 2013.

[109] A. Rahman and B. Verma. Ensemble classifier generation using non-uniform layered clustering and genetic algorithm. *Knowledge-Based Systems*, 43 :30–42, 2013.

[110] V. Ramamurti and J. Ghosh. Structurally adaptive localized mixtures of experts for non-stationary environments. *Department of Electrical and Computer Engineering. UT Austin. TX 78712-1084, USA*, 1996.

[111] V. Ramamurti and J. Ghosh. Advances in using hierarchical mixture of experts for signal classification. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 6, pages 3569–3572. IEEE, 1996.

[112] W. Rawat and Z. Wang. Deep convolutional neural networks for image classification : A comprehensive review. *Neural computation*, 29(9) :2352–2449, 2017.

[113] N. Saidani, K. Adi, and M. S. Allili. A semantic-based classification approach for an enhanced spam detection. *Computers & Security*, page 101716, 2020.

[114] R. E. Schapire. The boosting approach to machine learning : An overview. In *Nonlinear estimation and classification*, pages 149–171. Springer, 2003.

[115] D. Scherer, A. Müller, and S. Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, pages 92–101. Springer, 2010.

[116] M. Schmidt. minfunc : unconstrained differentiable multivariate optimization in matlab. *URL http ://www.di.ens.fr/mschmidt/Software/minFunc.html*, 2012.

[117] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Artificial Neural Networks ICANN'97*, number 1327 in Lecture Notes in Computer Science, pages 583–588. Springer Berlin Heidelberg, Jan. 1997.

[118] B. Shahbaba and R. Neal. Nonlinear models using dirichlet process mixtures. *Journal of Machine Learning Research*, 10(Aug) :1829–1850, 2009.

[119] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[120] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*, 2014.

[121] Y. Song, J. J. Zou, H. Chang, and W. Cai. Adapting fisher vectors for histopathology image classification. In *Biomedical Imaging (ISBI 2017), 2017 IEEE 14th International Symposium on*, pages 600–603. IEEE, 2017.

[122] D. F. Specht. Probabilistic neural networks. *Neural Networks*, 3(1) :109–118, 1990.

[123] V. Suárez-Paniagua and I. Segura-Bedmar. Evaluation of pooling operations in convolutional architectures for drug-drug interaction extraction. *BMC bioinformatics*, 19(8) :209, 2018.

[124] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems*, pages 1988–1996, 2014.

[125] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-15(1) :116–132, Jan. 1985.

[126] O. R. Terrades, E. Valveny, and S. Tabbone. Optimal classifier fusion in a non-bayesian probabilistic framework. *IEEE transactions on pattern analysis and machine intelligence*, 31(9) :1630–1644, 2008.

[127] E. Togban and D. Ziou. Classification using mixture of discriminative learners : The case of compositional data. In *International Conference Image Analysis and Recognition*, pages 416–425. Springer, 2017.

[128] E. Togban and D. Ziou. Classification using hierarchical mixture of discriminative learners : How to achieve high scores with few resources ? *Expert Systems with Applications*, 96 :14–24, 2018.

[129] M. Tsagris, S. Preston, and A. T. Wood. Improved classification for compositional data using the α-transformation. *Journal of Classification*, 33(2) :243–261, 2016.

[130] K. Tsunoda, Y. Yamane, M. Nishizaki, and M. Tanifuji. Complex objects are represented in macaque inferotemporal cortex by the combination of feature columns. *Nature Neuroscience*, 4(8) :832–838, 2001. ISSN 1097-6256. doi : 10.1038/90547.

[131] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24, 1963.

[132] M. Ventresca and H. Tizhoosh. Improving the convergence of backpropagation by opposite transfer functions. In *International Joint Conference on Neural Networks, 2006. IJCNN '06*, pages 4777–4784, 2006.

[133] M. Vezina, D. Ziou, and F. Kerouh. Color space identification for image display. In M. Kamel and A. Campilho, editors, *Image Analysis and Recognition*, pages 465–472, Cham, 2015. Springer International Publishing.

[134] M. Woźniak, M. Graña, and E. Corchado. A survey of multiple classifier systems as hybrid systems. *Information Fusion*, 16 :3–17, Mar. 2014.

[135] L. Xu, M. I. Jordan, and G. E. Hinton. An alternative model for mixtures of experts. In *Advances in neural information processing systems*, pages 633–640, 1995.

[136] H. Ykhlef and D. Bouchaffra. Induced Subgraph Game for Ensemble Selection. In *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 636–643, Nov. 2015.

[137] S. E. Yuksel and P. D. Gader. Mixture of hmm experts with applications to landmine detection. In *2012 IEEE International Geoscience and Remote Sensing Symposium*, pages 6852–6855. IEEE, 2012.

[138] S. E. Yuksel, J. N. Wilson, and P. D. Gader. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8) :1177–1193, 2012.

[139] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang. Efficient knn classification with different numbers of nearest neighbors. *IEEE transactions on neural networks and learning systems*, 29(5) :1774–1785, 2018.

[140] Y. Zhang, Y. Yin, D. Guo, X. Yu, and L. Xiao. Cross-validation based weights and structure determination of chebyshev-polynomial neural networks for pattern classification. *Pattern Recognition*, 47(10) :3414–3428, Oct. 2014.

[141] Z.-H. Zhou. *Ensemble Methods : Foundations and Algorithms*. Chapman & Hall/CRC, 1st edition, 2012.

[142] J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*, 14(1) :185–205, Mar. 2005.