Sorbonne Université
Master ANDROIDE

Classifieurs prudents : Décomposition par paires de classes

MADI

Jules Bouton Popper
Amélie Sun

Année universitaire 2023-2024

Table des matières

Chapitre 1

Présentation du classifieur

Nous avons choisi d'étudier un article [1] présentant une stratégie de classification prudente multi-classe, basée sur des schémas de décomposition et une méthode de combinaison des différents classifieurs. Ils permettent ensuite de faire une prédiction sous la forme d'un ensemble de classes.

1.1 Décomposition par paires de classes

On peut illustrer la technique de décomposition par paire de classes par le schéma sur la figure 1.1. On décompose le problème à K classes en I problèmes binaires sur lesquels on entraîne des classifieurs C_i prudents, renvoyant un intervalle de probabilité $\left[\underline{P(A_i)}, \overline{P(A_i)}\right]$. Enfin, on combine, en corrigeant les inconsistances entre intervalles, pour prédire l'ensemble $\mathcal{P}$ de classes en réponse au problème original.

1.1.1 Décomposition

L'idée derrière la décomposition est de se ramener à un problème binaire que l'on sait bien traiter dans un premier temps avant de combiner les résultats. Plusieurs stratégies peuvent être implémentées, comme le One-Versus-One (OVO) ou le One-Versus-All (OVA), qui sont les plus répandues. Il se trouve que ces stratégies sont des cas particuliers des *error-correcting output codes* (ECOC) [2]. On va donc considérer les stratégies suivantes :

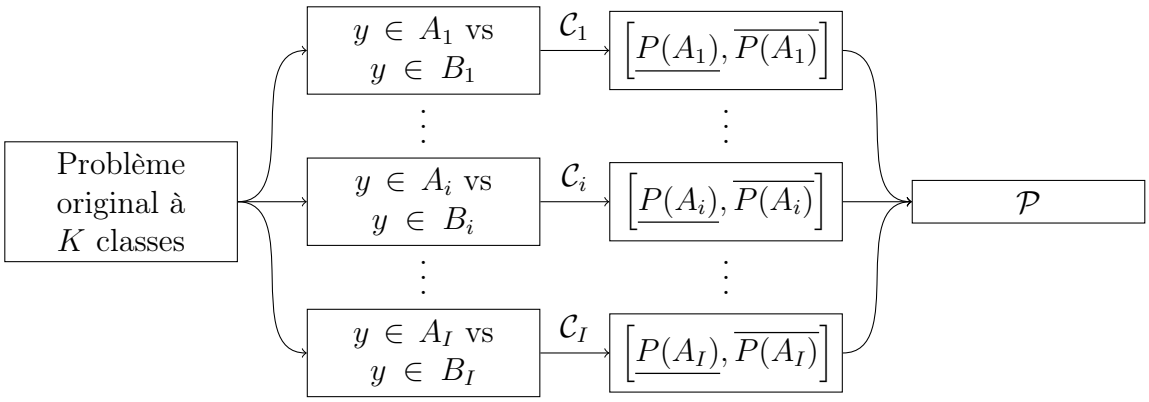

FIGURE 1.1 – Schéma de décomposition

$$\begin{pmatrix} +1 & 0 & 0 & -1 & +1 & 0 \\ -1 & +1 & 0 & 0 & 0 & +1 \\ 0 & -1 & +1 & 0 & -1 & 0 \\ 0 & 0 & -1 & +1 & 0 & -1 \end{pmatrix}$$

FIGURE 1.2 – OVO

$$\begin{pmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{pmatrix}$$

FIGURE 1.3 – OVA

$$\begin{pmatrix} +1 & -1 & -1 & -1 & +1 & -1 \\ -1 & +1 & 0 & 0 & 0 & +1 \\ 0 & -1 & +1 & 0 & -1 & 0 \\ -1 & 0 & +1 & +1 & 0 & -1 \end{pmatrix}$$

FIGURE 1.4 – *sparse*

$$\begin{pmatrix} +1 & +1 & +1 & -1 & -1 & +1 \\ -1 & +1 & -1 & -1 & +1 & -1 \\ +1 & -1 & +1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 & +1 & +1 \end{pmatrix}$$

FIGURE 1.5 – *dense*

FIGURE 1.6 – Exemples de matrices ECOC

— OVO ;
— OVA ;
— ECOC *sparse*, dont les classifieurs ne prennent pas toutes les classes ;
— ECOC *dense*, dont les classifieurs prennent toutes les classes.

Sur la figure 1.6 on peut voir la relation entre OVO, OVA et ECOC ainsi que des exemples de schémas décrit par des matrices ECOC.

1.1.2 Classifieur binaire imprécis

Le classifieur binaire implémenté est un arbre de décision CART [3]. Ce classifieur n'est pas imprécis par nature, donc il faut imprécifier sa prédiction. La méthode proposée dans l'article vise à différencier les feuilles de forte densité, contenant beaucoup d'échantillons, des zones de faible densité, avec très peu d'échantillons. Autrement dit, plus la feuille qui a permis de faire la décision aura d'exemple d'entraînement, moins l'intervalle sera large. Inversement, il sera large pour des feuilles de faible densité. On va donc utiliser la notion d'intervalle de crédibilité Bayésien pour cacluler les intervalles. L'article prend pour *a priori* une loi Bêta de paramètres $a_j = b_j = 3.5$, selon l'exemple donné en référence [4, p. 261]. Nous avons donc utilisé la formule de mise à jour de l'exemple qui est $p|x \sim \mathcal{B}e(x + a_j, n - x + b_j)$ où x est le nombre d'échantillons de la feuille de décision et n le nombre d'échantillons de la classe dans l'arbre. Nous avons pris $\alpha = 5\%$ pour le calcul des bornes. On a maintenant un classifieur binaire imprécis à notre disposition, qui est prudent quand la décision est faite à partir de peu d'exemples.

1.1.3 Combinaison des intervalles de probabilité

Chaque sous-problème consiste à entraîner un classifieur binaire probabilistes, dans le but de distinguer deux ensembles A_i et B_i (sous-ensembles de $\Omega = \{\omega_1, ..., \omega_K\}$). Lorsqu'une nouvelle instance x est présentée, chaque sous-classifieur génère une estimation de probabilité conditionnelle (1.1), fournissant des indications sur la probabilité que l'instance x

appartienne à A_i ou B_i.

$$\hat{P}(A_i|A_i, B_i, x) = 1 - \hat{P}(B_i|A_i, B_i, x) \tag{1.1}$$

Toutefois, l'article s'intéresse aux classifieurs imprécis fournissant des bornes inférieures et supérieures sur les probabilités conditionnelles (évaluation imprécise) plutôt qu'une estimation unique :

$$[\underline{P}(A_i|A_i, B_i, x), \overline{P}(A_i|A_i, B_i, x)] \tag{1.2}$$

Cet intervalle peut refléter l'incertitude résultant du manque d'information. Dorénavant, nous supprimerons la variable x dans les énoncés conditionnels, puisque la combinaison concerne toujours une instance unique dont les caractéristiques d'entrée demeurent constantes. De plus, désignons par α_j, β_j les bornes fournies par le j-ème classifieur (avec j = 1,...,J) :

$$\alpha_j \leq P(A_j|A_j \cup B_j) \leq \beta_j \tag{1.3}$$

$$1 - \beta_j \leq P(B_j|A_j \cup B_j) \leq \alpha_j \tag{1.4}$$

Après avoir collecté les intervalles [α_j,β_j] de tous les classifieurs, il faut résoudre un programme linéaire P1 où les contraintes ont été affaiblies pour rendre le problème faisable ($\epsilon \in [0, 1]$).

$$\min \quad \sum_{j=1}^{J} \epsilon_j$$

$$\text{subject to} \quad \begin{cases} \sum_{k=1}^{K} p_k = 1 \\ \epsilon_j(-\alpha_j) \leq (1 - \alpha_j)P(A_j) - \alpha_j P(B_j) \ \forall j \in \{1, ..., J\} \\ \epsilon_j(\beta_j - 1) \leq (\beta_j - 1)P(A_j) + \beta_j P(B_j) \ \forall j \in \{1, ..., J\} \\ 0 \leq p_k \leq 1 \ \forall k \in \{1, ..., K\} \\ 0 \leq \epsilon_j \leq 1 \ \forall j \in \{1, ..., J\} \end{cases} \tag{P1}$$

Après avoir résolu le programme linéaire P1, on récupère les facteurs de réduction ϵ_j des classifieurs et on résout le programme linéaire P2 ($\forall k \in \{1, ..., K\}$, $\forall l \in \{1, ..., K\}$, et $l \neq k$).

$$\min_{p} \quad p_k - p_l$$

$$\text{subject to} \quad \begin{cases} \sum_{k=1}^{K} p_k = 1 \\ \epsilon_j(-\alpha_j) \leq (1 - \alpha_j)P(A_j) - \alpha_j P(B_j) \ \forall j \in \{1, ..., J\} \\ \epsilon_j(\beta_j - 1) \leq (\beta_j - 1)P(A_j) + \beta_j P(B_j) \ \forall j \in \{1, ..., J\} \\ 0 \leq p_k \leq 1 \ \forall k \in \{1, ..., K\} \end{cases} \tag{P2}$$

La classe ω_l est alors écartée si et seulement si la solution est strictement positive. C'est la règle de la maximalité, cela traduit une relation de dominance entre les classes. Une classe en domine une autre si dans toutes les solutions réalisables du polytope défini par les intervalles, elle a une probabilité strictement plus élevée que l'autre. On ne garde donc que les classes non-dominées.

1.2 Exemple

On reprend l'exemple du cours afin d'illustrer le fonctionnement de cette méthode, en faisant une résolutions des PL en utilisant notre code. On peut par ailleurs rejouer cet exemple dans le repértoire du projet.

1.2.1 Le problème

On considère 3 classes d'objets volants :

— Renégat (R) ;
— Endommagé (D) ;
— Provocateur (P).

Un objet volant est définit par sa taille et sa vitesse. On sait *a priori* qu'un objet renégat est plutôt de grande taille et vole à grande vitesse, qu'un objet endommagé est de n'importe quelle taille et qu'il ne vole pas lentement en règle générale, et enfin qu'un appareil provocateur est presque tout le temps de petite taille et vole à basse vitesse.

1.2.2 Schéma de décomposition

Prenons le schéma OVA, on a donc 3 classifieurs à construire :

1. $y \in \{R\}$ vs $y \in \{D, P\}$
2. $y \in \{D\}$ vs $y \in \{R, P\}$
3. $y \in \{P\}$ vs $y \in \{R, D\}$

Maintenant, imaginons que l'on ait entraîné ces classifieurs. Dans cette base de données d'entraînement, on fait l'hypothèse qu'on a plus ou moins le même nombre d'exemples d'appareils provocateurs et endommagés, mais beaucoup moins d'exemples d'appareils renégats. On obtient les intervalles suivants sur un nouvel objet à classifier, qui est de grande taille et vole à vitesse moyenne :

— $\mathcal{C}_1 : \left[\underline{P(y \in \{R\})}, \overline{P(y \in \{R\})}\right] = [0.4, 0.9]$
— $\mathcal{C}_2 : \left[\underline{P(y \in \{D\})}, \overline{P(y \in \{D\})}\right] = [0.5, 0.7]$
— $\mathcal{C}_3 : \left[\underline{P(y \in \{P\})}, \overline{P(y \in \{P\})}\right] = [0, 0.3]$

La phase de *discount* donne $\epsilon_1 = \epsilon_3 = \epsilon_2 = 0$, on a donc des intervalles cohérents entre eux. Enfin, la seule classe éliminée est P car la résolution du PL correspondant à la maximalité donne un optimum strictement positif contre au moins une des deux autres classes (le deux ici). La prédiction est donc $\hat{Y} = \{R, D\}$. Cela semble raisonnablement prudent, étant donné que la seule chose dont on est presque sûr est que les appareils provocateurs sont de petite taille et que la vitesse et la taille pourrait correspondre à R ou D. De plus on on observe un intervalle large pour R, ce qui traduit le peu d'information disponible pour cette classe.

Chapitre 2

Expérimentations

2.1 Jeux de données synthétiques

Dans cette section, nous illustrons la mise en oeuvre du classifieur décrit sur des jeux de données synthétiques représentables graphiquement. Nous nous sommes efforcés de choisir une méthode de génération aussi paramétrable que possible pour représenter une variété de situation, et pour certaines desquelles il serait pertinent de faire de la classification imprécise.

2.1.1 Génération

Quand nous nous sommes posé la question de la génération de données, nous avons dans un premier temps essayé de paramétrer à la main des gaussiennes à 2 dimensions. Nous nous sommes rapidement rendu compte qu'il serait plus intéressant d'implémenter une méthode automatique. Après un peu de recherche, en lisant un autre article [5] proposé pour le projet, nous avons trouvé exactement ce que l'on voulait. La méthode choisie prend en entrée :

- p la dimension des points ;
- K le nombre de classes ;
- la forme des populations, sphériques ou elliptiques ;
- s_i $\forall i \in \{1, ..., K\}$ le nombre d'exemples de chaque classe ;
- la variance des populations les unes par rapport aux autres, homoscédastiques ou hétéroscédastiques ;
- q qui peut se voir comme la difficulté du jeu de données (décroissante avec q croissant), à quel point les populations vont se superposer. C'est le paramètre que l'on fera varier pour justifier l'utilisation d'un classifieur imprécis.

On peut facilement représenter les jeux de données obtenus sur deux ou trois dimensions. Dans la suite, on se limitera à deux.

2.1.2 Exemples et résultats

On peut voir sur l'exemple de la figure 2.1 un jeu de données avec 4 classes qui suivent des distributions sphériques homoscédastiques bien séparées et avec un nombre d'exemple important. On peut légitimement se demander si il est pertinent de faire de la classification

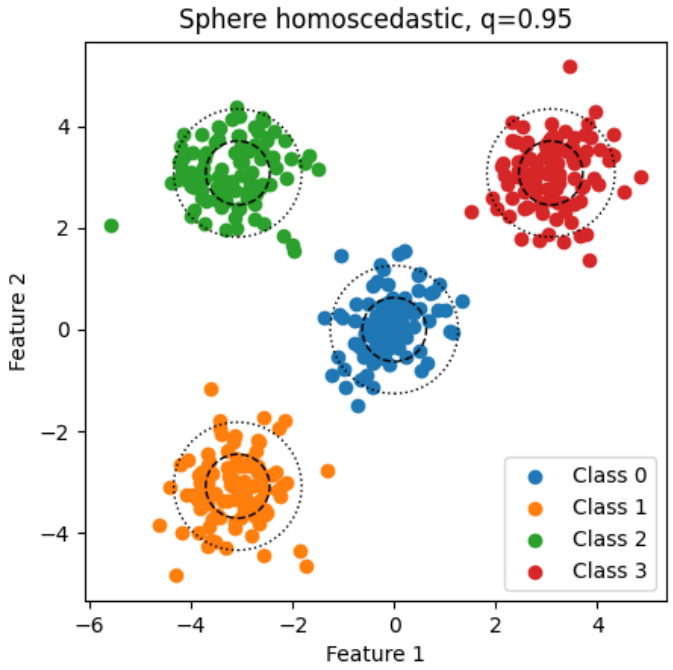

FIGURE 2.1 – Jeu de données synthétique facile

prudente sur ce genre de données. Néanmoins, nous pouvons introduire les critères que nous utilisons pour comparer les différents jeux de données sur cet exemple. Les résultats sont résumés dans la figure 2.2. Pour chaque schéma on mesure :

— La *utility discounted accuracy* qui donne $\frac{\alpha}{|\hat{Y}|} - \frac{\alpha-1}{|\hat{Y}|^2}$ si la classe est dans $\mathcal{P}$ et 0 sinon. On prend u65 et u80 qui sont les deux mesures communément choisies, u65 donnant une récompense modérée à la prudence et u80 une forte récompense ;

— La *weak accuracy* qui vaut $\frac{1}{|\hat{Y}|}$ si la classe est dans $\mathcal{P}$ et 0 sinon ;

— La *set accuracy* qui vaut 1 si la classe est dans $\hat{Y}$ et 0 sinon.

Sur les graphiques, la ligne pointillée indique le score obtenu en prédisant l'ensemble de toutes les classes. C'est en quelque sorte l'équivalent du 0.5 en classification précise. De plus, on peut comparer les tailles des ensembles prédits ainsi que le temps de calcul si nécessaire.

On peut étudier un exemple plus approprié avec :

— 4 classes ;

— des populations sphériques ;

— 30 exemples dont 20 serviront à l'entraînement et 10 à la validation ;

— des variances hétéroscédastiques.

Dans un second temps, on prend des distributions elliptiques et hétéroscédastiques, on met q à 0.8. On obtient le jeu plus difficile de la figure 2.3 et les résultats correspondants sur la figure 2.4. On observe immédiatement une perte de précision sur tout les schémas, OVA et OVO donnant des performances similaires, tandis que les ECOC générés aléatoirement peinent à dépasser les seuils en pointillés. Pour OVA et OVO, ce sont des résultats attendus, pour sparse et dense nous ne comprenons pas tout à fait pourquoi la chute en

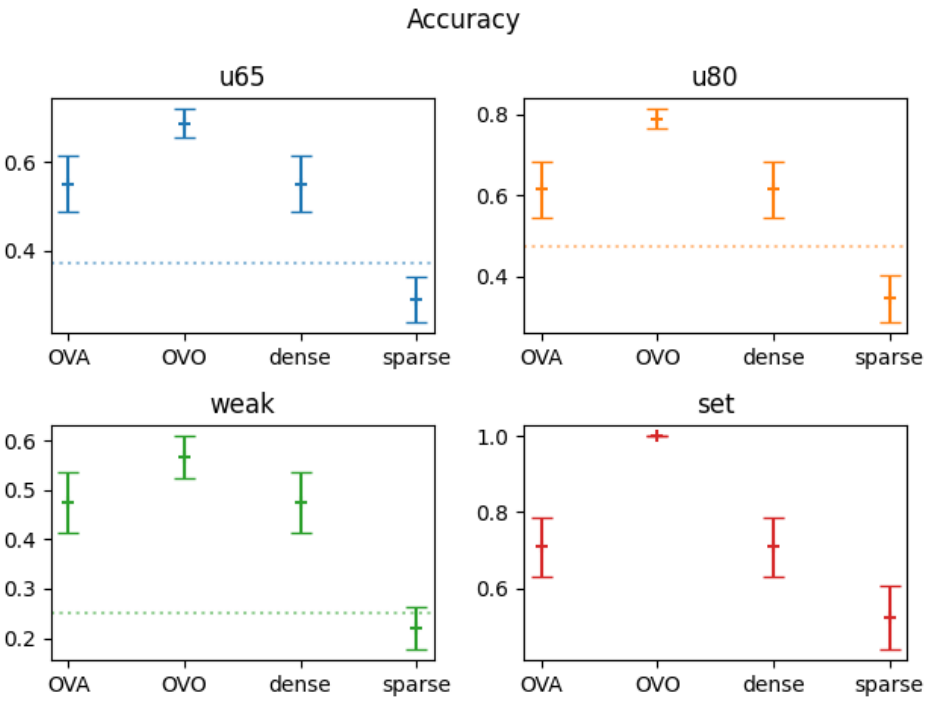

FIGURE 2.2 – Taux de précision pour 2.1

performance est si forte. On peut regarder les tailles des ensembles prédits sur la figure 2.5. Il semblerait que les ECOC aléatoire soient moins prudents que OVA et OVO.

Enfin, on peut aller encore plus loin et mettre q à 0.6 un cas où les données sont très proches et les distributions peu différentiables. On a doublé le nombre de points pour mieux visualiser la difficulté. On s'attend à avoir un grand nombre de points qui pourraient appartenir à plusieurs classes, et donc l'usage de prédiction prudente est pertinente. On obtient le jeu de données de la figure 2.6 avec les mesures de précision en 2.7, de taille en 2.8 et les temps de calcul en 2.9. On remarque que OVA, OVO et dense ont tendance à prédire 2 classes, et en regardant la visualisation du jeu de données on voit plusieurs zones d'intersections entre exactement deux classes, c'est donc probablement dans ces cas que la prédiction est prudente. On voit également des performances plus que satisfaisante en terme de précision. On peut en conclure que l'approche est pertinente dans ce genre de situation, où les classes peuvent se confondre.

2.2 Jeu de données réel

Dans cette section nous essayons de transposer nos résultats sur les données artificielles à un jeu de données réel. Nous avons choisi la base *Optical Recognition of Handwritten Digits* [6]. Nous pensons pertinent d'essayer de faire de la classification imprécise sur cette base car l'écriture de certains chiffres dépendent fortement de la personne. On se retrouve donc avec des cas où plusieurs chiffres seraient possible et si on demandait à quelqu'un de les classer il aurait lui même du mal. Il y a donc un fort intérêt à prédire un ensemble de classes.

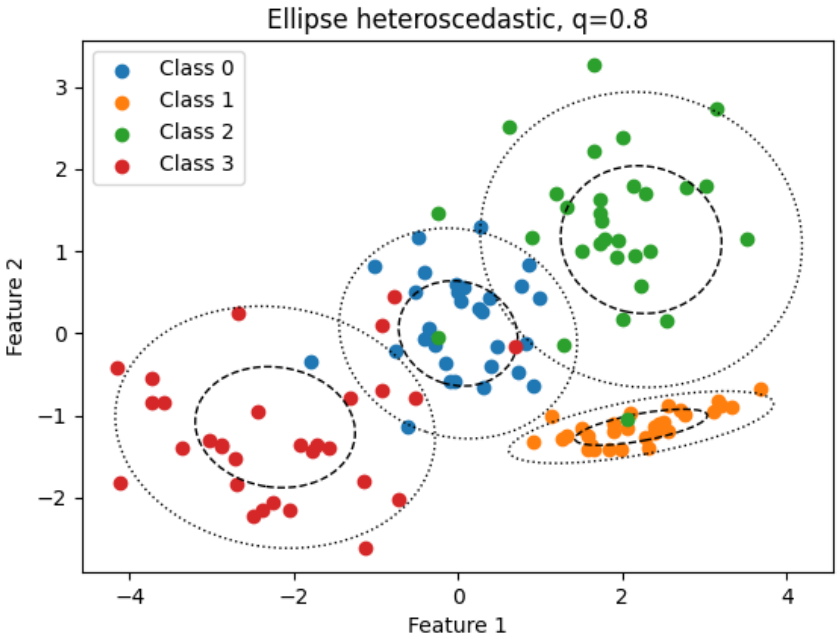

FIGURE 2.3 – Deuxième jeu synthétique

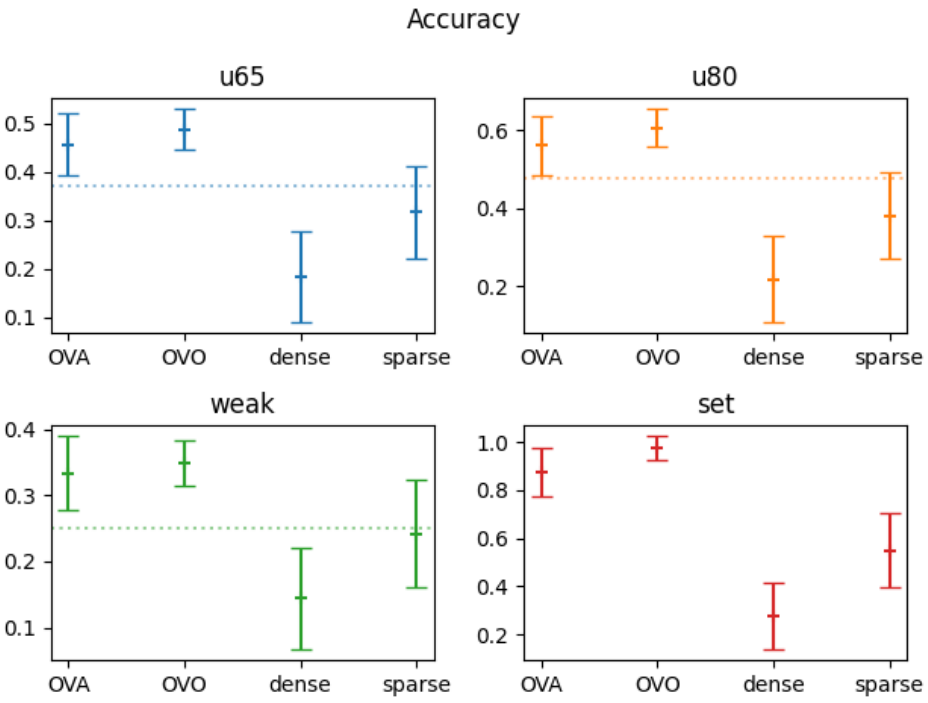

FIGURE 2.4 – Taux de précision pour 2.3

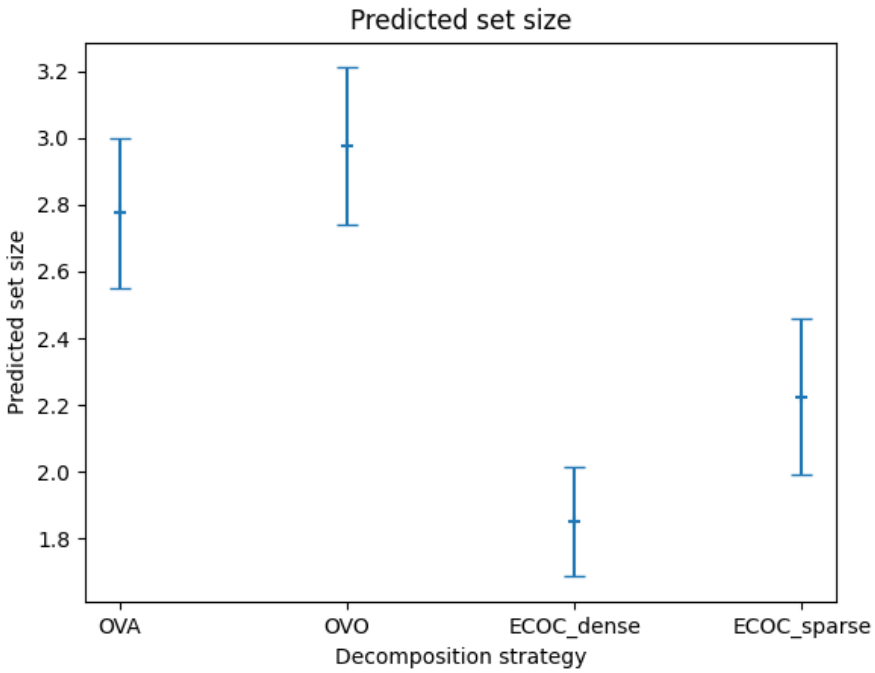

FIGURE 2.5 – Taille de l'ensemble prédit pour 2.3

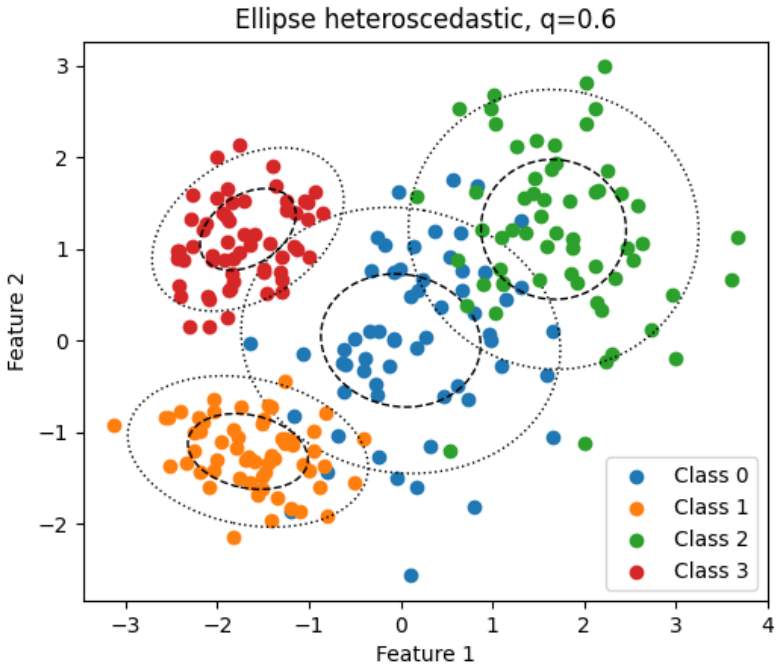

FIGURE 2.6 – Troisième jeu synthétique

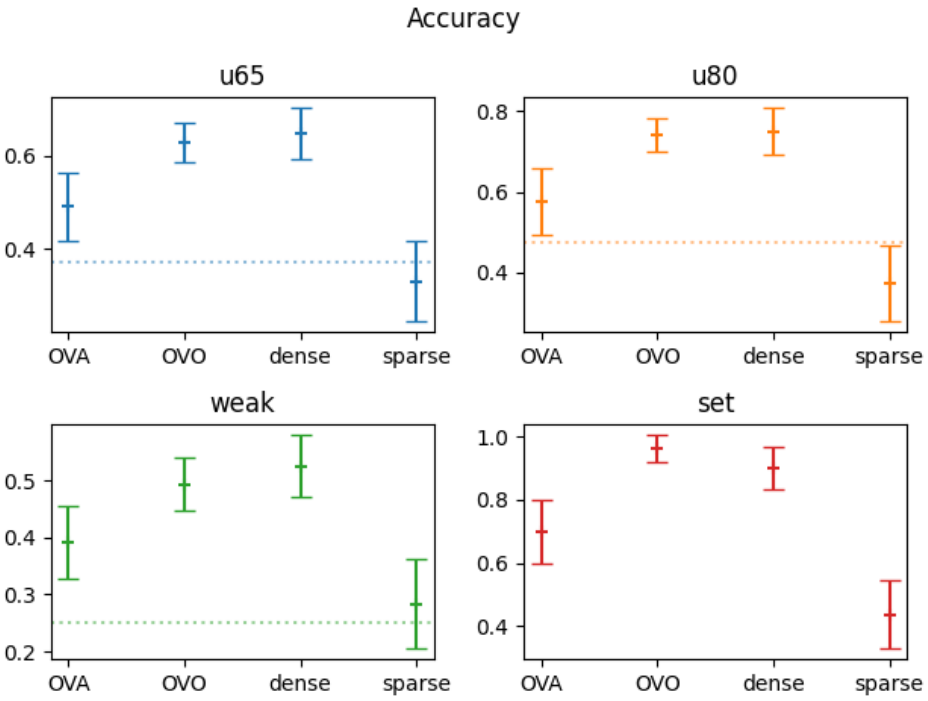

FIGURE 2.7 – Taux de précision pour 2.6

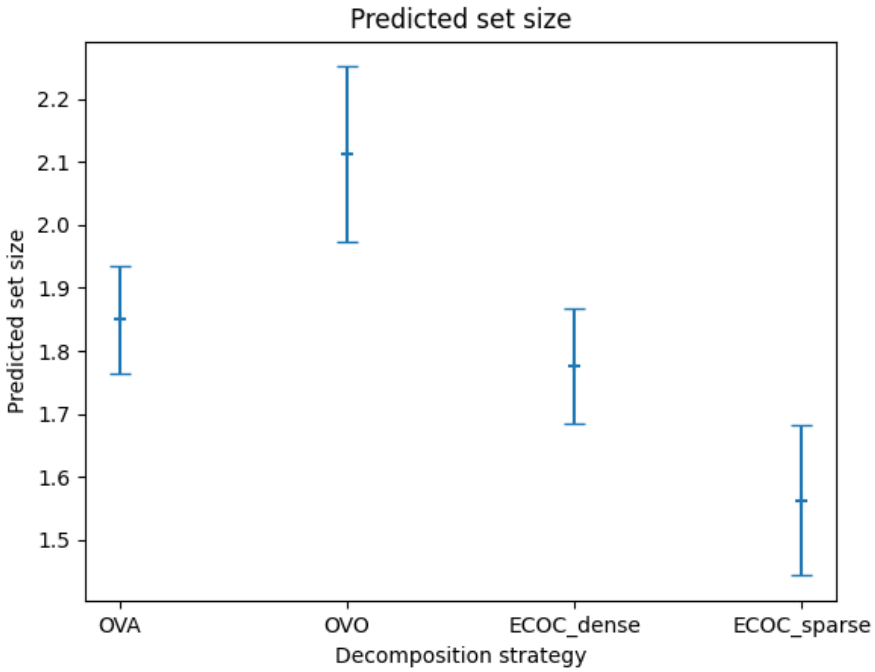

FIGURE 2.8 – Taille de l'ensemble prédit pour 2.6

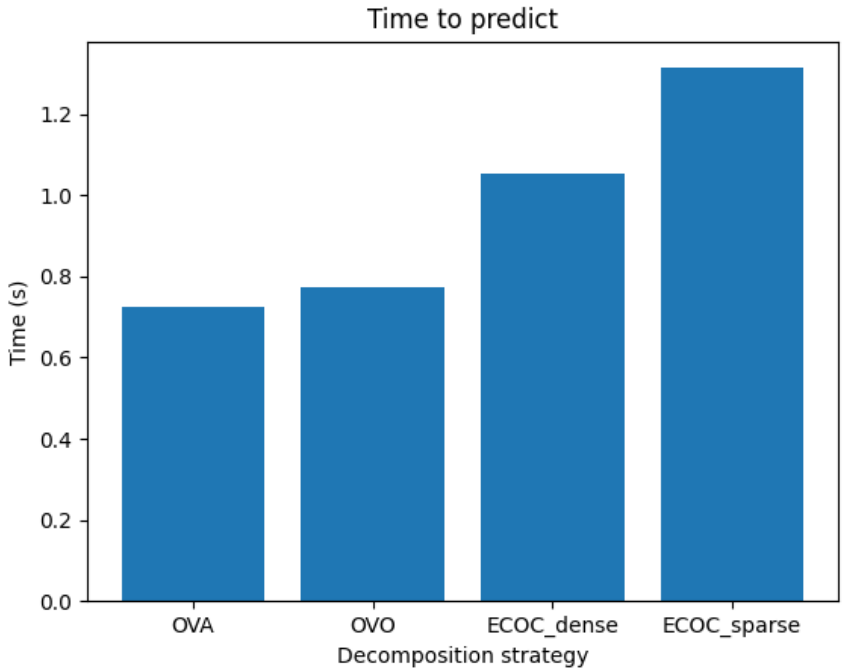

FIGURE 2.9 – Temps d'exécution pour 2.6

2.2.1 Présentation de la base de donnée ORHD

Les caractéristiques du jeu de données ORHD sont de type entier, comprenant un total de 5620 instances et 64 attributs, avec 10 classes. Des images normalisées de chiffres manuscrits ont été extraites en utilisant des programmes de prétaitement de NIST. Sur un total de 43 personnes, 30 ont contribué à l'ensemble d'entraînement et 13 autres à l'ensemble de test. Les images de 32x32 pixels sont divisées en blocs sans chevauchements de 4x4 et le nombre de pixels activés est compté dans chaque bloc. Cela génère une matrice d'entrée de 8x8 où chaque élément est un entier dans la plage de 0 à 16.

2.2.2 Résultats sur ORHD

Nous avons testé les 4 schémas de décomposition pour le jeu de données ORHD. Les résultats sont présentés sur la figure 2.10 pour la précision. D'emblée on peut dire qu'ils ne correspondent pas à nos attentes, et ne confirment pas les résultats obtenus sur les jeux synthétiques. On observe des performances inintéressantes pour OVA, dense et sparse, alors que dans l'article on a plutôt OVA qui domine. Seul OVO obtient des résultats satisfaisants, même très bons. Ici la taille de l'ensemble, sur la figure 2.11, n'explique pas ces différences étant donné que tous prédisent entre 3 et 4 classes, plus entre 2 et 3 pour sparse. En regardant la set accuracy, on voit que seul OVO arrive de manière consistante à inclure la bonne classe. Enfin, les temps d'exécutions sur 2.12 sont plutôt longs et il est à noter que notre implémentation n'est pas parfaite puisque sur certains points de l'ensemble de test, la phase de discounting échoue et ne permet pas de trouver une solution. Cependant, ce n'est arrivé que dans le cas sparse.

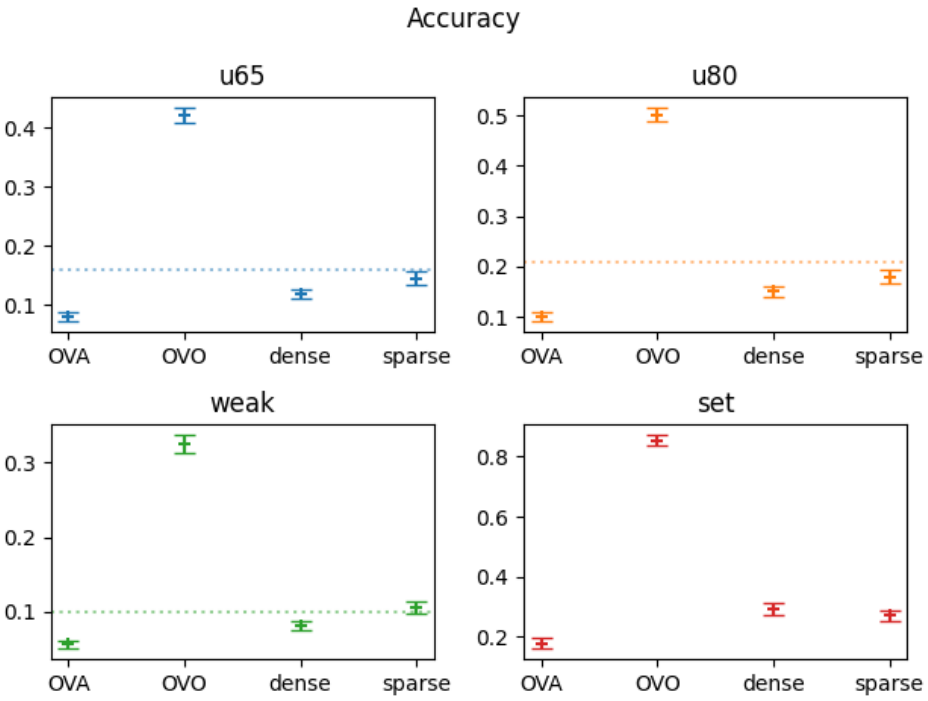

FIGURE 2.10 – Taux de précision pour ORHD

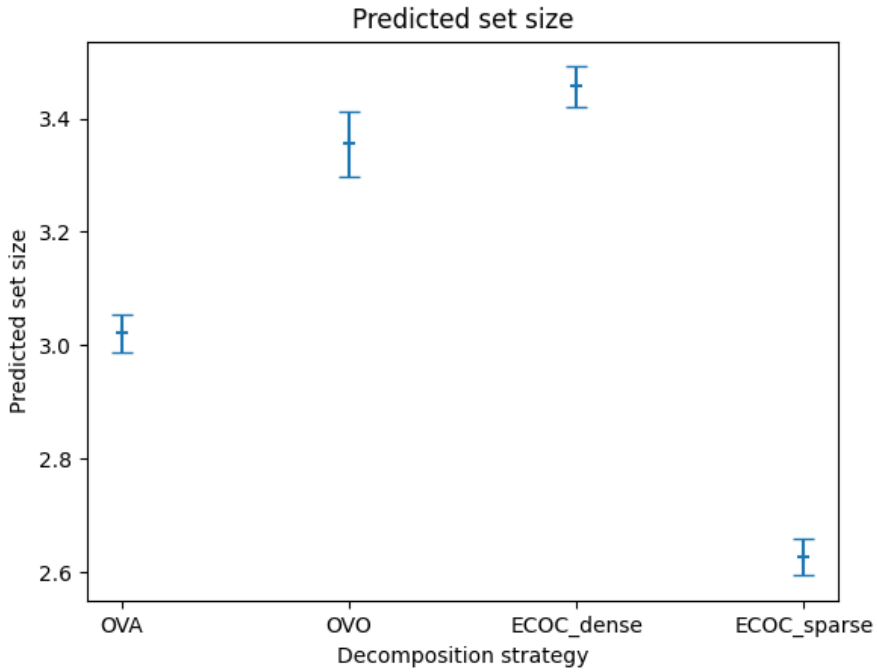

FIGURE 2.11 – Taille de l'ensemble prédit pour ORHD

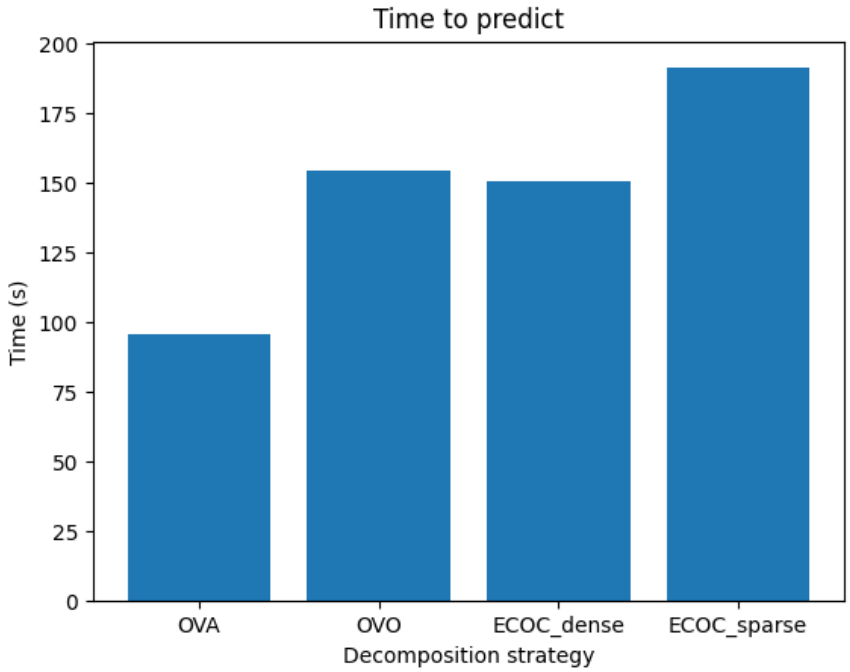

FIGURE 2.12 – Temps d'exécution pour ORHD

2.3 Conclusions

Ces expérimentations nous ont permis d'implémenter une méthode de génération des jeux de données intéressante et très paramétrable. Nous avons obtenus des résultats encourageant sur ces jeux artificiels, ce qui dans un premier temps nous a permis de valider le modèle dans les cas où la classification prudente est pertinente. Malgré cela nous nous sommes heurtés à des résultats inattendus et décevants sur le jeu de données réel. Le code pour rejouer les expériences est trouvable sur le répertoire en ligne https://github.com/b0gz1b/Projet-MADI.

Bibliographie

[1] Benjamin Quost et Sébastien Destercke. « Classification by pairwise coupling of imprecise probabilities ». In : *Pattern Recognition* 77 (2018), p. 412-425. issn : 0031-3203. doi : https://doi.org/10.1016/j.patcog.2017.10.019. url : https://www.sciencedirect.com/science/article/pii/S0031320317304247 (page 1).

[2] T. G. Dietterich et G. Bakiri. *Solving Multiclass Learning Problems via Error-Correcting Output Codes*. 1995. arXiv : cs/9501101 [cs.AI] (page 1).

[3] Leo Breiman. *Classification and regression trees*. Routledge, 2017 (page 2).

[4] Christian P Robert et al. *The Bayesian choice : from decision-theoretic foundations to computational implementation*. T. 2. Springer, 2007 (page 2).

[5] Yonatan Carlos Carranza Alarcón et Sébastien Destercke. « Imprecise Gaussian discriminant classification ». In : *Pattern Recognition* 112 (2021), p. 107739. issn : 0031-3203. doi : https://doi.org/10.1016/j.patcog.2020.107739. url : https://www.sciencedirect.com/science/article/pii/S0031320320305422 (page 5).

[6] E. Alpaydin et C. Kaynak. *Optical Recognition of Handwritten Digits*. UCI Machine Learning Repository. DOI : https ://doi.org/10.24432/C50P49. 1998 (page 7).