Systèmes de décision

CentraleSupélec - 3A

Clustering and modeling customer preferences in a supermarket

2024-02-12

CentraleSupélec

Cindy Hua
Rémi Boutonnier
Romain Ageron

Table des matières

I Introduction

Ce rapport est rédigé dans le cadre du cours de système de décision. Tout le cadre du projet est expliqué dans le document le présentant, nous supposons donc que ce document a été lu avant ce rapport. Comme demandé dans le sujet, ce rapport se concentre dans un premier temps sur un modèle de programmation linéaire (ou Mixed Integer Programming) pour résoudre le problème de manière exacte. Cependant, puisque que cette méthode ne passe pas à l'échelle, nous présentons par la suite notre travail sur des modèles heuristiques donnant une solution approchée en un temps raisonnable, même pour des grosses instances.

II Mixed Integer Programming pour la résolution exacte

Dans cette section, nous reprenons les notations de l'énoncé.

II.1 Modèle pour un nombre quelconque de clusters

Variables

— Valeur pour le cluster k de la fonction marginale pour le critère i au point x_i^l :
$\forall k \in [\![1, K]\!], \forall i \in [\![1, n]\!], (u_{k,i,0} = 0 \wedge \forall l \in [\![1, L]\!], u_{k,i,l} \in \mathbb{R}^+)$

— Erreur lorsque la préférence j n'est pas représentée :
$\forall j \in [\![1, P]\!], \sigma_j \in \mathbb{R}^+$

— Indicatrice de « le cluster k représente la préférence j » (pas besoin du sens réciproque) :
$\forall k \in [\![1, K]\!], \forall j \in [\![1, P]\!], \delta_{k,j} \in \{0, 1\}$

Objectif

Minimiser la somme des termes d'erreur :

$$\sum_{j=1}^{P} \sigma_j$$

Contraintes

1. Pour chaque préférence il existe un cluster qui doit l'expliquer :
   $\forall j \in [\![1, P]\!], \sum_{k=1}^{K} \delta_{k,j} \geqslant 1$
2. Normalisation des fonctions d'utilité :
   $\forall k \in [\![1, K]\!], \sum_{i=1}^{n} u_{k,i,L} = 1$
3. Croissance des fonctions marginales :
   $\forall k \in [\![1, K]\!], \forall i \in [\![1, n]\!], \forall l \in [\![1, L]\!], u_{k,i,l} \geqslant u_{k,i,l-1}$
4. $\delta_{k,j} = 1 \implies x^{(j)} \succcurlyeq_k y^{(j)}$:
   $\forall k \in [\![1, K]\!], \forall j \in [\![1, P]\!], u_k(x^{(j)}) + \sigma_j \geqslant u_k(y^{(j)}) + \epsilon + M(\delta_{k,j} - 1)$
   $M \in \mathbb{R}^+$ est ici une constante majorante pour notre inégalité (en pratique $M = 2$), pour effectuer le couplage entre les variables binaires et la satisfaction d'inégalités.

II.2 Modèle spécifique pour deux clusters

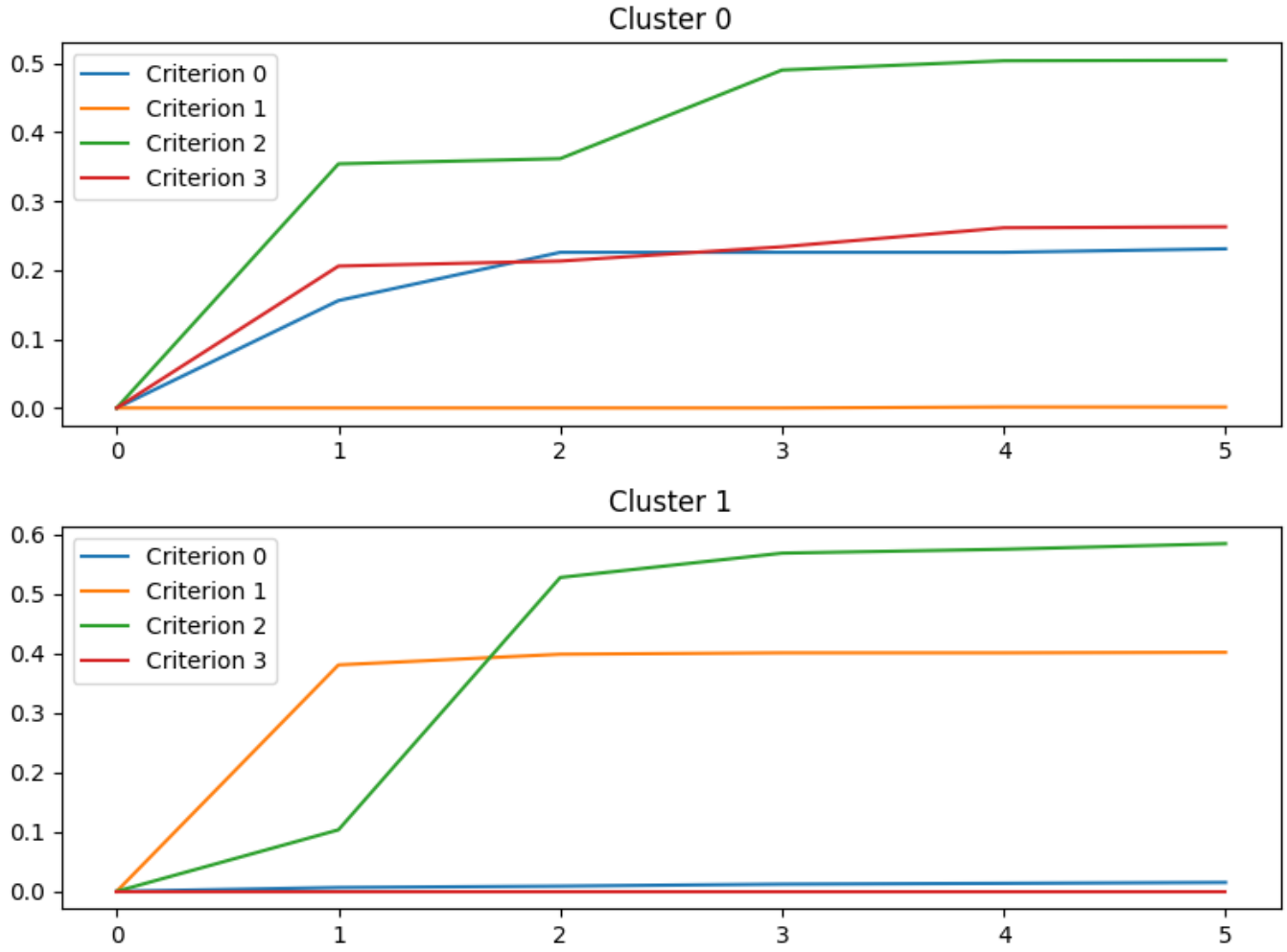
Dans le cas de deux clusters, il est possible de simplifier le modèle précédent ce qui permet de le résoudre rapidement. Cela se fait en utilisant un δ par préférence et en modifiant la contrainte 4 comme suit, les autres variables et contraintes ainsi que la fonction objectif restant inchangées.

4.1. $\delta_j = 0 \implies x^{(j)} \succcurlyeq_1 y^{(j)}$:
$\forall j \in [\![1, P]\!], u_1(x^{(j)}) + \sigma_j \geqslant u_1(y^{(j)}) + \epsilon - M\delta_j$

4.2. $\delta_j = 1 \implies x^{(j)} \succcurlyeq_2 y^{(j)}$:
$\forall j \in [\![1, P]\!], u_2(x^{(j)}) + \sigma_j \geqslant u_2(y^{(j)}) + \epsilon + M(\delta_j - 1)$

II.3 Fonctions de décision pour le cas dataset 4

En résolvant le modèle présenté précédemment sur le dataset 4, nous obtenons les fonctions de décision suivantes :



Le critère 2 est important dans les deux clusters. En revanche, les critères restants sont opposés dans les deux clusters : les critères 0 et 3 sont importants dans le cluster 1 alors que le critère 1 est important dans le cluster 2. Ces fonctions de décision ont donc l'air cohérente pour modéliser les préférences des utilisateurs selon le cluster.

III Modèle heuristique pour le passage à l'échelle

Notre modèle heuristique consiste en la détermination de clusters initiaux puis de manière itérative à répéter les deux étapes suivantes.

— Détermination des clusters

— Apprentissage d'un modèle UTA par cluster

III.1 Détermination des clusters initiaux

L'algorithme itératif est sensible à la qualité des clusters initiaux. Un clustering initial semblable aux « vrais clusters » permet de converger plus rapidement et vers un meilleur résultat qu'un clustering initial peu semblable aux « vrais clusters ». Cette sous-section discute des différentes méthodes étudiées pour la détermination des clusters initiaux.

III.1.1 Cluster initiaux aléatoires

Une première méthode consiste à simplement affecter chaque préférence dans un cluster initial de manière aléatoire. La convergence est lente (toujours de lents progrès après 40 itérations) et n'atteint pas d'aussi bons résultats qu'en initialisant avec des clusters proches des « vrais clusters ». Même si le nombre de paires expliquées atteint les 96%, le Rand index reste aux alentours de 0.66.

III.1.2 Fonction d'utilité initiale aléatoire - répartition uniforme selon les critères

Une autre méthode consiste à tirer de manière uniforme une fonction d'utilité aléatoire par cluster. On peut espérer qu'avoir des clusters initiaux qui dépendent du comportement des préférences vis-à-vis de fonctions d'utilités permettrait d'avoir un clustering initial plus pertinent qu'un clustering aléatoire, mais en pratique cela ne permet pas d'améliorer les résultats.

Note : pour tirer une fonction d'utilité aléatoire de manière uniforme, on commence par tirer les poids w_i de chaque critère à l'aide d'une distribution de Dirichlet symétrique en dimension n et de paramètre L. Puis on tire, pour chaque critère i, $L-1$ nombres aléatoires suivant la loi $\mathcal{U}([0, w_i])$ et on les trie dans l'ordre croissant.

III.1.3 Fonction d'utilité initiale aléatoire - répartition inégale selon les critères

En biaisant la distribution de Dirichlet vers les coins (en prenant le paramètre égal à 1), on obtient aussi des résultats de qualité similaire mais la convergence est légèrement plus rapide. Cela est peut-être du au fait que les fonctions d'utilités tirées ont une répartition des poids non-uniforme et sont donc peut-être plus discriminantes pour les préférences.

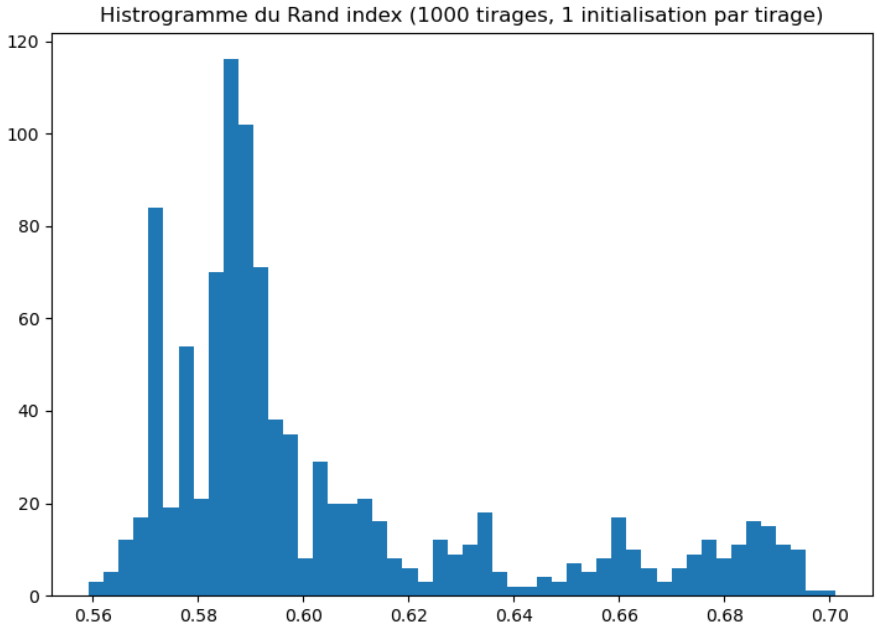
III.1.4 K-Means

Note : D'autres algorithmes de clustering (classiques et fuzzy) ont été essayés mais c'est *K-Means* qui fournit les meilleurs résultats. Dans cette section nous étudions le choix de l'espace euclidien dans lequel on effectue l'algorithme *K-Means*.

III.1.4.1 Approche naïve

Nous considérons chaque préférence $(x^{(j)}, y^{(j)})$ comme un vecteur de $\mathbb{R}^{2n}$ en concaténant les deux vecteurs $x^{(j)}$ et $y^{(j)}$. Nous effectuons alors un clustering *K-Means* sur les P préférences, avec en paramètre K le nombre de clusters recherchés. On utilise l'initialisation avec *K-Means++* puisqu'elle donne les meilleur résultats.

Dans le cas où l'on effectue une seule initialisation de l'algorithme *K-Means*, on obtient un Rand index moyen de 0.605 ± 0.035 (1000 tirages).



Lorsque l'on utilise l'algorithme *K-Means*, il est possible d'effectuer plusieurs initialisation. Dans ce cas, le clustering avec la variance intra-cluster la plus faible est retourné.

Avec 100 initialisations, on obtient un Rand index moyen de 0.5875 ± 0.0005 (100 tirages). Ainsi, on constate qu'une diminution de la variance intra-cluster amène en réalité à un clustering de moins bonne qualité. Non seulement la probabilité d'obtenir un bon clustering est faible, mais la variance intra-cluster (rapide à calculer) n'est pas un bon indicateur de la qualité du clustering, il faut donc apprendre un modèle UTA sur chaque cluster pour vérifier la qualité du clustering ce qui est nettement coûteux en terme de calculs.

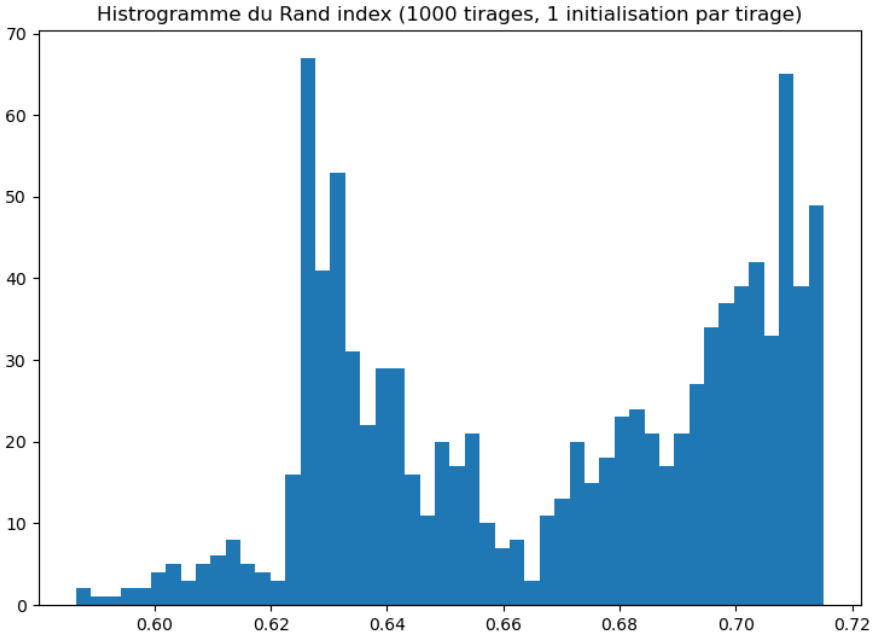
III.1.4.2 Un espace plus proche des fonctions d'utilités

Plutôt que de représenter chaque préférence $(x^{(j)}, y^{(j)})$ en concaténant les deux vecteurs $x^{(j)}$ et $y^{(j)}$, on cherche à représenter les vecteurs des produits dans un espace plus proche des fonctions d'utilité.

Pour $l \in [\![0, L]\!]$, on pose $a_l \in \mathbb{R}^{L+1}$ le vecteur donc la composante l vaut 1 et les autres composantes valent 0. Soit y un vecteur représentant un produit et soit $i \in [\![1, n]\!]$ un critère. Soit $l \in [\![1, L]\!]$ tel que $y_i \in [x_i^{l-1}, x_i^l]$ et posons $\lambda_i = x_i^l - y_i$. On pose alors $b_i = \lambda_i a_{l-1} + (1 - \lambda_i) a_l$. On représente alors y en concaténant les vecteurs b_i, noté $\tau(y)$. En utilisant une telle représentation, l'évaluation d'une fonction d'utilité devient un simple produit scalaire.

Pour représenter une préférence $(x^{(j)}, y^{(j)})$ pour l'algorithme *K-Means*, on utilise alors le meilleur hyperplan séparateur (à la manière d'une SVM) entre $\tau(x^{(j)})$ et $\tau(y^{(j)})$. En pratique, les hyperplans sont représentés par un vecteur normal unitaire dirigé de y vers x et le terme d'ordonnée à l'origine est omis puisqu'il dégrade légèrement les résultats.

Dans le cas où l'on effectue une seule initialisation de l'algorithme *K-Means*, on obtient un Rand index moyen de 0.669 ± 0.033 (1000 tirages).



Avec 100 initialisations de l'algorithme *K-Means*, on obtient un Rand index moyen de 0.7012 ± 0.0015 (100 tirages). Ainsi, dans cet espace, une diminution de la variance intra-cluster est un indicateur assez fiable de la qualité du clustering obtenu.

III.1.4.3 Adaptation du produit scalaire au dataset

Utiliser la distance euclidienne usuelle comme dans la sous-section précédente revient à dire que l'on ne peut pas faire d'hypothèse sur la distribution des préférences. En pratique ce n'est pas le cas, et puisque l'on dispose d'un dataset, on peut utiliser l'erreur empirique sur le dataset. Soit u, v deux fonctions d'utilités. On définit alors une norme euclidienne $\|.\|_S$ où $S \in \mathcal{S}_{n(L+1)}(\mathbb{R})$ est la matrice symétrique définie positive associée au produit scalaire.

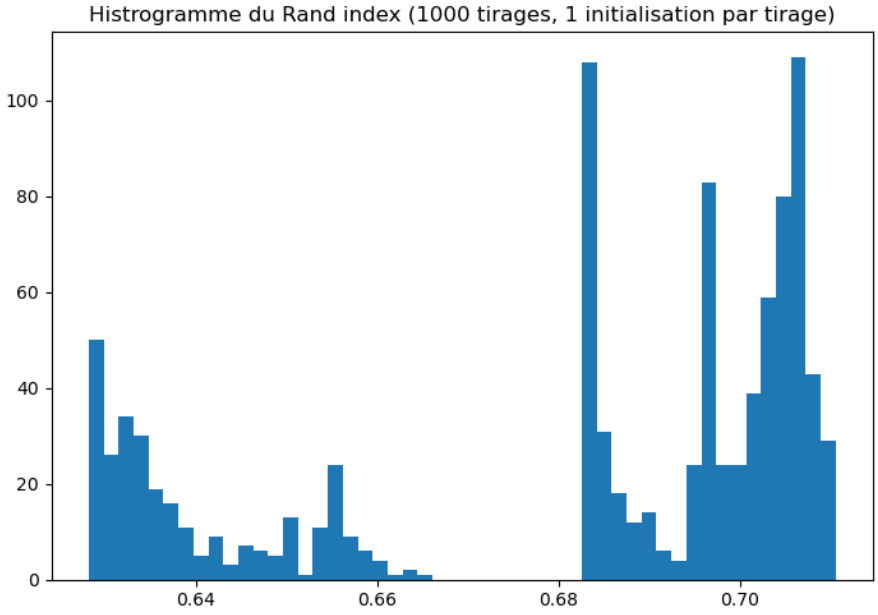
$$\|u - v\|_S^2 = \sum_{j=1}^{P} \left(u\left(x^{(j)}\right) - u\left(y^{(j)}\right) - v\left(x^{(j)}\right) + v\left(y^{(j)}\right) \right)^2$$

On a $u\left(x^{(j)}\right) - u\left(y^{(j)}\right) - v\left(x^{(j)}\right) + v\left(y^{(j)}\right) = \left\langle u - v | \tau\left(x^{(j)}\right) - \tau\left(y^{(j)}\right) \right\rangle$. On en déduit l'expression suivante pour la matrice S.

$$S = \sum_{j=1}^{P} \left(\tau\left(x^{(j)}\right) - \tau\left(y^{(j)}\right) \right) \otimes \left(\tau\left(x^{(j)}\right) - \tau\left(y^{(j)}\right) \right)$$

Note : S est en fait seulement une matrice symétrique semi-définie positive et non pas définie positive puisqu'elle possède n valeurs propres nulles. En effet, on remarque que pour chaque critère i, la partie p_i correspondant au critère i dans $\tau\left(x^{(j)}\right) - \tau\left(y^{(j)}\right)$ est dans le noyau de la matrice $M = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} \in \mathcal{M}_{1,L}(\mathbb{R})$. En pratique,les p_i ré-exprimées dans une base orthonormale du noyau de M (obtenue en prenant les L derniers vecteurs singuliers droits de M) avant de calculer la matrice $S \in \mathcal{S}_{nL}(\mathbb{R})$. Étant donné que l'algorithme *K-Means* de *Scikit-Learn* utilise uniquement la distance euclidienne usuelle, il faut effectuer un changement de base pour pouvoir utiliser le produit scalaire associé à S. Cela se fait en multipliant la racine carrée matricielle de S (S est semi-définie positive donc la racine carrée existe et est unique).

Dans le cas où l'on effectue une seule initialisation de l'algorithme *K-Means*, on obtient un Rand index moyen de 0.682 ± 0.028 (1000 tirages).



Avec 100 initialisations de l'algorithme *K-Means*, on obtient un Rand index moyen de 0.7033 ± 0.0011 (100 tirages). Dans ce cas aussi, une diminution de la variance intra-cluster est un indicateur assez fiable de la qualité du clustering obtenu.

III.2 Algorithme itératif

III.2.1 Algorithmes d'itération sur le clustering existant

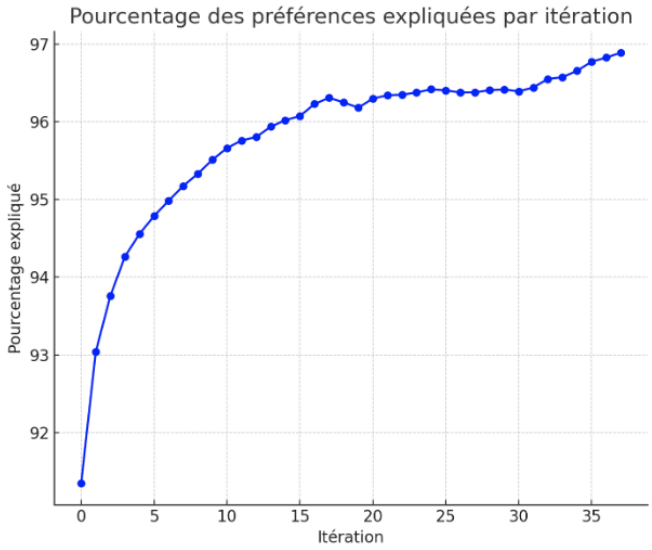
Inclusion de la similarité des paires dans la distance Une première idée serait d'ajouter à la distance euclidienne initiale les similarités entre chaque préférence. Notre distance L2 serait alors composées pour chaque paire j, des $K+2$ ordonnées :

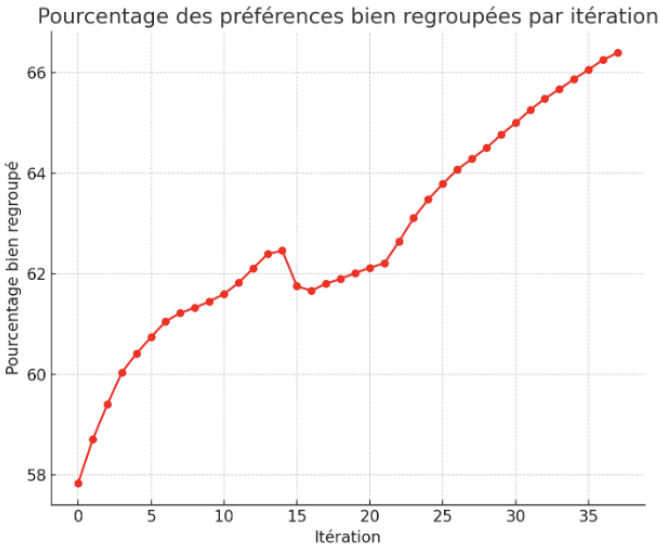
$$x^{(j)}, y^{(j)}, \sum_{i=1}^{n}(u_{k,i}(x^{(j)}) - u_{k,i}(y^{(j)})), k \in [\![1, K]\!]$$

Nous pouvons éventuellement ajouter un hyperparamètre h pour faire varier l'importance de la similarité des paires. Malheuresement, pour quelques valeurs de h , nous n'avons pas réussi à obtenir un pourcentage de paires dans les bons clusters croissants.

Cluster des similarités des paires Une autre idée pour éviter l'ambiguité entre les paires et les similarités des paires est de faire les clusters directement en prenant pour cluster à l'étape suivante le cluster maximisant la l'écart d'utilité de la paire : plus $x^{(j)}$ et $y^{(j)}$ diffèrent, moins elles ont la probabilité de faire parties du même cluster.

Nous obtenons alors pour résultats sur tout le dataset les résultats suivants :





III.3 Modèle UTA sur les clusters

Pour obtenir les modèles UTA sur les clusters, on formule l'existence d'un tel modèle comme un problème de programmation linéaire que l'on résout à l'aide de Gurobi.

Variables

— Valeur de la fonction marginale pour le critère i au point x_i^l :
$\forall i \in [\![1, n]\!], (u_{i,0} = 0 \wedge \forall l \in [\![1, L]\!], u_{i,l} \in \mathbb{R}^+)$

— Erreur lorsque la préférence j n'est pas représentée :
$\forall j \in \mathit{Cluster}, \sigma_j \in \mathbb{R}^+$

Objectif

Minimiser la somme des termes d'erreur :

$$\sum_{j \in Cluster}^{P} \sigma_j$$

Contraintes

1. Normalisation des fonctions d'utilité :
$\sum_{i=1}^{n} u_{i,L} = 1$
2. Croissance des fonctions marginales :
$\forall i \in [\![1, n]\!], \forall l \in [\![1, L]\!], u_{i,l} \geqslant u_{i,l-1}$
3. Explication des préférences :
$\forall j \in \mathit{Cluster}, u(x^{(j)}) + \sigma_j \geqslant u(y^{(j)}) + \epsilon$

Gurobi utilise l'algorithme du simplex pour résoudre ce type de problème. Étant donné que cet algorithme n'est pas parrallèle et que les machines modernes possèdent plusieurs cœurs, on peut donc résoudre le problème sur chaque cluster simultanément (en pratique il y a 3 clusters) sans qu'il y ait une compétition pour les ressources de la machine.

III.4 Résultats et pistes d'amélioration

Les meilleurs résultats ont été obtenus avec la méthode itérative (20 itérations) en initialisant à l'aide de l'algorithme *K-Means* en utilisant le produit scalaire modifié. Le modèle obtenu explique 99.985% des préférences (6 préférences non expliquées) et a un Rand index de 0.94723.

Les clusters obtenus ne correspondent pas parfaitement à la ground truth. Une solution que nous envisageons afin d'améliorer la qualité du clustering obtenu est d'apprendre plusieurs modèles, cinq par exemple, puis d'essayer de combiner d'une manière appropriée les clusters obtenus pour espérer avoir de meilleurs clusters, et éventuellement reprendre les itérations.