Clustering par apprentissage de distance guidé par des préférences sur les attributs

Adnan El Moussawi*,**, Ahmed Cheriat**,
Arnaud Giacometti*, Nicolas Labroche*, Arnaud Soulet*

*Université François Rabelais de Tours, 3 place Jean Jaures, 41000 Blois
{Prénom}.{Nom}@univ-tours.fr,
**Groupe Kalidea, 47 rue de l'Est, 92774 Boulogne-Billancourt Cedex
{aelmoussawi, acheriat}@kalidea.com

Résumé. Ces dernières années de nombreuses méthodes semi-supervisées de clustering ont intégré des contraintes entre paires d'objets ou d'étiquettes de classe, afin que le partitionnement final soit en accord avec les besoins de l'utilisateur. Pourtant dans certains cas où les dimensions d'études sont clairement définies, il semble opportun de pouvoir directement exprimer des contraintes sur les attributs pour explorer des données. De plus, une telle formulation permettrait d'éviter les écueils classiques de la malédiction de la dimensionnalité et de l'interprétation des clusters. Cet article propose de prendre en compte les préférences de l'utilisateur sur les attributs afin de guider l'apprentissage de la distance pendant le clustering. Plus précisément, nous montrons comment paramétrer la distance euclidienne par une matrice diagonale dont les coefficients doivent être au plus proche des poids fixés par l'utilisateur. Cette approche permet d'ajuster le clustering pour obtenir un compromis entre les approches guidées par les données et par l'utilisateur. Nous observons que l'ajout des préférences est parfois essentiel pour atteindre un clustering de meilleure qualité.

1 Introduction

Les méthodes de segmentation ou de clustering sont particulièrement adaptées à la recherche des différents profils des clients, par exemple pour leur proposer des programmes spécifiques de fidélisation. Néanmoins, des cas d'usages au sein de l'entreprise Kalidea [1] démontrent que de nombreux problèmes restent posés. Tout d'abord, par croisement des différentes mesures (chiffre d'affaire, quantité de produits vendus ou invendus, etc) et dimensions d'analyse possibles (temporelles, géographiques, gammes de produits, etc), les attributs candidats pour segmenter les données sont extrêmement nombreux. Ainsi, il s'avère rapidement illusoire d'effectuer une segmentation sur l'ensemble de tous les attributs d'analyse possibles, que ce soit du fait de la malédiction de la dimensionnalité ou de la difficulté à interpréter et analyser les clusters extraits. Par ailleurs, on constate souvent que les clusters produits ne sont pas toujours en accord avec les besoins ou intuitions des utilisateurs des outils de segmentation.

1. http://www.kalidea-ce.com/

Pour résoudre ces difficultés, différents types d'approches ont été proposées. Tout d'abord, face à la malédiction de la dimensionnalité, un premier type d'approche consiste à sélectionner ou pondérer les attributs d'analyse (Alelyani et al., 2013 ; Kumar et Minz, 2014) de manière à augmenter la discrimination entre données, par exemple en supprimant les attributs jugés redondants ou non pertinents. Beaucoup de ces travaux proposent des méthodes de sélection ou pondération d'attributs indépendantes des méthodes de clustering, avec le risque de supprimer des attributs intéressants. Dans ce papier, nous proposons une approche de pondération d'attributs directement intégrée à la méthode de clustering utilisée.

Dans le domaine du subspace clustering (Parsons et al., 2004), un deuxième type d'approche consiste à rechercher les différents sous-espaces (définis par un sous-ensemble d'attributs d'analyse) où des clusters pertinents peuvent être extraits. Un des inconvénients de ce type d'approche est que le nombre de sous-espaces où sont découverts des clusters pertinents peut être très important et rendre finalement l'interprétation des résultats difficiles. Dans ce papier, nous proposons une approche où l'utilisateur peut exprimer dès le départ des préférences sur les attributs d'analyse qui lui semblent potentiellement intéressants, ce qui lui permet d'orienter la recherche d'un sous-espace intéressant pour segmenter les données.

Dans ce cadre, l'approche présentée dans ce papier combine pendant la construction des clusters, à la fois l'apprentissage de poids sur les attributs d'analyse et l'utilisation de préférences utilisateurs sur ces attributs pour guider l'apprentissage des poids. Plus précisément, nous paramétrons la distance euclidienne utilisée par la méthode de clustering par une matrice diagonale dont les coefficients doivent être au plus proche des poids fixés par l'utilisateur. Cette approche ajuste le clustering pour obtenir un compromis entre une approche guidée par les données et une approche guidée par l'utilisateur. Notons que par rapport à des travaux où les utilisateurs peuvent exprimer des contraintes sur les données (Davidson et Basu, 2007) (de type must-link et cannot-link), notre approche exprime des contraintes sur les attributs d'analyse. De plus, ces contraintes sont souples et peuvent être remises en cause par les données si elles ne permettent pas de bien les discriminer.

Après la présentation d'un état de l'art dans la section 2, nous présentons dans la section 3 une formulation de notre proposition sous la forme d'une fonction objectif à minimiser, incluant à la fois une mesure de qualité du clustering recherché et une mesure de la distance entre les poids recherchés sur les attributs d'analyse et les poids souhaités d'un utilisateur. Dans la section 4, nous proposons un nouvel algorithme dérivé de K-Means, l'algorithme MAPK-Means permettant de minimiser la fonction objectif précédemment définie. Enfin, dans la section 5, nous présentons des expériences montrant comment l'ajout de préférences utilisateurs sur les attributs d'analyse peut conduire à l'obtention de clusterings de meilleur qualité. Nous concluons dans la section 6 par quelques perspectives, en particulier en soulignant l'intérêt de notre proposition dans le cadre du clustering exploratoire.

2 Travaux existants

Notre contribution est au carrefour de plusieurs domaines de recherche en clustering que nous allons mettre ici en perspective. Comme indiqué dans (Alelyani et al., 2013 ; Kumar et Minz, 2014) de nombreux travaux se sont intéressés à la sélection ou à la pondération d'attributs en classification ou en clustering, dans le but d'augmenter la discrimination entre données en supprimant les attributs redondants ou non pertinents. La sélection d'attributs repose sur

l'exploration de nombreux sous-espaces et de leur évaluation selon un critère (Parsons et al., 2004). Nos travaux se rapprochent plus de la pondération d'attributs (*features weighting*) et en particulier des méthodes de recherche d'attributs dites embarquées (*embedded*) (Jing et al., 2007), qui cherchent les attributs pertinents pendant le clustering. Nous ne nous plaçons donc pas dans le contexte des méthodes de filtrage (Dash et al., 2002) qui sont décorrelées de la méthode d'apprentissage et peuvent supprimer des attributs jugés à tort comme non pertinents. Parmi les approches dérivées de K-Means, Entropy Weighted K-Means (Jing et al., 2007) pénalise les solutions dans lesquelles les attributs ont le même poids, et LFSBSS (Li et al., 2008) minimise le recouvrement des clusters. Ces algorithmes reposent sur une recherche locale à chaque cluster des attributs pertinents, et ce faisant, se rapprochent plus du domaine de la recherche de sous-espaces ou *subspace clustering* (Kriegel et Zimek, 2010 ; Parsons et al., 2004) ou du clustering multi-vues (Bickel et Scheffer, 2004 ; Sublemontier et al., 2011).

Dans le cadre du subspace clustering (Kriegel et Zimek, 2010 ; Parsons et al., 2004), les méthodes énumèrent tous les sous-espaces où les clusters sont pertinents. Il existe donc un très grand nombre de clustering différents et la difficulté réside dans le choix de la bonne partition. De plus, comme pour les méthodes de pondération d'attributs précédentes, chaque cluster est défini dans son propre espace. Ce problème de partitions multiples existe aussi en clustering multi-vues (Bickel et Scheffer, 2004 ; Sublemontier et al., 2011) pour lequel plusieurs partitions possibles sont estimées, mais sur des sous-ensembles disjoints des attributs.

Nos travaux se rapprochent également des méthodes semi-supervisées qui utilisent des informations fournies par un expert sous la forme de contraintes sur des étiquettes de classe ou encore entre paires de données indiquant si les objets doivent ou non appartenir au même cluster (contraintes must-link et cannot-link) (Davidson et Basu, 2007). Plus formellement, trois approches principales ont été proposées pour prendre en compte les contraintes utilisateurs. La première consiste à modifier directement les instructions de l'algorithme en prenant explicitement en compte les contraintes, soit durant la phase d'initialisation (Basu et al., 2002), soit pendant la phase de convergence comme dans l'algorithme COP-Kmeans (Wagstaff et al., 2001). La deuxième approche consiste à sanctionner les partitions ne vérifiant pas l'ensemble des contraintes en ajoutant un terme de pénalité dans la fonction objectif. Par exemple, (Antoine et Labroche, 2015) pénalisent les solutions d'une partition crédale en fonction du nombre de clusters possibles pour l'affectation des points et, (Bouchachia et Pedrycz, 2003) introduisent un terme de pénalité dans un algorithme de type K-Means floues qui prend en compte la différence entre les appartenances observées et souhaitées des points aux clusters. La troisième étend les deux approches précédentes avec des modèles qui apprennent la distance pour le clustering en fonction des contraintes exprimées soit sous la forme d'une distance de Mahalanobis (Xing et al., 2002), soit sous la forme d'une distance euclidienne paramètrée par une matrice de poids indiquant l'importance relative des attributs (Bilenko et al., 2004).

Les travaux proposés dans cet article visent à produire une partition qui permet à la fois d'optimiser la qualité intrinsèque du clustering par l'apprentissage d'une distance la plus discriminante possible pour les données, mais également de respecter les contraintes d'un expert exprimées sur les attributs. Ce dernier point n'a, à notre connaissance, jamais été abordé dans la littérature. Nous proposons dans la section suivante une formulation de notre proposition sous la forme de termes de pénalité ajoutés à la fonction objectif, qui permettent d'apprendre une distance sous la forme de pondération des attributs de l'espace euclidien initial et en fonction

de préférences utilisateurs sur les attributs. Par simplicité, à la différence de (Bilenko et al., 2004), notre solution n'apprend qu'une seule distance pour l'ensemble des clusters, ce travail restant envisageable pour de futurs travaux.

3 Formulation du problème

Nous proposons de modéliser les préférences de l'utilisateur par une *matrice de préférences* qui est une matrice **A*** définie positive et diagonale qui porte sur les dimensions plutôt que sur des combinaisons de ces dernières. Chaque coefficient a_{ii} de $\mathbf{A}^*$ correspond à la pondération que l'utilisateur assigne au i-ème attribut [2]. Nous pourrions naïvement utiliser cette distance pour paramétrer la distance euclidienne de la manière suivante : $\|x - c_j\|_{\mathbf{A}^*} = \sqrt{(x - c_j)^T \mathbf{A}^* (x - c_j)}$. Cependant, comme cette représentation des données peut se révéler insuffisante pour bien séparer les clusters, il est préférable d'ajuster cette distance pour déformer l'espace. Cette déformation vise à réduire les distances entre les objets d'un même cluster, tout en maximisant les distances entre les objets de différents clusters. Suivant l'approche de Xing et al. (2002), cela revient à apprendre une distance en utilisant une matrice symétrique définie positive $\mathbf{A}$ qui paramètre également la distance euclidienne. Maximiser la fonction de vraisemblance avec cette généralisation du modèle des K-Means revient alors à minimiser la fonction objectif suivante :

$$\mathcal{I}_m = \sum_{j=1}^{K} \sum_{x \in \mathcal{X}_j} \|x - c_j\|^2_{\mathbf{A}} - N \log(\det(\mathbf{A})) \tag{1}$$

où K est le nombre de clusters, c_j le centre du j-ème cluster $\mathcal{X}_j$ et N le nombre total de points.

A ce stade, la matrice apprise $\mathbf{A}$ n'est pas biaisée par la matrice de préférences $\mathbf{A}^*$. Pour contraindre la matrice $\mathbf{A}$ à suivre les recommandations de l'utilisateur, il est nécessaire d'introduire une pénalité en cas de dissimilarité. Pour cela, nous proposons d'utiliser la divergence de Kullback-Leibler qui mesure la dissimilarité entre deux distributions. Dans notre cas, les deux distributions correspondent aux coefficients diagonaux de la matrice à apprendre $\mathbf{A}$ et d'une matrice de référence $\mathbf{B}$. Par conséquent, les matrices $\mathbf{A}$ et $\mathbf{B}$ sont normalisées telles que $\sum_{i=1}^{M} a_i = 1$ et $\sum_{i=1}^{M} b_i = 1$, où M est le nombre d'attributs :

$$D_{KL}(\mathbf{B} \| \mathbf{A}) = \sum_{i=1}^{M} b_i \log\left(\frac{b_i}{a_i}\right) \tag{2}$$

Nous modifions l'équation 1 en utilisant cette divergence à deux niveaux. Comme suggéré ci-avant, nous contrebalançons l'apprentissage de la distance en ajoutant un terme $D_{KL}(\mathbf{A}^* \| \mathbf{A})$ qui mesure la divergence entre la distance apprise $\mathbf{A}$ et les préférences utilisateurs $\mathbf{A}^*$ sur cette distance. Nous homogénéisons ensuite la partie séparation de clusters avec la partie satisfaction des préférences en adaptant la fonction objectif $\mathcal{I}_m$. Intuitivement, le terme $-N \log(\det(\mathbf{A}))$ de l'équation 1 correspond à un apprentissage de distance en empêchant que $\mathbf{A}$ s'écarte trop d'un K-Means traditionnel où toutes les dimensions ont un poids équivalent. Ce terme peut être exactement reformulé comme étant la divergence entre la matrice à apprendre et une matrice uniforme i.e., $D_{KL}(\mathbf{U} \| \mathbf{A})$ avec $\mathbf{U} = 1/M \mathbf{I}_M$ (où $\mathbf{I}_M$ la matrice identité de dimension M).

2. Dans la suite, comme tous les coefficients manipulés sont sur la diagonale le coefficient a_{ii} est dénoté a_i.

Ainsi, pour toute matrice diagonale définie positive $\mathbf{A}$ dont la trace est égale 1, nous définissons la fonction objectif suivante :

$$\mathcal{I}_{map} = \sum_{j=1}^{K} \sum_{x \in \mathcal{X}_j} \|x - c_j\|_{\mathbf{A}}^2 + (1-\omega) N D_{KL}(\mathbf{U}\|\mathbf{A}) + \omega N D_{KL}(\mathbf{A}^*\|\mathbf{A}) \quad (3)$$

où nous introduisons le coefficient ω qui pondère l'importance des préférences utilisateurs face à un apprentissage de distance classique où toutes les dimensions ont le même poids. Il est à noter que l'ajout du nombre de points N devant la divergence $D_{KL}(\mathbf{A}^*\|\mathbf{A})$ est nécessaire pour que cette divergence est le même poids que l'autre. Si $\omega = 1$, $\mathcal{I}_{map}$ fixe la matrice $\mathbf{A}$ en ne tenant compte que des préférences utilisateur et des distances intra-clusters. Si $\omega = 0$, la minimisation de $\mathcal{I}_{map}$ correspond à la minimisation de l'équation 1 (i.e., la fonction objectif de MPCK-Means sans contraintes (Bilenko et al., 2004)).

Etant donné un ensemble de points $\mathcal{X}$, un nombre de clusters $K \geq 1$, une matrice de préférences $\mathbf{A}^*$ et $\omega \in [0, 1]$, notre objectif est de trouver une K-partition $\{\mathcal{X}_j\}_{j=1}^{K}$ des données minimisant la fonction objectif $\mathcal{I}_{map}$ en apprenant une matrice $\mathbf{A}$.

4 Algorithme MAPK-Means

Notre algorithme suit le schéma d'optimisation introduit par MPCK-Means (Bilenko et al., 2004) qui intuitivement consiste à alterner trois phases : 1) réaffecter les points pour avoir une partition minimisant la distance paramétrée par $\mathbf{A}$, 2) recalculer les centres de chaque cluster et 3) ajuster la matrice $\mathbf{A}$ de sorte à minimiser la fonction objectif. Plus précisément, l'algorithme MAPK-Means (pour *Metric Attribute Preferential K-Means* présenté par l'algorithme 1) prend en entrée un ensemble de points $\mathcal{X}$, un nombre de clusters K, les préférences sur les attributs $\mathbf{A}^*$ (avec $\sum_{i=1}^{M} a_i = 1$) et $\omega \in [0, 1]$. Il retourne une K-partition $\{\mathcal{X}_j\}_{j=1}^{K}$ des données minimisant la fonction objectif $\mathcal{I}_{map}$ en apprenant une matrice $\mathbf{A}$.

Initialisation de l'algorithme Notre initialisation reprend celle de K-Means++ (Arthur et Vassilvitskii, 2007) à la ligne 1. Le premier centre est choisi au hasard parmi les données $\mathcal{X}$. Ensuite, chacun des $K-1$ autres centres initiaux est tiré aléatoirement avec une distribution proportionnelle à la somme des distances avec les centres déjà choisis. Ensuite, la matrice $\mathbf{A}$ est initialisée de sorte à donner le même poids à chacun des attributs (ligne 2) : $a_i = \frac{1}{M}$ pour tout $i \in \{1, \ldots, M\}$.

Affecter les points à chaque cluster L'affectation des points à un cluster s'effectue de la même manière que K-Means (ligne 4 à 6) mais avec une distance paramétrée par $\mathbf{A}$. A chaque itération, chaque point est affecté au centre le plus proche (ligne 5). Cette affectation en minimisant la distance intra-cluster tend également à minimiser globalement la fonction objectif $\mathcal{I}_{map}$.

Recalculer les centres de chaque cluster Une fois que tous les points sont affectés à un cluster, on met à jour le centre de chaque cluster en calculant le barycentre des points qui lui sont affectés (ligne 6). Contrairement à certaines approches comme MPCK-Means, le calcul des centres est insensible à l'ordre de l'affectation des points dans l'étape précédente.

Algorithme 1 : MAPK-Means

Input : un ensemble de points $\mathcal{X}$, un nombre de clusters K,
une matrice de préférences $\mathbf{A}^*$, ω
Output : une partition $\{\mathcal{X}_j\}_{j=1}^K$

1 : Tirer K centres $\{c_j\}_{j=1}^K$ selon l'initialisation de K-Means++
2 : Initialiser $\mathbf{A} := \frac{1}{M}\mathbf{I_m}$ où I_M est la matrice identité de dimension M
3 : **repeat**
4 : *// mettre à jour la partition* $\{\mathcal{X}_j\}_{j=1}^K$
5 : $\mathcal{X}_j := \left\{x \in \mathcal{X} : j = \arg\min_{l\in\{1,\dots,K\}} \|x - c_l\|_{\mathbf{A}}\right\}$ pour tout $j \in \{1,\dots,K\}$
6 : $c_j[i] := \frac{\sum_{x\in\mathcal{X}_j} x[i]}{|\mathcal{X}_j|}$ pour tout $i \in \{1,\dots,M\}$ et pour tout $j \in \{1,\dots,K\}$
7 : *// mettre à jour la matrice* $\mathbf{A}$
8 : $a_i := \frac{(1-\omega)/M + \omega a_i^*}{S_i}$ pour tout $i \in \{1,\dots,M\}$
9 : Normaliser a_i pour tout $i \in \{1,\dots,M\}$ pour que $\sum_{i=1}^M a_i = 1$
10 : **until** partition $\{\mathcal{X}_j\}_{j=1}^K$ est stable
11 : **return** $\{\mathcal{X}_j\}_{j=1}^K$

Apprendre la distance Cette dernière étape effectue l'apprentissage de distance où $\mathbf{A}$ est recalculée de sorte à minimiser la fonction objectif $\mathcal{I}_{map}$ (ligne 7 à 9). Chaque mise à jour de la matrice $\mathbf{A}$ est calculée en prenant la dérivée $\frac{\partial \mathcal{I}_{map}}{\partial a_i}$ égale à zéro :

$$\frac{\partial \mathcal{I}_{map}}{\partial a_i} = \sum_{j=1}^{K}\sum_{x\in\mathcal{X}_j} \|x[i] - c_j[i]\|^2 - \frac{(1-\omega)N}{Ma_i} - \omega N\frac{a_i^*}{a_i} = 0$$

Nous posons que $S_i = \sum_{j=1}^K \sum_{x\in\mathcal{X}_j} \|x[i] - c_j[i]\|^2$ est la distance intra-cluster sur la dimension i (supposée non-nulle). De cette manière, nous obtenons la mise à jour à apporter à a_i :

$$a_i = N\frac{(1-\omega)/M + \omega a_i^*}{S_i} \tag{4}$$

On constate bien que le poids de l'attribut i augmente lorsque la distance S_i sur cette dimension est faible. Au numérateur, on retrouve le compromis entre la matrice équilibrée $\mathbf{U}$ et la matrice de préférences $\mathbf{A}^*$ pondérées par ω. Bien sûr, lorsque ω est nul, les préférences de l'utilisateur ne sont plus prises en compte et l'étape de mise à jour est similaire à celle de MPCK-Means. A l'inverse, lorsque ω est égal à 1, chaque poids a_i^* fixé par l'utilisateur est considéré avec une pondération liée à la distance inter-cluster sur la dimension i.

La mise à jour issue de l'équation 4 est opérée à la ligne 8 où le nombre de points N qui devrait apparaître au numérateur est supprimé à cause de l'étape de normalisation. En effet, nous procédons à une étape de normalisation pour que $\sum_{i=1}^M a_i$ soit égale à 1 (ligne 9). Cette étape est essentielle pour respecter les propriétés de la divergence de Kullback-Leibler.

5 Expérimentations

5.1 Protocole expérimental

Jeu de données : afin d'illustrer le fonctionnement de MAPK-Means, nous avons utilisé trois jeux de données : *Iris* ($N = 150$ données, $M = 4$ attributs et $K = 3$ clusters), *Wine* ($N = 178$, $M = 13$, $K = 3$) et *Ionosphere* ($N = 351$, $M = 34$, $K = 2$), issus de la base UCI [3]. Ces cas de tests permettent de mettre en évidence le comportement de MAPK-Means par rapport à des données de dimensions réduite (*Iris*) mais également des données de plus grande dimension (*Wine* et *Ionosphere* [4]).

Évaluation de résultats : afin d'évaluer les résultats de notre algorithme, nous avons étudié l'évolution de deux facteurs par rapport à la variation des valeurs de ω.

Le premier facteur concerne les poids associés aux dimensions. Pour cela, les travaux effectués dans le domaine de *feature selection* de (Dy et Brodley, 2004) sur les mêmes jeux de données permettent de connaître une forme de vérité terrain qui nous permet de qualifier par la suite les dimensions de "bonnes" ou "mauvaises" dans nos tests.

Le second facteur est l'Indice de Rand Ajusté (ARI) (Hubert et Arabie, 1985), une valeur proche de 1 indiquant que le clustering produit est très similaire au clustering "naturel" (i.e., forte similarité entre le partitionnement généré par notre démarche et le clustering proposé par les données sources).

Démarche suivie : nos expérimentations ont pour objectif de montrer l'impact du choix de l'utilisateur sur le résultat de clustering. En effet, notre démarche sert à mettre en évidence que la meilleure solution n'est pas toujours celle obtenue par un apprentissage automatique, ni celle guidée uniquement par l'utilisateur mais une troisième option où l'utilisateur indique ses préférences et l'algorithme cherche à trouver la meilleure solution, en étant le plus proche possible de ses préférences.

Pour ce faire, pour chaque jeu de données, des poids initiaux ont été associés aux dimensions selon les quatre scénarios suivants : (1) des poids aléatoires sur les différentes dimensions, (2) une distribution uniforme des poids, (3) des poids plus élevés sont associés aux bonnes dimensions, (4) des poids plus élevés sont accordés aux mauvaises dimensions.

Ensuite, pour chacun des cas ci-dessus, nous avons fait décroître la valeur de ω de 1 à 0 avec un pas égal à 1/50. Le but est de commencer par un clustering pour lequel le choix de l'utilisateur sera favorisé ($\omega = 1$) pour aller vers un clustering avec apprentissage automatique pur ($\omega = 0$).

Enfin, pour chaque valeur de ω, plusieurs initialisations ($N/10$) des centres de clusters ont été réalisées avec K-Means++ (cf. section 4). Le clustering sélectionné est celui qui permet d'avoir la plus petite valeur de la fonction objectif de K-Means (1).

3. `archive.ics.uci.edu/ml/datasets.html`

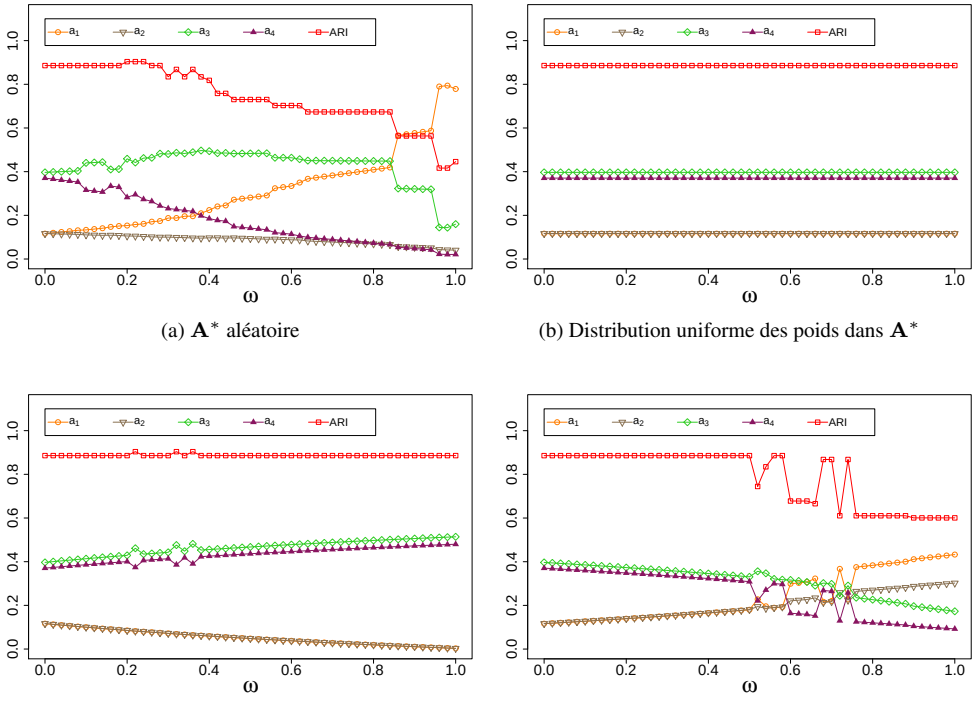
4. Les dimensions 1 et 2 sont éliminées de l'analyse, la première est binaire et la deuxième est invariante (égale à 0) pour tous les points.

5.2 Résultats et discussions

Dans cette partie, nous présentons les résultats obtenus suite aux quatre scénarios d'initialisations de poids (cf. section 5.1) en utilisant les données *Iris*. Ensuite, nous nous limitons au scénario donnant un résultat remarquable, pour *Wine* et *Ionosphere*.

5.2.1 Données *Iris*

La première expérience concerne une initialisation avec des poids aléatoires sur les différentes dimensions ($a_1^* = 0.6, a_2^* = 0.1, a_3^* = 0.25, a_4^* = 0.05$). La figure (1a) indique l'évolution des poids des dimensions (a_i) par rapport aux valeurs de ω. On observe que les préférences utilisateurs sont bien respectées lorsque $\omega = 1$. Ensuite, en réduisant la valeur de ω, les poids des attributs changent de manière à associer les poids les plus forts aux bonnes dimensions (3 et 4), et les plus faibles aux autres. L'ARI augmente également, ce qui indique que le clustering obtenu se rapproche de plus en plus du clustering naturel. A noter que l'instabilité des poids pour certaines valeurs de ω est liée à l'initialisation des centres des clusters.



(a) $\mathbf{A}^*$ aléatoire

(b) Distribution uniforme des poids dans $\mathbf{A}^*$

(c) Poids plus élevés pour les bonnes dimensions

(d) Poids plus élevés pour les mauvaises dimensions

FIG. 1 : *Iris* : Variation des poids des dimensions et de l'Indice de Rand Ajusté (ARI) par rapport aux valeurs de ω, avec quatre scénarios différents de préférences utilisateur.

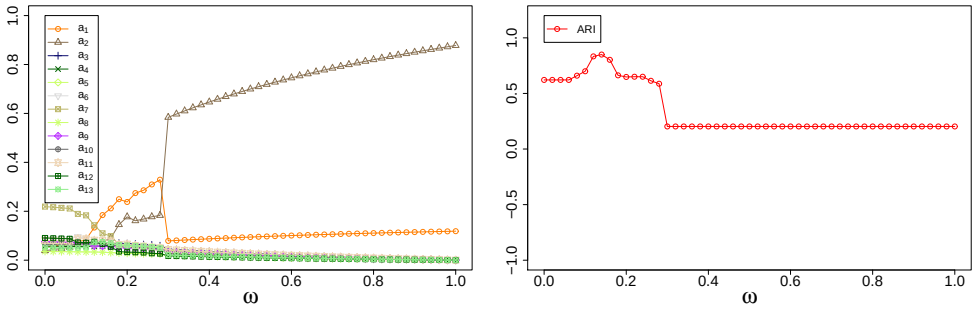


FIG. 2 : Variation des poids des dimensions (gauche) et de l'Indice de Rand Ajusté (droite) par rapport à ω pour les données *Wine*.

Si les préférences utilisateurs représentent une distribution uniforme des poids, alors le poids de chaque dimension i (où $i = 1, ..., 4$) est stable et égal à $\frac{1}{M.S_i}$ pour tout ω selon l'équation (4, section 4). La figure (1b) montre cette stabilité qui a permis d'avoir une solution unique de clustering avec un ARI fixe. Les poids les plus forts dans cette solution sont toujours attribués aux bonnes dimensions (3 et 4).

Lorsque les préférences utilisateurs donnent des poids plus élevés aux bonnes dimensions, par exemple avec des poids initiaux à 0.01 pour les dimensions 1 et 2 et 0.49 pour les dimensions 3 et 4, la figure (1c) montre la cohérence entre les préférences d'utilisateur (poids) et les résultats obtenus par MAPK-Means où les valeurs les plus élevées sont associées aux dimensions privilégiées dès le départ. L'ARI, quasiment stable et élevé pour tout ω, indique que le résultat obtenu est très proche du clustering attendu.

Enfin, on considère le cas où l'utilisateur donne des poids plus élevés aux mauvaises dimensions. Ce test sert à simuler le scénario où l'utilisateur choisit des dimensions non pertinentes pour le clustering. Les poids initiaux dans ce cas sont 0.4 pour les dimensions 1 et 2, et 0.1 pour les dimensions 3 et 4. Comme le montre la figure (1d), l'ARI est faible au départ (entre $\omega = 0.8$ et $\omega = 1$), ce qui est cohérent avec les poids élevés des mauvaises dimensions (1 et 2). En revanche, l'ARI augmente sensiblement quand les dimensions 3 et 4 sont de nouveaux favorisées par l'apprentissage de la distance avec MAPK-Means.

Les quatre résultats décrits ci-dessus illustrent l'efficacité de notre approche dans le cas de données de dimensions réduites : *(i)* les préférences de l'utilisateur définies par $\mathbf{A}^*$ sont toujours respectées pour un ω proche de 1, *(ii)* l'apprentissage automatique de la distance a tendance à associer les poids les plus élevés aux *bonnes dimensions*, *(iii)* la stabilité de l'Indice de Rand Ajusté, pour certaines valeur de ω, montre qu'il existe un nombre limité de clustering alternatifs pouvant être proposés à l'utilisateur pour tout ω.

5.2.2 Données *Wine*

Dans ce test, les poids initiaux sont 0.4945 pour les deux premiers dimensions et 0.001 pour toutes les autres, figure (2). Les poids des dimensions 1 et 2 ont tendance à décroître lentement (pour ω allant de 1 à 0.3). L'ARI correspondant est constant. Pour $\omega = 0.28$, la divergence

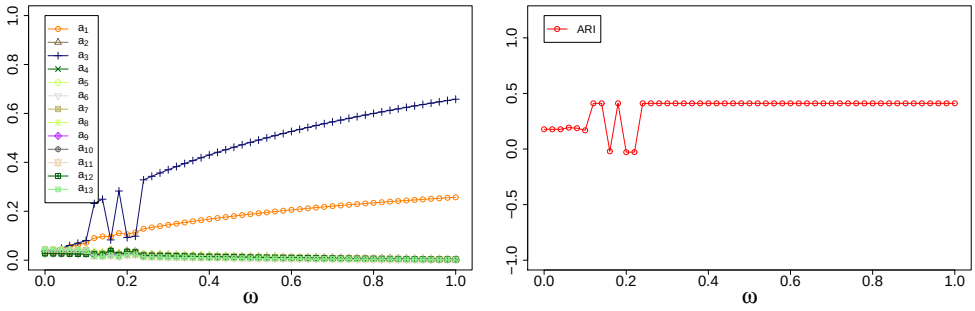


FIG. 3 : Variation des poids des dimensions (gauche) et de l'Indice de Rand Ajusté (droite) par rapport à ω pour les données *Ionosphere*.

entre $\mathbf{A}$ et $\mathbf{A}^*$ est devenue plus faible que sa divergence avec $\mathbf{U}$ (la distribution uniforme des poids) ce qui force MAPK-Means à équilibrer les poids, cela explique la grande chute de a_2. L'ARI est maximisé pour ω proche de 0.14, où le poids a_7 de la dimension 7 (qualifié parmi les bonnes dimensions) est devenu plus élevé.

5.2.3 Données *Ionosphere*

Dans cette expérimentation, des poids plus élevés ont été affectés initialement aux *bonnes dimensions* 1 et 3 ($a_1 = a_3 = 0.401$) et des poids faibles pour les autres ($a_i = 0.0066$, pour $i \neq 1, 3$). La figure (3) montre que les valeurs a_1 et a_3 diminuent en allant de $\omega = 1$ à $\omega = 0$, jusqu'à atteindre un état d'équilibre entre les différents poids [5] à partir de $\omega = 0.12$. La valeur de l'ARI, qui correspond à $\omega \in [0.12, 1]$, est quasiment stable, puis il décroît en se rapprochant de $\omega = 0$. Cette décroissance est due à un apprentissage avec un grand nombre de dimensions pouvant contenir du bruit, or initialement le résultat est généré à partir de données avec un nombre réduit de bonnes dimensions.

Les résultats obtenus avec *Wine* et *Ionosphere* montrent que le meilleur clustering peut être obtenu par un compromis entre un apprentissage automatique de la distance et les préférences de l'utilisateur. Nos résultats montrent que l'intervention de l'utilisateur est importante pour arriver à la bonne solution dans le cas d'un nombre important d'attributs.

6 Conclusion

Nous avons montré comment prendre en compte des préférences utilisateurs sur les attributs d'analyse dans un processus de clustering. Pour ce faire, nous avons introduit dans la fonction objectif à minimiser - qui inclut un terme classique visant à rechercher le clustering de meilleure qualité -, un terme supplémentaire mesurant la distance entre les poids appris sur

5. Nous avons limité l'affichage aux poids des 13 premières dimensions, pour des raison de lisibilité du graphe. Sachant que les poids obtenus pour les autres dimensions sont compris entre 0 et 0.05 pour toutes les valeurs de ω.

les attributs et les poids préférés de l'utilisateur. Cette approche permet d'obtenir un clustering qui est un compromis entre une approche guidée par les données et une approche guidée par l'utilisateur. Dans la partie expérimentale, nous avons montré que les clustering ainsi obtenus peuvent être de meilleure qualité que le clustering obtenu sans préférences utilisateurs, en particulier lorsque le nombre d'attributs d'analyse possibles est élevé.

Du fait de l'approche suivie (par ajout d'un terme à une fonction d'objectif classique), notre approche peut aisément se généraliser au cas où une matrice de poids est apprise pour chacune des classes construites. Cette généralisation est particulièrement importante pour les jeux de données de grande dimension, où les attributs d'analyse les plus pertinents ne sont pas nécessairement les mêmes pour tous les clusters. Par ailleurs, en intégrant à notre fonction objectif les termes de pénalité proposés dans (Bilenko et al., 2004), on peut noter que notre approche pourrait prendre simultanément en compte des contraintes sur les objets (de type must-link et cannot-link) et nos préférences sur les attributs.

En collaboration avec l'entreprise Kalidea, nous étudions actuellement comment intégrer notre approche à un système de clustering exploratoire utilisant les opérateurs OLAP classiques. Par exemple, après avoir calculé le clustering optimal selon les préférences de l'utilisateur (calculé avec $\omega = 1$), et en recherchant le paramètre $\omega < 1$ permettant d'obtenir un clustering alternatif significativement différent du clustering optimal selon les préférences utilisateurs, il est possible de recommander à l'utilisateur de nouveaux attributs d'analyse, à savoir les attributs d'analyse pour lesquels le poids appris est significativement différent du poids désiré par l'utilisateur.

Remerciement : Nous tenons à remercier le groupe Kalidea pour son soutien en termes de données, de *use cases* ainsi que la disponibilité de ses équipes. Nous remercions également l'ANRT pour leur soutien financier dans le cadre d'une thèse CIFRE (2014/0658).

Références

Alelyani, S., J. Tang, et H. Liu (2013). Feature selection for clustering : A review. In *Data Clustering : Algorithms and Applications*, pp. 29–60. Chapman and Hall/CRC.

Antoine, V. et N. Labroche (2015). Classification évidentielle avec contraintes d'étiquettes. In *Proc. of EGC Conference*, pp. 125–136.

Arthur, D. et S. Vassilvitskii (2007). k-means++ : The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035. Society for Industrial and Applied Mathematics.

Basu, S., A. Banerjee, et R. J. Mooney. (2002). Semi-supervised clustering by seeding. In *Proceeding of the 19th International Conference on Machine Learning*, pp. 27–34.

Bickel, S. et T. Scheffer (2004). Multi-view clustering. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004)*, pp. 19–26.

Bilenko, M., S. Basu, et R. J. Mooney (2004). Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 11. ACM.

Bouchachia, A. et W. Pedrycz (2003). A semi-supervised clustering algorithm for data exploration. In *Proc. Internat. Fuzzy Systems Association World Congress*, pp. 328–337.

Dash, M., K. Choi, P. Scheuermann, et H. Liu (2002). Feature selection for clustering a filter solution. In *Proc. of the 2nd International Conference on Data Mining*, pp. 115–122.

Davidson, I. et S. Basu (2007). A survey of clustering with instance level constraints. *ACM Transactions on Knowledge Discovery from data*, 1–41.

Dy, J. G. et C. E. Brodley (2004). Feature selection for unsupervised learning. *J. Mach. Learn. Res. 5*, 845–889.

Hubert, L. et P. Arabie (1985). Comparing partitions. *Journal of Classification 2*(1), 193–218.

Jing, L., M. K. Ng, et J. Z. Huang (2007). An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. *IEEE Trans. on Knowl. and Data Eng. 19*(8), 1026–1041.

Kriegel, H.-P. et A. Zimek (2010). Subspace clustering, ensemble clustering, alternative clustering, multiview clustering : what can we learn from each other. *Proceedings of MultiClust, 1st Intl. Workshop on Discovering, Summarizing and Using Multiple Clusterings, KDD*.

Kumar, V. et S. Minz (2014). Feature selection : A literature review. *Smart CR 4*(3), 211–229.

Li, Y., M. Dong, et J. Hua (2008). Localized feature selection for clustering. *Pattern Recognition Letters 29*(1), 10–18.

Parsons, L., E. Haque, et H. Liu (2004). Subspace clustering for high dimensional data : A review. *SIGKDD Explor. Newsl. 6*(1), 90–105.

Sublemontier, J.-H., G. Cleuziou, M. Exbrayat, et L. Martin (2011). Clustering multi-vues : une approche centralisée. *Revue des Nouvelles Technologies de l'Information, numéro spécial "Fouille de Données Complexes : données multiples*.

Wagstaff, K., C. Cardie, S. Rogers, et S. Schroedl (2001). Constrained k-means clustering with background knowledge. In *Proceedings of the 18th International Conference on Machine Learning*, pp. 577–584.

Xing, E. P., M. I. Jordan, S. Russell, et A. Y. Ng (2002). Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pp. 505–512.

Summary

In recent years many semi-supervised clustering methods have integrated constraints between pairs of objects or class of labels, so that the final partition is consistent with the needs of the user. However in some cases where the dimensions of studies are clearly defined, it seems appropriate to directly express constraints on the attributes to explore the data. Furthermore, such formulation would avoid the classic problems of the curse of dimensionality and the interpretation of the clusters. This article proposes to take into account the preferences of the user on the attributes to guide the learning of the distance for clustering. Specifically, we show how to parameterize the Euclidean distance with a diagonal matrix whose coefficients must be closest to the weight set by the user. This approach builds a compromise clustering between a data-driven and a user-driven solution. We observe experimentally that the addition of preferences may be essential to achieve a better clustering.