Clustering en haute dimension par accumulation de clusterings locaux

Marc-Ismaël Akodjènou-Jeannin *, Kavé Salamatian*
Patrick Gallinari *

*104, avenue du Président Kennedy
75016 Paris
{Marc-Ismael.Akodjenou, Kave.Salamatian, Patrick.Gallinari}@lip6.fr
http://www.lip6.fr

Résumé. Le clustering est une tâche fondamentale de la fouille de données. Ces dernières années, les méthodes de type *cluster ensembles* ont été l'objet d'une attention soutenue. Il s'agit d'agréger plusieurs clusterings d'un jeu de données afin d'obtenir un clustering "moyen". Les clusterings individuels peuvent être le résultat de différents algorithmes. Ces méthodes sont particulièrement utiles lorsque la dimensionalité des données ne permet pas aux méthodes classiques basées sur la distance et/ou la densité de fonctionner correctement. Dans cet article, nous proposons une méthode pour obtenir des clusterings individuels à faible coût, à partir de projections partielles du jeu de données. Nous évaluons empiriquement notre méthode et la comparons à trois méthodes de différents types. Nous constatons qu'elle donne des résultats sensiblement supérieurs aux autres.

1 Introduction

Le clustering consiste à découvrir automatiquement des groupes ("clusters") présents dans le jeu de données. Une littérature abondante existe sur le sujet (une revue des principales méthodes peut être trouvée dans Rui et Wunsch (2005)). Nous nous plaçons ici dans le cadre des "cluster ensembles" (Strehl et Ghosh (2002)). Les "cluster ensembles" sont une sorte de méta-clustering : à partir de plusieurs clusterings du même jeu de données, on déduit un clustering "moyen" (Strehl et Ghosh (2002)). Plusieurs alternatives ont été proposées pour trouver le clustering moyen (méthodes agglomératives, ou basées sur des graphes). Indépendamment de la méthode de synthèse choisie, il est clair que le clustering moyen dépend fortement de la qualité et de la diversité de chaque clustering individuel (Fern et Brodley (2003)). Par exemple, agréger plusieurs clusterings issus de l'algorithme K-means avec des initialisations différentes atténuera les erreurs particulières dues à chaque clustering individuel ; cependant cela ne permettra pas de contourner les limitations fondamentales de l'algorithme (clusters de forme sphérique, sensibilité à la dimension...). La situation idéale pour les cluster ensembles est celle où les clusterings individuels sont variés, de bonne qualité et obtenus à faible coût.

L'idée explorée par Topchy et al. (2003) est d'obtenir ces clusterings individuels en projetant le jeu de données sur une direction aléatoire, et en faisant un clustering simple sur la

projection (qui revient essentiellement à trouver les modes de la densité de la projection). L'intuition est qu'avec suffisamment de droites, les séparations entre clusters seront mises en évidence et permettront donc d'obtenir un clustering moyen de bonne qualité. Dans cet article, nous proposons d'améliorer cette méthode en construisant les clusterings individuels avec d'autres projections, toujours à faible coût mais avec de meilleures performances.

Le reste de cet article est structuré de la façon suivante : La section 2 explique notre approche, et décrit formellement les projections partielles des points, la recherche de modes et le clustering final. La section 3 compare les performances de la méthode avec celles d'autres méthodes sur des jeux de données réels. La section 4 conclut et évoque des pistes de travaux futurs.

2 Cluster ensembles et projections

2.1 Projections intéressantes pour le clustering

La question que nous nous posons est la suivante : comment trouver des projections *intéressantes pour le clustering*, qui soient variées et obtenues à faible coût ? On trouve dans la littérature de nombreux algorithmes de clustering qui utilisent des projections linéaires du jeu de données. Cela est le plus souvent justifié par la "malédiction de la dimensionalité" : quand la dimension augmente, les clusters deviennent épars (les points d'un cluster sont moins concentrés) et les distances entre les points tendent à perdre leur signification. Les clusters sont également dans des sous-espaces de dimension assez basse par rapport à la dimension de l'espace (Parsons et al. (2004)). Cette "malédiction" est un frein à l'efficacité de plusieurs méthodes de clustering basées sur les distances et/ou la densité. L'intérêt des projections est qu'elles permettent de faire apparaître plus clairement les clusters. Traditionnellement, c'est dans le domaine de la réduction dimensionnelle qu'on a d'abord cherché à trouver des directions "intéressantes" en fonction de certains critères (une direction est simplement une droite qui passe par l'origine). La plus utilisée est sans doute l'analyse en composantes principales (ACP), qui consiste à déterminer les k directions où le jeu de données présente la plus grande variance. La Projection Pursuit est une approche du même type, où un problème d'optimisation est résolu pour trouver les directions qui maximisent un "intérêt" lié à l'entropie. On peut aussi citer les projections aléatoires qui reviennent à projeter les données sur un ensemble de directions choisies uniformément sur la sphère unité.

Une démarche répandue consiste à d'abord réduire la dimension du jeu de données en projetant, puis d'appliquer un algorithme de clustering quelconque sur cette représentation. Cependant, la difficulté de paramétrer la réduction dimensionnelle pour obtenir un bon clustering a conduit certains auteurs à intégrer cette réduction directement dans l'algorithme de clustering. Par exemple, certains algorithmes de clustering divisif partitionnent récursivement le jeu de données en trouvant une direction "intéressante" et en coupant le jeu de données en deux par rapport à cette direction (Boley (1998) avec une ACP, Miasnikov et al. (2004) avec une Projection Pursuit). Les approches de Aggarwal et Yu (2000) et Ding et Li (2007) utilisent des ACP lors du clustering pour associer des sous-espaces "intéressants" à chaque cluster.

Dans l'article de Topchy et al. (2003) cité dans l'introduction, les droites de projection sont choisies au hasard et passent par l'origine. Le problème est que lorsque la dimension augmente, la projection sur une droite aléatoire tend à avoir une densité gaussienne unimodale,

donc inintéressante pour le clustering. Une amélioration qui vient immédiatement à l'esprit est d'utiliser par exemple une ACP, ou plusieurs ACP locales. Mais cette alternative est coûteuse : la complexité du calcul d'une ACP est quadratique avec la dimension. La direction de plus forte variance n'est d'ailleurs pas forcément la meilleure pour séparer des clusters. Le même problème de complexité et de manque de garanties se pose pour les Projection Pursuit. De plus ces deux alternatives cherchent des *directions*, c'est-à-dire essentiellement des droites passant par l'origine. Il est probable cependant qu'une direction intéressante pour le clustering à un endroit donné de l'espace ne le soit pas à un autre.

2.2 Notre approche

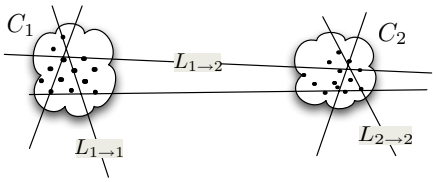

FIG. 1 – *Droites inter-points*

Nous proposons deux idées simples pour obtenir des clusterings individuels de bonne qualité. La première est d'utiliser des droites inter-points comme droites de projection. Si les deux points définissant la droite sont dans deux clusters différents, chacun dans son sous-espace, (comme c'est le cas pour $L_{1\to 2}$ sur la Figure 1), les projections de ces deux clusters sur cette droite seront vraisemblablement bien séparés. Si en revanche les deux points définissant la droite sont dans le même sous-espace, comme $L_{1\to 1}$ par exemple, la droite pourra séparer des clusters appartenant au même sous-espace.

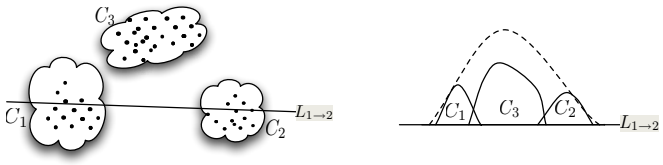

FIG. 2 – *Clusters, droite de projection et densité résultante*

La deuxième idée est de ne pas projeter tout le jeu de données sur toutes les droites, afin de mieux faire ressortir les clusters. Considérons par exemple le cas de la Figure 1, où les clusters C_1 et C_2 sont dans deux sous-espaces différents. Si une droite appartient à un sous-espace seulement, par exemple $L_{1\to 1}$, il n'est pas vraiment judicieux de projeter les points de C_2 dessus. Plus généralement, seuls les points proches de la droite (au sens de la projection orthogonale) sont intéressants pour une droite donnée. Ceci est illustré sur la Figure 2 : si on ne projette que les points des clusters C_1 et C_2, la densité résultante permet de bien les séparer,

alors que si on projette tous les points, la densité résultante (en pointillés) est unimodale et ne sépare plus rien.

Dans la suite, on note $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$ le jeu de données. La sphère unité est notée $\mathbb{S}^d$. Le produit scalaire est noté $\langle \cdot, \cdot \rangle$. Le cardinal d'un ensemble E est noté $|E|$.

2.3 Projections partielles

Droites de projection Nous choisissons au hasard M droites inter-points comme droites de projection. Chaque droite $D_k = (u_k, o_k)$ est définie par son vecteur directeur $\vec{u_k} \in \mathbb{S}^d$ et son "origine" $o_k \in \mathbb{R}^d$. D_k est définie en prenant au hasard deux points $x_i, x_j \in X$ de sorte que

$$\vec{u_k} = \frac{(x_j - x_i)}{\|x_j - x_i\|} \qquad \text{et} \qquad o_k = x_i$$

La distance d'un point x à la droite D_k est la distance orthogonale du point à la droite. On peut décomposer le vecteur $\vec{Ox}$ comme étant la somme vectorielle $(\vec{Ox'} + \vec{x'x})$, x' étant la projection de x sur la droite D_k. La distance $d(x, D_k)$ du point x à la droite D_k est la norme de $\vec{x'x}$, qui s'écrit

$$d(x, D_k) = \sqrt{\|x - o_k\|^2 - \langle (x - o_k), u_k \rangle^2}$$

Projections partielles Pour éviter les effets évoqués dans la sous-section 2.1, on ne projette réellement un point $x \in X$ que sur les droites les plus proches de lui. Pour cela, on calcule les distances entre chaque point et les droites. On sélectionne pour x les m droites les plus proches de lui en triant ces distances par ordre croissant. A la fin de ce traitement, chaque droite D_k est donc associée à un certain ensemble de points X_k. On note X_k' la projection de X_k sur D_k.

2.4 Recherche de modes

Pour produire un clustering individuel à partir de X_k', nous devons identifier les modes de la densité de X_k'. Pour effectuer cet étape, nous utilisons un estimateur de densité à noyaux :

$$\hat{f}_k(t) = \frac{1}{h|X_k'|} \sum_{x' \in X_k'} K\left(\frac{t - x'}{h}\right)$$

où $K(u) = (1/\sqrt{2\pi})e^{-u^2/2}$ est un noyau gaussien (Silverman (1986)). Nous avons choisi le noyau gaussien car il est le plus répandu et donne de bons résultats : notre but est simplement de disposer d'une fonction plus lisse que le simple histogramme pour la recherche de modes. Il y a plusieurs manières de choisir la bande passante h de l'estimateur ; nous prenons la plus répandue, la "rule-of-thumb" de Silverman (1986) : $h = (1.06 \cdot \hat{\sigma}|X_k'|)$, où $\hat{\sigma}$ est la variance empirique de X_k'.

La seconde étape consiste à identifier les modes de cette densité. Pour cela, nous devons identifier les minimums locaux de $\hat{f}$. Un minimum local est un point où la dérivée $\hat{f}'(t)$ est nulle et la dérivée seconde $\hat{f}''(t)$ est positive. En pratique, il est suffisant de parcourir la droite et d'identifier les zones où la dérivée devient tour à tour négative, nulle puis positive. Nous

faisons un parcours linéaire de la densité (en discrétisant avec un pas δ) des valeurs de $\hat{f}$. La dérivée en un point $y = i \cdot \delta$ est approchée par $\frac{\hat{f}(y)-\hat{f}(y-\delta)}{\delta}$. L'Algorithme 1 décrit en pseudo-code la recherche de modes.

Entrées : $X_k^{'} = (x_1^{'}, \ldots, x_p^{'})$, δ
Sorties : Modes correspondants $M = (m_1, \ldots, m_p)$

$\hat{f} \leftarrow$ estimateur à noyaux de $X_k^{'}$
$x \leftarrow \min(X_k^{'})$
$M = \{\},\ m = 1$
tant que $x \leq max(X_k^{'})$ **faire**
  si *ChangementSigne(i-1,i,i+1)* **alors**
    $m \leftarrow m + 1$
  fin
  $M \leftarrow (M, m)$
  $x \leftarrow x + \delta$
fin

Algorithme 1 : Recherche de modes de la densité de $X_k^{'}$

2.5 Clustering final

Après les étapes précédentes, on a identifié les modes sur chacune des droites $D_1, \ldots, D_M$. On rassemble cette information de façon synthétique dans une matrice X^{modes} (de N lignes et M colonnes) telle que

$$x_{ik}^{modes} = \begin{cases} \text{mode}_k(x_i) & \text{si} \quad x \in X_k \\ 0 & \text{sinon} \end{cases}$$

où $\text{mode}_k(x_i)$ est le numéro du mode contenant la projection de x sur D_k. La i-ème ligne de X^{modes} indique donc dans quels modes et sur quelles droites se projette $x_i \in X$.

Pour terminer le clustering, nous devons maintenant effectuer la synthèse des clusterings individuels. Nous avons choisi la méthode la plus simple : on effectue un clustering agglomératif de type average-link (Duda et al. (2000)) sur les lignes de la matrice X^{modes}. La mesure de distance entre deux lignes doit refléter l'intuition suivante : deux points sont "proches" si ils sont projetés dans les mêmes modes sur les mêmes droites. En comparant deux lignes, nous devons donc ignorer les colonnes où les deux lignes ont des composantes nulles, c'est-à-dire les droites où aucun des deux points ne sont projetés. Sur chacune des droites restantes, si les deux points sont dans des modes différents, cela contribue à les éloigner. Cela nous amène naturellement à la distance de Jaccard qui est simplement la proportion de coordonnées non

nulles qui diffèrent :

$$d_{Jaccard}(x_i, x_j) = \frac{|\{m|x_{im} \neq x_{jm}\}|}{|\{x_{im} \neq 0 \quad \text{ou} \quad x_{jm} \neq 0\}|}$$

2.6 Complexité

Effectuer les projections sur les M droites a une complexité de $O(MNd)$. Calculer les distances entre les points et les droites est $O(MNd)$ également. Trier les distances aux droites pour chacun des points a une complexité totale de $O(N \cdot M\log M)$ dans le pire cas. La recherche des modes sur les droites a une complexité de $O(MNc)$ dans le pire cas, où c correspond au nombre de points discrétisés $\left(\frac{\max(X_k^{'})-\min(X_k^{'})}{\delta} \quad \text{pour chaque } k\right)$. En pratique on choisit δ en fonction du diamètre $\left(\max(X_k^{'}) - \min(X_k^{'})\right)$ pour obtenir un c constant ($c \approx 100$ par exemple). Finalement, la procédure d'agglomération a une complexité de $O(N_u^2 \log N_u)$ dans le pire cas, où N_u est le nombre total de lignes distinctes dans la matrice X_{modes}. Pour résumer, l'étape de projection et de recherche de modes est linéaire avec le nombre de points N et la dimension d, et l'étape d'agglomération est au pire de $O(N^2 \log N)$.

3 Expériences

3.1 Critères d'évaluation

L'évaluation de la qualité du clustering est une question délicate. Dans cet article, nous disposons pour chaque jeu de données de la "vraie" classe $Y = \{y_1, \ldots, y_n\} \in [1, n_Y]$. Dans ce cas, évaluer un clustering revient à comparer un clustering $Z = \{z_1, \ldots, z_N\} \in [1, n_Z]$ trouvé par l'algorithme, et les "vraies" classes dans Y. Il existe de nombreux critères de comparaison. Nous avons retenu deux critères classiques et complémentaires.

Il arrive bien souvent, en particulier avec des données réelles, que les clusters trouvés par l'algorithme contiennent des points venant de classes différentes. Un premier critère classique pour évaluer un clustering est la pureté. On la calcule de la façon suivante : on étiquette chaque cluster $C \subset Z$ par la classe dominante qu'il contient $\mathrm{cldom_Y(C)} \in [1, n_Y]$. La pureté d'un cluster $\mathrm{PUR}_Y(c)$ est

$$\mathrm{PUR}_Y(C) = \frac{|\{c \in C | c = \mathrm{cldom}_Y(c)\}|}{|C|}$$

La pureté $\mathrm{PUR}_Y(Z)$ de tout le clustering Z est simplement la moyenne pondérée par la taille des clusters

$$\mathrm{PUR}_Y(Z) = \sum_{cluster\ C} \frac{|C|}{N} \mathrm{PUR}_Y(C)$$

Le second critère est l'information mutuelle normalisée (IMN) (voir Strehl et Ghosh (2002)) : soit $\mathrm{IM}(Y, Z) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{p_i p_j}$ l'information mutuelle entre les deux variables aléatoires Y et Z ; alors l'IMN est

$$\mathrm{IMN}(Y, Z) = \frac{\mathrm{IM}(Y, Z)}{\sqrt{H(Y)H(Z)}}$$

avec $H(\cdot)$ dénotant l'entropie discrète. Ce critère est au maximum de 1 quand les deux étiquetages sont parfaitement semblables. Si chaque classe de Y a un cluster correspondant dans Z, à part l'une d'elles divisée en deux clusters dans Z, alors l'IMN sera plus petite que 1. Ainsi, le critère pénalise un clustering qui n'a pas la même structure que la "vraie" information de classe. Ceci permet de juger à quel point les deux étiquetages ont la même structure.

3.2 Jeux de données

Nous avons pris trois jeux de données répandus pour l'évaluation des algorithmes. Le premier est *CHART* [1] : il s'agit de séries temporelles de contrôle. Il comporte $N = 600$ exemples en dimension $d = 60$. Il contient $c = 6$ sortes de séries temporelles différentes. Le second est *USPS* [2] (US Postal Service). Il s'agit d'une base de $N = 7291$ caractères manuscrits de dimension $d = 256$ (16 pixels sur 16). Il contient $c = 10$ classes (les caractères manuscrits entre 0 et 9). Le troisième jeu *Coil20* [3] (Columbia University Image Library) contient $N = 1440$ images de dimensions $d = 1024$ (32 pixels par 32). Ce jeu de données est constitué de $c = 20$ objets qui sont pris chacun en photo selon 72 angles différents. Les angles correspondent à des rotations successives de 5 degrés des objets sur une table tournante.

3.3 Algorithmes et paramètres

Pour chacun des jeux de données, nous avons évalué la performance de quatre méthodes :

- CLIP (Clustering par Lignes Inter-Points) : notre méthode ;
- RP : la méthode de Topchy et al. (2003) ;
- ACP+EM : mixture de gaussiennes, précédée d'une ACP gardant 85% de la variance des données ;
- LAC : l'algorithme de subspace clustering de Domeniconi et al. (2007) qui est un K-means qui pondère les dimensions pour localiser le sous-espace particulier de chaque cluster.

Pour chaque méthode, nous avons effectué le clustering en paramétrant un nombre de clusters k variable. Cela permet de mieux se rendre compte de l'efficacité de l'algorithme (voir Fern et Brodley (2003)), sachant qu'il est rare que le nombre de classes dans le jeu de données corresponde au nombre de clusters du point de vue géométrique du terme. Chaque expérience, pour un algorithme et un jeu de données fixés, a été effectuée 10 fois (avec des graines aléatoires différentes pour CLIP et RP, et des initialisations différentes pour ACP+EM et LAC). Les tables de résultats affichent la moyenne et les écarts-types de la performance selon les deux critères de Pureté et d'Information Mutuelle Normalisée décrits plus haut. Les résultats de notre méthode CLIP ont été obtenus avec les paramètres suivants : pour *CHART*, nous avons pris $M = 100$ droites et projeté chaque point sur ses $m = 10$ droites les plus proches. Pour *USPS*, on a pris $M = 400$ et $m = 10$. Pour *Coil20*, on a pris $M = 500$ et $m = 10$. On a évalué la méthode RP avec le même nombre M de droites que CLIP.

[1] http ://kdd.ics.uci.edu/databases/synthetic_control/synthetic_control.html
[2] http ://cervisia.org/machine_learning_data.php
[3] http ://www1.cs.columbia.edu/CAVE/software/softlib/coil-20.php

k	6	8	10	12
		Pureté		
CLIP	**0.7363 ± 0.0399**	**0.8540 ± 0.0531**	**0.8943 ± 0.0492**	**0.9297 ± 0.0451**
RP	0.5320 ± 0.0111	0.5437 ± 0.0223	0.5547 ± 0.0229	0.5737 ± 0.0171
ACP+EM	0.7112 ± 0.0558	0.7872 ± 0.0464	0.7988 ± 0.0410	0.8293 ± 0.0350
LAC	0.6928 ± 0.1073	0.7895 ± 0.0479	0.8295 ± 0.0695	0.8247 ± 0.0715
		Information Mutuelle Normalisée		
CLIP	**0.8191 ± 0.0068**	**0.8255 ± 0.0373**	**0.8209 ± 0.0295**	**0.8170 ± 0.0319**
RP	0.6967 ± 0.0338	0.6927 ± 0.0330	0.6887 ± 0.0365	0.6831 ± 0.0349
ACP+EM	0.7350 ± 0.0480	0.7403 ± 0.0251	0.7351 ± 0.0283	0.7272 ± 0.0243
LAC	0.7109 ± 0.1025	0.7349 ± 0.0384	0.7565 ± 0.0532	0.7360 ± 0.0574

TAB. 1 – *Résultats pour le jeu de données CHART*

k	10	12	14	16
		Pureté		
CLIP	**0.8333 ± 0.0401**	**0.8784 ± 0.0215**	**0.8797 ± 0.0217**	**0.8933 ± 0.0080**
RP	0.5848 ± 0.0656	0.6045 ± 0.0484	0.6233 ± 0.0369	0.6501 ± 0.0565
ACP+EM	0.7593 ± 0.0237	0.7993 ± 0.0223	0.8284 ± 0.0215	0.8462 ± 0.0270
LAC	0.7290 ± 0.0225	0.7489 ± 0.0203	0.7551 ± 0.0311	0.7875 ± 0.0222
		Information Mutuelle Normalisée		
CLIP	**0.7916 ± 0.0164**	**0.7843 ± 0.0081**	**0.7822 ± 0.0069**	**0.7793 ± 0.0078**
RP	0.5806 ± 0.0635	0.5861 ± 0.0595	0.5914 ± 0.0505	0.5985 ± 0.0548
ACP+EM	0.7210 ± 0.0147	0.7328 ± 0.0148	0.7402 ± 0.0139	0.7450 ± 0.0176
LAC	0.6225 ± 0.0117	0.6307 ± 0.0133	0.6351 ± 0.0231	0.6492 ± 0.0110

TAB. 2 – *Résultats pour le jeu de données USPS*

k	20	23	26	29
		Pureté		
CLIP	**0.8186 ± 0.0218**	**0.8285 ± 0.0143**	**0.8308 ± 0.0175**	**0.8382 ± 0.0179**
RP	0.4601 ± 0.0484	0.4921 ± 0.0412	0.5210 ± 0.0339	0.5350 ± 0.0290
ACP+EM	0.6802 ± 0.0431	0.7276 ± 0.0319	0.7436 ± 0.0412	0.7605 ± 0.0244
LAC	0.2257 ± 0.0378	0.3338 ± 0.0752	0.4031 ± 0.0832	0.4254 ± 0.0805
		Information Mutuelle Normalisée		
CLIP	**0.8716 ± 0.0125**	**0.8655 ± 0.0077**	**0.8534 ± 0.0097**	**0.8456 ± 0.0107**
RP	0.6746 ± 0.0275	0.6849 ± 0.0243	0.6986 ± 0.0267	0.7069 ± 0.0227
ACP+EM	0.7860 ± 0.0225	0.7995 ± 0.0183	0.8077 ± 0.0183	0.8048 ± 0.0173
LAC	0.4579 ± 0.0694	0.3338 ± 0.0752	0.4031 ± 0.0832	0.4254 ± 0.0805

TAB. 3 – *Résultats pour le jeu de données Coil20*

3.4 Résultats

Pour chacun des jeux de données, les expériences montrent que notre approche CLIP donne des résultats sensiblement supérieurs aux autres approches. Ces bons résultats sont assez stables pour un nombre de clusters k différents, ce qui confirme la robustesse de la méthode. Les mauvais résultats de RP montrent quant à eux que des droites choisies complètement aléatoirement et indépendamment des données ne permettent pas d'obtenir des clusterings individuels intéressants. (Nous avons également testé RP avec des droites aléatoires non contraintes à passer par l'origine, ce qui a donné des résultats légèrement moins bons que ceux de RP). L'algorithme classique de clustering par mixture de gaussiennes (précédée d'une ACP) est le deuxième meilleur après notre méthode. L'algorithme LAC vient ensuite. Les Figures 3, 4 et 5 illustrent graphiquement un clustering obtenu avec notre méthode pour chaque jeu de données.

4 Conclusion et perspectives

Nous avons proposé une nouvelle méthode dans le cadre des clustering ensembles. Chaque clustering individuel est réalisé à partir d'une projection partielle du jeu de données sur une droite reliant deux points de données pris au hasard. Nous avons évalué et comparé la méthode avec d'autres approches ; les expériences montrent que notre méthode donne de meilleurs résultats. Les perspectives de travail futur portent sur deux aspects : le mécanisme de sélection des droites de projection pour chaque point, et le clustering global. Pour qu'un point influence les droites les plus proches de lui, on peut imaginer d'utiliser tous les points pour chaque droite, mais en pondérant l'influence de chaque point dans l'estimateur de densité en fonction de sa distance à la droite. Cela permettrait de s'affranchir du calcul des distances entre points et droites. Quant à la performance du clustering global, elle est très probablement due à la nouvelle représentation des données et non pas à la nature agglomérative de l'algorithme de synthèse des clusterings individuels. Une variante de K-means avec la distance de Jaccard devrait donner des résultats similaires pour une complexité moindre.

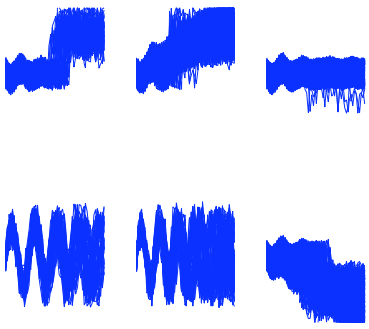

FIG. 3 – *Résultats du clustering $k = 6$ pour CHART*

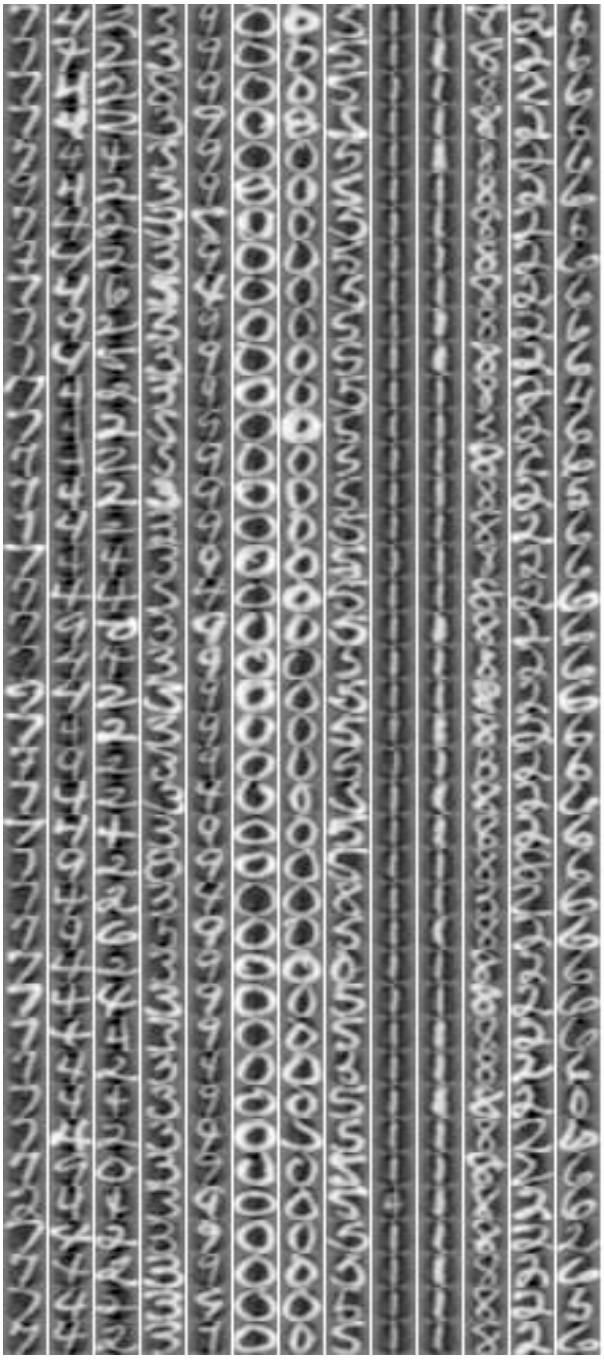

FIG. 4 – *Résultats du clustering* $k = 10$ *pour USPS*

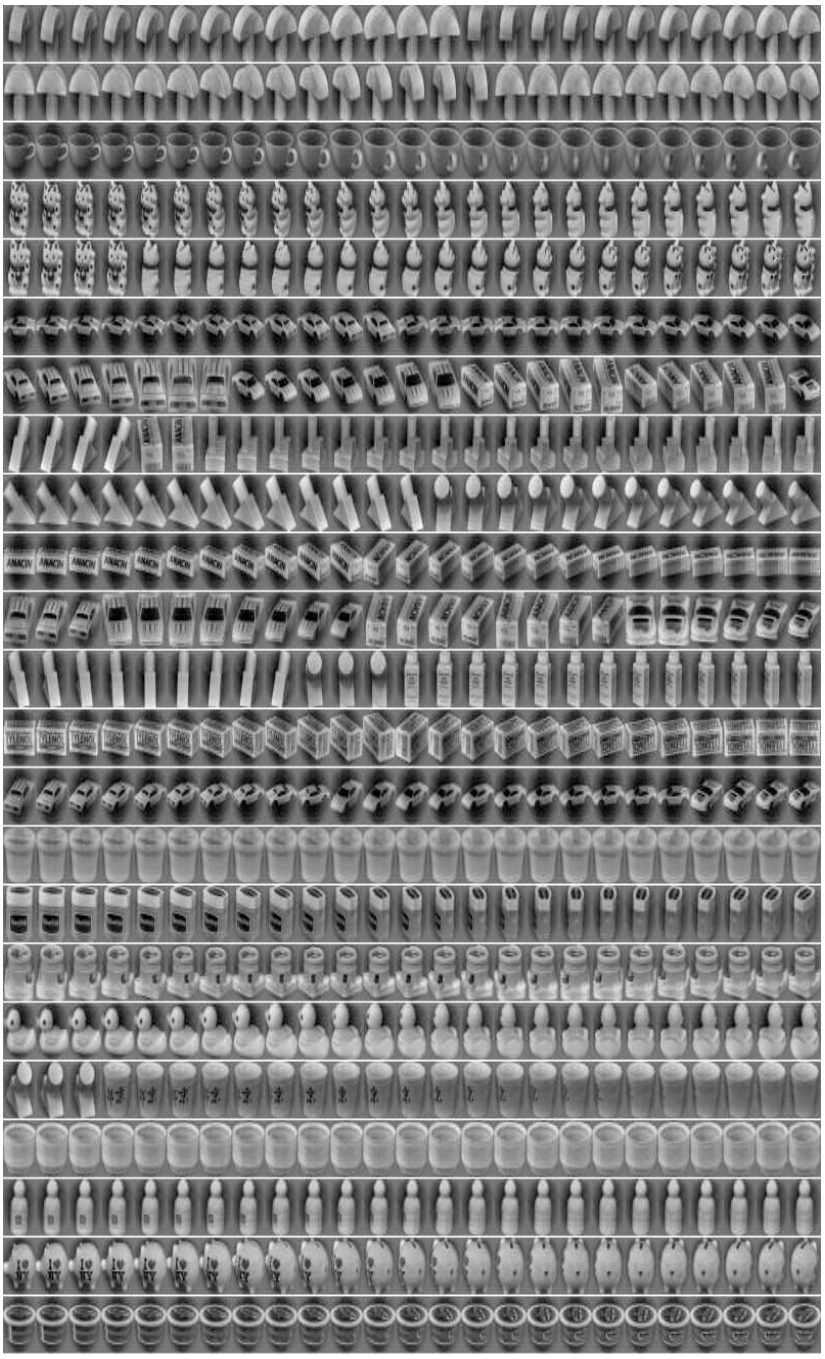

FIG. 5 – *Résultats du clustering* $k = 23$ *pour Coil20*

Références

Aggarwal, C. C. et P. S. Yu (2000). Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 70–81.

Boley, D. (1998). Principal direction divisive partitioning. *Data Mining and Knowledge Discovery 2*(4), 325–344.

Ding, C. et T. Li (2007). Adaptive dimension reduction using discriminant analysis and k-means clustering. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 521–528.

Domeniconi, C., D. Gunopoulos, S. Ma, B. Yan, M. Al-Razgan, et D. Papadopoulos (2007). Locally adaptive metrics for clustering high dimensional data. *Data Mining and Knowledge Discovery 14*(1), 63–97.

Duda, R. O., P. E. Hart, et D. G. Stork (2000). *Pattern Classification*. Wiley-Interscience Publication.

Fern, X. et C. Brodley (2003). Random projection for high dimensional data clustering : A cluster ensemble approach. In *Proceedings of the Twentieth International Conference on Machine Learning*, pp. 186–193.

Miasnikov, A., J. Rome, et R. Haralick (2004). A hierarchical projection pursuit algorithm. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04)*.

Parsons, L., E. Haque, et H. Liu (2004). Subspace clustering for high dimensional data : A review. *SIGKDD Explorations, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining 6*, 186–193.

Rui, X. et D. Wunsch (2005). Survey of clustering algorithms. In *IEEE Transactions on Neural Networks*, Volume 16, pp. 645–678.

Silverman, B. (1986). *Density Estimation in Statistics and Data Analysis*. Chapman and Hall CRC.

Strehl, A. et J. Ghosh (2002). Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research (JMLR) 3*, 583–617.

Topchy, A., A. K. Jain, et W. Punch (2003). Combining multiple weak clusterings. In *Third IEEE International Conference on Data Mining (ICDM 2003)*, pp. 331–338.

Summary

Clustering is an essential base block for the data mining task. Recently, the so-called "cluster ensembles" methodology attracted much attention. The principle is to aggregate many clusterings of a dataset in order to obtain an "average clustering. The individual clusterings can be the output of different clustering algorithms. This methodology is particularly useful when the data dimensionality hinders methods based on distance and/or density from giving good results. In this article, we propose a new way of obtaining individual clusterings at low cost, from partial projections of the dataset. We evaluate empirically our method and compare it to three methods of different types. Our method obtains noticeably superior results.