Projet SDP : classification multicritères et mutliclasses

Jérome Auguste
jerome.auguste@student-cs.fr

Nathan Bruckmann
nathan.bruckmann@student-cs.fr

Ruben Partouche
ruben.partouche@student-cs.fr

1. Introduction

L'objectif de ce projet est l'implémentation et la comparaison de différents modèle de décision multi-critères, avec pour prétexte l'admission d'élèves suivant leurs différentes notes. À partir d'un ensemble d'élève classés manuellement, le système sera capable d'en déduire les règles sous-jacentes et ainsi classer de nouveau élèves. Le nombre de critères ainsi que de catégories sont fixés arbitrairement, et ces derniers seront dans un premier temps ordonnées, puis imbriqué sous une forme "single peak".

2. Modèles

2.1. MR-Sort

Ce modèle, présenté dans [LMP11] est une simplification d'ELECTRE Tri. On va chercher à définir pour chaque critère un poids et une frontière, et la somme des poids des critères supérieurs à leur frontière est comparée à une valeur λ pour déterminer l'admission. Lors de l'apprentissage, on introduit une variables de slack α a maximiser afin de maximiser la séparation des classes, à la manière d'un SVM.

$$\sum_i w_i(s) - \lambda - \sigma_s = 0 \ \forall s \in A^*$$
$$\sum_i w_i(s) - \lambda + \sigma_s = 0 \ \forall s \in R^*$$
$$\alpha = \min_s \sigma_s$$

Le problème linéaire correspondant, pour deux classes, est alors :

$$\max \alpha$$

s.t.

$$\begin{cases} \sum_{i\in N} w_i(s) + \sigma_s + \epsilon = \lambda & \forall s \in R^* \\ \sum_{i\in N} w_i(s) = \lambda + \sigma_s & \forall s \in A^* \\ \alpha \leq \sigma_s & \forall s \in A^* \cup R^* \\ w_i(s) \leq w_i \quad \forall s \in A^* \cup R^*, & \forall i \in N \\ w_i(s) \leq \delta_i(s) \quad \forall s \in A^* \cup R^*, & \forall i \in N \\ w_i(s) \geq \delta_i(s) - 1 + w_i \quad \forall s \in A^* \cup R^*, & \forall i \in N \\ M\delta_i(s) + \epsilon \geq s_i - b_i \quad \forall s \in A^* \cup R^*, & \forall i \in N \\ M(\delta_i(s) - 1) \leq s_i - b_i \quad \forall s \in A^* \cup R^*, & \forall i \in N \\ \sum_{i\in N} w_i = 1, & \lambda \in [0.5, 1] \\ w_i \in [0,1] & \forall i \in N \\ w_i(s) \in [0,1],\ \delta_i(s) \in \{0,1\} & \forall s \in A^* \cup R^*, \forall i \in N \\ \sigma_s \in \mathbb{R} & \forall s \in A^* \cup R^* \\ & \alpha \in \mathbb{R} \end{cases}$$

où R^* est l'ensemble des candidats rejetés de l'ensemble d'apprentissage, A^* celui des candidats admis, et N l'ensemble des critères. On peut facilement étendre ce principe à plus de deux classes en introduisant une nouvelles frontières par classe supplémentaire. On applique alors les mêmes contraintes pour chaque couple de classes (A_h, A_{h+1}) plutôt que pour A^* et R^*.

2.2. Inv-NCS en utilisant SAT

2.2.1 Cas monotone

Cette approche, totalement différente de la précédente et présentée dans [Bel+18], transforme le problème de classification en problème de satisfaisabilité. En partant des mêmes hypothèses que le papier de référence, on génère de nombreuses variables booléennes à partir des données d'entraînement qui modéliseront notre problème:

- $x_{i,h,k}$ pour tout critère $i \in \mathcal{N}$, toute classe $1 \leq h \leq p-1$ (uniquement pour les frontières intérieures) et toute alternative $k \in \mathbb{X}^*$.
- y_B pour toute coalition de critères (matières) $B \subseteq \mathcal{N}$

A partir de ces variables, nous définissions des règles de monotonie entre les valeurs, entre les classes (sous

l'hypothèse d'une hiérarchie des préférences sur les classes) et entre les coalitions de critères (avec l'hypothèse d'une unique coalition de critères pour l'évaluation par rapport à toutes les classes); ainsi que des règles sur la mauvaise classification des alternatives:

$$\forall i \in \mathcal{N}, \forall 1 \leq h \leq p-1, \forall k < k', x_{i,h,k} \Rightarrow x_{i,h,k'}$$

$$\forall i \in \mathcal{N}, \forall 1 \leq h < h' \leq p-1, \forall k, x_{i,h',k} \Rightarrow x_{i,h,k}$$

$$\forall B \subset B' \subseteq \mathcal{N}, y_B \Rightarrow y_{B'}$$

$$\forall B \subseteq \mathcal{N}, \forall 1 \leq h \leq p-1 \forall u \in X^* : A(u) = C^{h-1},$$

$$\bigwedge_{i \in B} x_{i,h,u_i} \Rightarrow \neg y_B$$

$$\forall B \subseteq \mathcal{N}, \forall 1 \leq h \leq p-1 \forall a \in X^* : A(a) = C^{h},$$

$$\bigwedge_{i \in B} \neg x_{i,h,a_i} \Rightarrow y_{\mathcal{N} \setminus B}$$

Les trois premières équations représentent respectivement la monotonie des valeurs, des classes et des coalitions et les deux dernières représentent les conséquences d'un problème de classification sur une alternative.

En utilisant un moteur SAT (gophersat), nous pouvons donc déduire l'appartenance de chaque alternative à une classe (sachant qu'une alternative appartient à toutes les classes inférieures par hypothèse)

2.2.2 Cas du Single Peak

Dans le cadre de classification multicritère, nous sommes parfois confrontés à des classifications non monotones (taux de glycémie par exemple) et devons donc traiter un cas plus général qu'est le Single Peak. Ce problème pose que les frontières entre les classes ne sont plus monotones mais dans un certain intervalle. Nous supposons alors, par la hiérarchie entre les classes, que les frontières des classes inférieures sont plus "écartées" que les frontières des classes supérieures. Ainsi, pour une frontière de classe h sur le critère i de type $[b_i^{h-}, b_i^{h+}]$, nous avons nécessairement $b_i^{h-1-} <= b_i^{h-}$ et $b_i^{h+} <= b_i^{h-1+}$ (permettant ainsi d'inclure le cas monotone). Afin de traduire la règle de monotonie sur les valeurs, nous avons changé la première règle en :

$$\forall i \in \mathcal{N}, \forall 1 \leq h \leq p-1, \forall k < k' < k'',$$

$$x_{i,h,k} \wedge x_{i,h,k''} \Rightarrow x_{i,h,k'}$$

2.3. Inv-NCS en utilisant MaxSAT sur Single-Peak

Fréquemment, il arrive que le modèle en formulation SAT n'aie pas de solution. Le solveur renvoie alors que le problème n'a pas de solution sans plus d'explications. Or, il serait sûrement possible, en enlevant un certain nombre de clauses, c'est a dire en diminuant le nombre de contraintes sur notre modèle, d'obtenir un modèle qui satisfait partiellement nos clauses. La question naturelle qui vient derrière est de savoir quelle proportion du learning set peut être représenté sous cette solution.

En nous référant a [Tli+22], nous avons adapté les clauses en ajoutant un nouveau booléen z_x qui signale si une alternative $x \in X^*$ est bien classifiée par le modèle ou non. Les clauses (4) et (5) sont donc modifiées ainsi:

$$\forall B \subseteq \mathcal{N}, \forall 1 \leq h \leq p-1 \forall u \in X^* : A(u) = C^{h-1},$$
$$\bigwedge_{i \in B} x_{i,h,u_i} \Rightarrow \neg y_B \vee z_u (4')$$
$$\forall B \subseteq \mathcal{N}, \forall 1 \leq h \leq p-1 \forall a \in X^* : A(a) = C^{h},$$
$$\bigwedge_{i \in B} \neg x_{i,h,a_i} \Rightarrow y_{\mathcal{N} \setminus B} \vee z_a (5')$$

Cette modification permet de relaxer sur les clauses mal classifiées : la clause peut toujours être vraie. Nous avons aussi du ajouter une clause qui permet de maximiser le nombre d'alternatives bien classifiées, pour éviter que le modèle relaxe trop de clause ce qui affecterait négativement les performances :

$$\forall x \in X^*, z_x$$

Il pourrait être intéressant d'observer le nombre de clauses relaxées en fonction de différents paramètres pour notre solveur MaxSAT, comme le bruit et la taille du learning set.

3. Génération des données

3.1. Modélisation pour les problèmes MR-Sort et NCS

Pour tester notre implémentation de la résolution des problèmes, nous avons créé une classe utilitaire pour la génération et la lecture de données factices. Tous les paramètres du problème sont générés aléatoirement, selon les suggestions faites dans [LMP11], au paragraphe 4 : Empirical design and result. Aux notations précédentes, on ajoute le nombre de classes de notre problème $N_c \geq 2$.

$$\lambda \sim \mathcal{U}(0.5, 1)$$
$$\forall i \in N, w_i \sim \mathcal{N}(\mu = 2, \sigma^2 = 1)$$
$$\forall b_{ij}, i \in N, j \in N_c^*, b_{ij} \sim \mathcal{U}(b_{ij-1}, j/N_c)$$
$$\forall b_{i0}, i \in N, b_{i0} \sim \mathcal{U}(0, 1/N_c)$$

On normalise le vecteur de poids généré. Tirer les poids selon une loi normale a permis d'éviter d'obtenir des poids trop différents. On génère ensuite une matrice (nombre d'exemples $\times$ nombre de critères), en tirant dans une loi uniforme, puis on affecte les "étudiants" dans la (les) classes en comparant leur vecteur de notes à la (aux) frontière(s).

3.2. Modélisation pour le problème NCS non monotone

Pour le problème non monotone, nous avons modifié le tirage des frontières. Pour certains critères j, les frontières sont de dimension 2 et contiennent une borne inférieure et une borne supérieure, et il faut se trouver entre les deux bornes pour valider le critère. Puisque le problème où il faut se trouver à l'extérieur des bornes pour valider le critère est identique, à une transformation linéaire près, nous avons ignoré ce cas et généré nos données toujours pour le cas d'un "pic".

$$\forall b_{ij}, i \in N, j \in N_c^*, b_{ij} \sim \mathcal{U}(b_{ij-1,0}, b_{ij-1,1})$$
$$\forall b_{i0}, i \in N, b_{i0} \sim \mathcal{U}(0, 1)$$

Juste après avoir généré les vecteurs de taille 2, on les trie ensuite pour éviter d'avoir des frontières inversées.

3.3. Méthode d'entraînement et de mesure

Pour compartimenter nos expériences, et obtenir des mesures de performances fiables dans des "conditions réelles", nous enregistrons nos données, puis nous séparons systématiquement notre dataset en un train set et un test set qui permet d'avoir une mesure de performance significative et non aléatoire d'un entraînement a l'autre.

Nous avons également ajouté du bruit à nos labels sur le train set en sélectionnant aléatoirement un certain pourcentage (5% dans le cadre de nos expériences) de données, et en leur affectant aléatoirement une classe. Cela nous a permis de tester la robustesse des méthodes face au bruit qui peut être présent dans les données.

3.4. Problème d'équilibrage de classes

Le tirage dans une loi uniforme, à la fois des données et des frontières, crée fréquemment un déséquilibre des classes "problématique", a savoir qu'il existe parfois des classes presque vides. Sans être une difficulté pour la résolution des problèmes, cela rend parfois la résolution triviale puisqu'il suffit d'affecter tous les étudiants a une seule classe pour obtenir des modèles très performants. Nous avons corrigé ce défaut en forçant le générateur à retirer aléatoirement des matrices jusqu'à ce que le déséquilibre de classe ne soit pas trop important, en comparant l'écart entre le pourcentage de données d'une classe et le pourcentage que les classes devraient avoir si elles étaient parfaitement équilibrées ($1/N_c$).

4. Résultats

4.1. Conditions expérimentales

Les résultats suivant sont obtenues en moyennant sur 5 essais différents, avec un nombre de critères variant de 3 à 6, et un nombre de classes **ordonnées** compris entre 2 et 4. Sauf mention contraire, les données d'apprentissages ne sont pas bruitées.

4.2. Temps de calcul

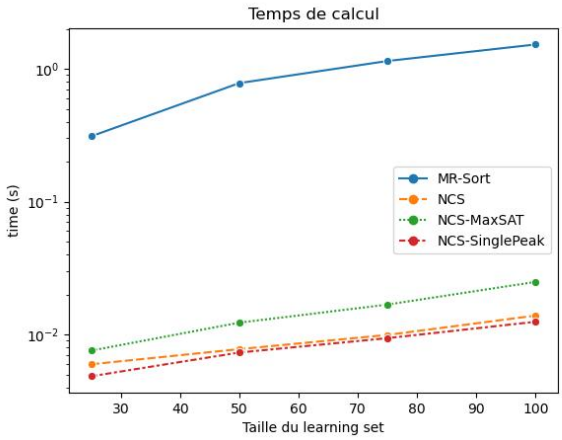

Figure 1. Temps de calcul en fonction de la taille du learning set

Pour les deux types de modèles, le temps de calcul croit de manière exponentielle avec la taille du learning set ainsi que le nombre de critères. Et pour ces 2 paramètres, le temps de calcul des modèles NCS, basés sur les solveurs SAT, sont presque 100 fois inférieurs à ceux du modèle MR-Sort qui repose sur de la programmation linéaire. C'est là tout l'avantage du NCS sur MR-Sort, qui offre des temps de calcul prohibitif lorsque le nombre de paramètres augmente. Les modèles NCS et NCS-SinglePeak sont virtuellement identiques et fonctionnent sur le même principe, même si le second permet de prendre en compte aussi les données en *single peak*. En revanche, MaxSAT est un peu plus lent, car il doit trouver le nombre maximum de clauses permettant la convergence.

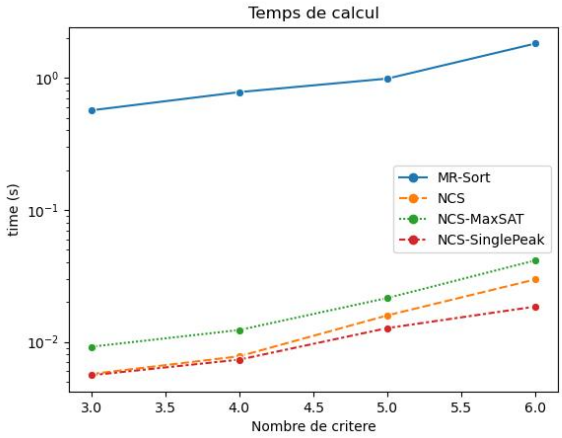

Figure 2. Temps de calcul en fonction du nombre de critères

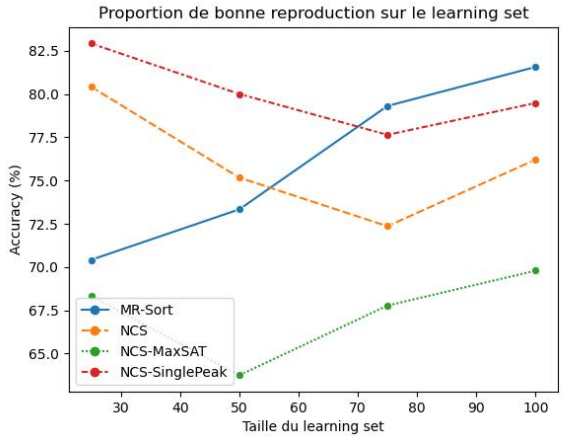

Figure 3. Reproduction du learning set

4.3. Précision

MR-Sort est le modèle le plus performant vis-à-vis de la précision. Étonnamment, MaxSAT ne parvient pas à surpasser les modèles SAT (NCS) classiques. Si on exclut un problème d'implantation, la piste la plus probable de ce phénomène serait qu'en cas d'incompatibilité, les modèles SAT prédisent la classe zéro, généralement la plus représentée, et aurait donc des performances injustement élevée. NCS et et NCS-SinglePeak sont censés être équivalent, et offrent des performances similaires.

De manière générale, plus le learning set est grand, meilleure est généralisation pour tout les modèles. La précision sur le test set augmente ainsi en fonction de la taille du learning set. En revanche seul MR-Sort et le modèle basé sur MaxSAT suivent la même tendance lors de la reproduction du learning set. En effet, les données

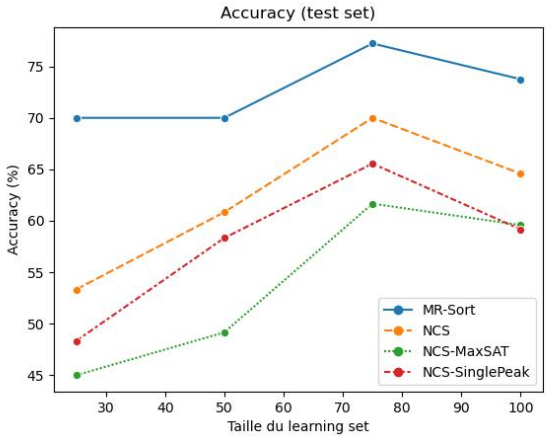

Figure 4. Généralisation sur un test set

sont générés selon la formulation proposé par [LMP11] pour MR-Sort, et donnent parfois des modèles non satisfaisable par NCS. Cette probabilité augmente avec la taille du dataset, d'où la perte en performance. MaxSAT permet d'éliminer les clauses non satisfaisables et ne souffre donc pas de ce problème.

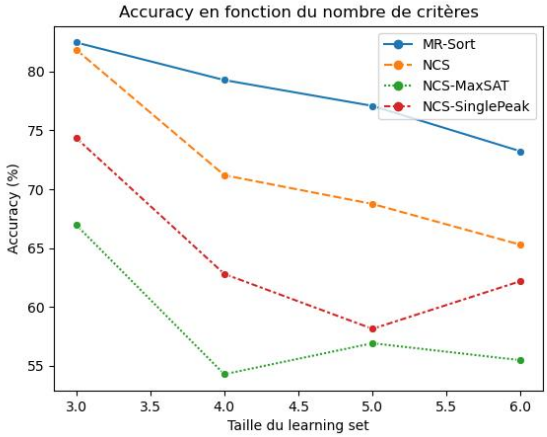

Figure 5. Nombre de critères

Lorsque le nombre de critères ou de catégories augmentent, le problème devient plus complexe. La précision diminue alors, et de manière similaires sur tout les modèles.

La présence de bruit diminue les performances des modèles. NCS y est le plus sensible, car le bruit rend rapidement en modèle SAT non satisfaisable. AU contraire, MR-Sort et MaxSAT offre toujours un modèles satisfaisable. MaxSAT s'en sort ainsi remarquablement mieux que NCS, et offre de meilleurs performances dès 5% de données bruitées.

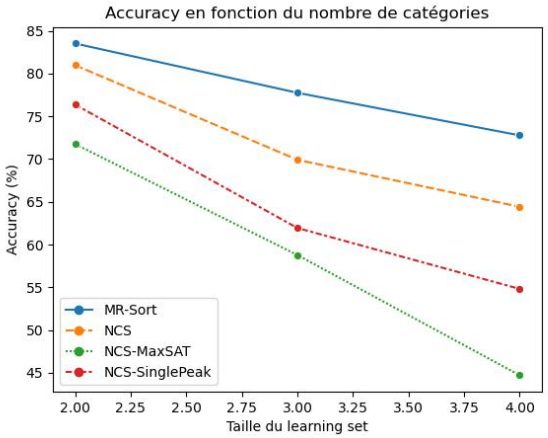

Figure 6. Nombre de catégories

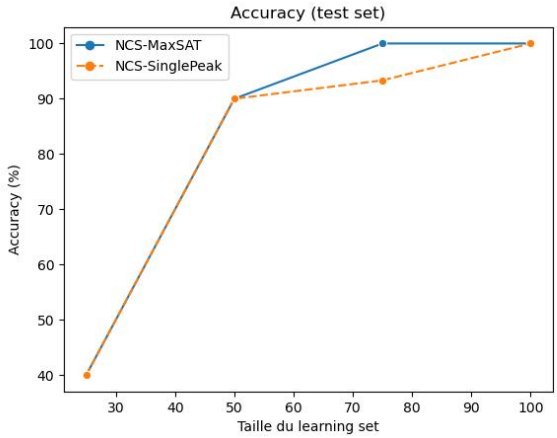

Figure 8. Performance sur la formulation *Single Peak*

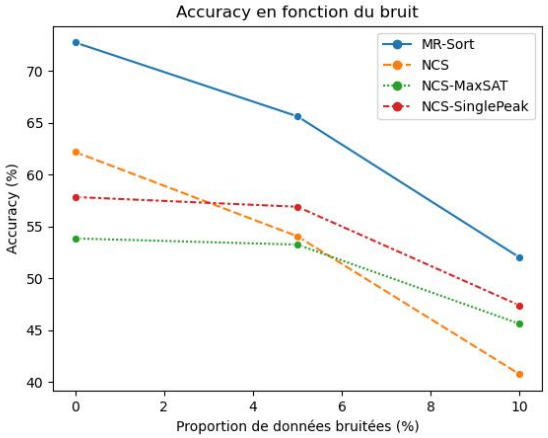

Figure 7. Résistance aux données bruitées

4.4. Formulation Single-Peak

On observe les même tendances sur la formulation *Single Peak* pour les modèles compatibles, les performances grimpent avec la taille du learning set, et sont par ailleurs excellente, atteignant les 100% pour un learning set de 75 éléments (learning set non trivial avec différentes classes bien présentes).

Conclusion

En terme de précision pure, MR-Sort reste la meilleure alternative dans tous les cas, pour des données bruitées, ou des datasets de grande taille, si l'on souhaite obtenir les meilleurs résultats de classification. En revanche, les performances en terme de temps de calcul sont bien moins bonnes et augmentent rapidement avec la taille du dataset, ou le nombre de critères. Ainsi, si on recherche un compromis entre performance et temps de calcul, la représentation du problème en terme de satisfaisabilité redevient intéressante, car on gagne 2 ordres de grandeur en temps de calcul, pour des performances qui peuvent être dégradées de 10 a 35% selon les conditions expérimentales . Par ailleurs, la flexibilité de la représentation du problème SAT nous permet de tester des variantes (SinglePeak, MaxSAT) qui sont plus complexes a implémenter sous forme de problème d'optimisation, et qui pourraient avoir des performances en temps de calcul encore dégradées (notamment pour le problème MaxSAT).

Pour poursuivre ce travail, il serait donc intéressant de comparer les performances en temps de calcul et en précision des problèmes SinglePeak et MaxSAT sous la forme d'un problème d'optimisation linéaire comme MR-Sort, afin de voir si on arrive toujours a obtenir de meilleurs performances que dans le problème SAT pour un temps de calcul qui reste raisonnable.

References

[LMP11] Agnès Leroy, Vincent Mousseau, and Marc Pirlot. "Learning the Parameters of a Multiple Criteria Sorting Method". In: Oct. 2011, pp. 219–233. ISBN: 978-3-642-24872-6. DOI: 10.1007/978-3-642-24873-3_17.

[Bel+18] K. Belahcène et al. "An efficient SAT formulation for learning multiple criteria non-compensatory sorting rules from examples". In: *Computers Operations Research* 97 (2018), pp. 58–71. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2018.04.019. URL: https://www.sciencedirect.com/science/article/pii/S0305054818301102.

[Tli+22] Ali Tlili et al. “Learning non-compensatory sorting models using efficient SAT/MaxSAT formulations”. In: *European Journal of Operational Research* 298.3 (2022), pp. 979–1006. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2021.08.017. URL: https://www.sciencedirect.com/science/article/pii/S0377221721006858.