Post-traitement pour la classification probabiliste non supervisée sous contraintes

Nguyen-Viet-Dung Nghiem *, Christel Vrain*
Thi-Bich-Hanh Dao* Ian Davidson**

*Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022, F-45067, Orléans, France
prenom.nom@univ-orleans.fr
**Department of Computer Science, University of California, Davis, USA
davidson@cs.ucdavis.edu

Résumé. Le clustering sous contraintes permet d'intégrer des connaissances expertes, que ce soit des contraintes classiques must-link ou cannot-link ou des contraintes plus complexes. La plupart des algorithmes de clustering probabilistes intègrent les contraintes en ajoutant un terme dans la fonction objectif représentant leur satisfaction. Ils intègrent donc difficilement différents types de contraintes et ne garantissent pas la satisfaction de toutes les contraintes. Nous proposons une méthode qui, à partir du résultat d'un algorithme de clustering sous la forme d'une matrice de probabilité d'affectation de points aux clusters, trouve la meilleure affectation qui satisfasse toutes les contraintes. Cette méthode peut s'appliquer à tout algorithme probabiliste y compris ceux utilisant l'apprentissage profond. Les expérimentations montrent que notre méthode est compétitive avec des méthodes parmi les plus récentes.

1 Introduction

Le clustering semi-supervisé est un domaine déjà longuement étudié. Généralement, il consiste à introduire des connaissances initiales sur les étiquettes des points sous la forme de contraintes *must-link* ou *cannot-link* entre paires de points spécifiant si ces points doivent apparaître ou non dans le même cluster (Wagstaff et Cardie, 2000; Wagstaff et al., 2001; Bilenko et al., 2004; Davidson et Ravi, 2005; Wang et Davidson, 2010). Intégrer des connaissances plus générales dans un processus de clustering nécessite souvent d'adapter les algorithmes existants, conduisant à des systèmes dédiés à certains types de contraintes. Récemment il a été montré que des formalismes déclaratifs comme la Programmation Linéaire en Nombres Entiers (PLNE) ou la Programmation par Contraintes permettaient d'intégrer facilement des connaissances de formes variées, cependant au prix de la complexité (Davidson et al., 2010; Babaki et al., 2014; Ouali et al., 2016; Dao et al., 2017). De plus, comme la plupart des approches en clustering, elles reposent sur une distance calculée dans l'espace des entrées, et l'on sait que l'utilisation d'une distance en grande dimension pose de nombreux problèmes.

Ces dernières années, les avancées en apprentissage profond permettent de transformer les données d'entrée dans des espaces de dimension plus faible grâce à des plongements non linéaires. Cela a conduit au développement du *deep clustering* (Song et al., 2013; Xie et al.,

2016; Guo et al., 2017) et à l'émergence de quelques méthodes pour intégrer des contraintes (Zhang et al., 2019; Ren et al., 2019). A ces fins, la fonction de perte est en général adaptée, avec l'introduction d'un second critère mesurant le taux de satisfaction des contraintes. Cela nécessite de modifier le processus d'optimisation et rend les systèmes dépendants du type de contraintes considéré. De plus, le poids donné à la satisfaction des contraintes doit être minutieusement choisi : un poids trop important risque de conduire à une solution très mauvaise du point de vue de la partition engendrée et à l'inverse un poids trop faible revient à ne pas considérer les contraintes. Enfin ces systèmes ne garantissent pas la satisfaction de toutes les contraintes. Ce dernier point pourrait sembler un avantage dans le cas de contraintes bruitées, mais il n'est pas possible de contrôler les contraintes qui seront réalisées, même si certaines sont des contraintes dures qui doivent être satisfaites. L'introduction de nouvelles contraintes par un expert au vu des résultats produits nécessite de relancer le processus d'apprentissage avec aucune garantie de construire une partition proche de la précédente.

Dans ce papier nous proposons un post-traitement des résultats d'un clustering pour forcer la réalisation de contraintes. On peut envisager deux cas d'usage : l'algorithme de clustering n'a pas satisfait toutes les contraintes ou de nouvelles contraintes ont été introduites et il ne paraît pas souhaitable de réapprendre le modèle. Nous nous plaçons dans un cadre probabiliste du clustering où chaque point a une probabilité d'appartenance à un cluster et nous formalisons le problème comme la recherche d'une partition satisfaisant toutes les contraintes et maximisant la probabilité d'affection des points aux clusters. Notre approche est modélisée en PLNE et a plusieurs avantages :

— différents types de contraintes peuvent être intégrés sans changer la fonction de perte dans le cœur du système d'apprentissage profond,
— la méthode est efficace permettant de traiter de grandes bases de données,
— la satisfaction des contraintes est garantie,
— la méthode prend en entrées une matrice représentant les probabilités d'affectations des points aux clusters et peut en conséquence être utilisée avec tout algorithme de clustering produisant une telle matrice, algorithme d'apprentissage profond ou EM par exemple.

2 Travaux connexes

Le clustering sous contraintes permet d'intégrer des connaissances expertes, sous forme de contraintes, dans un processus de clustering. Ces connaissances peuvent être des informations sur l'étiquette des objets, permettant ainsi d'engendrer des contraintes d'instances, must-link ou cannot-link, suivant que les objets doivent être ou non dans le même cluster. Ces contraintes d'instances sont les plus populaires et de nombreuses méthodes en clustering sous contraintes ont été développées pour les intégrer (Wagstaff et Cardie, 2000; Wagstaff et al., 2001; Bilenko et al., 2004; Davidson et Ravi, 2005; Wang et Davidson, 2010). Mais les connaissances peuvent aussi porter sur les clusters et plusieurs méthodes ont été développées pour intégrer de telles contraintes : contraintes sur la taille minimale des clusters (Bradley et al., 2000), nécessité d'avoir des clusters équilibrés en taille (Ge et al., 2007; Tang et al., 2019), contraintes sur la densité des clusters (Davidson et Ravi, 2005), ...Comme on peut le voir, ces méthodes sont adaptées à un type de contraintes spécifique. Ces dix dernières années, des méthodes plus générales ont été développées, elles proposent un cadre générique en se fondant sur des formalismes déclaratifs, comme la programmation linéaire en nombre entiers (Babaki et al.,

2014; Mueller et Kramer, 2010; Ouali et al., 2016), SAT (Davidson et al., 2010; Métivier et al., 2012) ou la programmation par contraintes (Dao et al., 2016, 2017). Ces méthodes souffrent cependant du problème de passage à l'échelle et ne peuvent pas traiter de grandes bases de données.

Les méthodes précédentes considèrent les données dans l'espace initial, ce qui peut rendre le clustering inapproprié lorsque la dimension est grande. Une solution est de changer la représentation en un espace de plus faible dimension, puis d'appliquer le clustering dans ce nouvel espace. L'apprentissage profond a permis d'envisager de nouvelles méthodes de clustering, regroupées sous le terme de *deep clustering*, intégrant des mécanismes d'apprentissage d'espaces de représentation. On peut distinguer deux familles suivant la manière dont l'apprentissage du changement de représentation est intégré dans le système : (i) apprendre une nouvelle représentation et appliquer une méthode de clustering, (ii) optimiser simultanément la représentation et la qualité du clustering. Nous considérons dans ce papier cette dernière catégorie. Song et al. (2013) intègre dans l'auto-encodeur un objectif de clustering de telle sorte que dans la nouvelle représentation, les données soient proches du centre de leur cluster. Xie et al. (2016) propose une méthode de clustering profond (DEC pour Deep Embedded Clustering), qui simultanément apprend l'espace de représentation et l'affectation des objets aux clusters. Cette méthode a été améliorée dans la méthode IDEC (Improved Deep Embedded Clustering), qui intègre également un critère de préservation de la structure locale (Guo et al., 2017).

Les méthodes DEC/IDEC ont été étendues pour introduire des connaissances utilisateur. Elles sont modélisées sous forme de contraintes et intégrées par le biais d'un nouveau terme ajouté dans la fonction objectif et représentant la satisfaction de ces contraintes. La fonction objectif est donc une combinaison linéaire de termes représentant la qualité du clustering, la qualité de la reconstruction par l'auto-encodeur et la satisfaction des contraintes. Les contraintes considérées sont des contraintes d'instances must-link/cannot-link (Ren et al., 2019), des contraintes sur des triplets ou des contraintes demandant des clusters de taille équilibrée (Zhang et al., 2019). Ces méthodes ne garantissent pas la satisfaction des contraintes puisqu'il ne s'agit que d'un terme dans la fonction objectif. De plus, en présence de nouvelles contraintes, le modèle doit être de nouveau appris, ce qui peut devenir très coûteux.

L'approche proposée dans ce papier est radicalement différente puisqu'elle propose de traiter les contraintes a posteriori. Etant donnée la sortie d'une méthode de clustering, donnant pour chaque point et chaque cluster, la probabilité d'affecter ce point à ce cluster, notre méthode cherche la meilleure affectation satisfaisant toutes les contraintes. Elle peut intégrer différents types de contraintes sans nécessité de réapprendre le modèle et peut être utilisée avec toute méthode de clustering probabiliste, dont le clustering profond.

3 Préliminaires

Etant donné un ensemble de N objets (points) $\{x_i \in X\}_{i=1}^{N}$, un algorithme de clustering cherche à partitionner les objets en K groupes (clusters) disjoints, tels que les objets d'un même cluster soient similaires et les objets de différents clusters dissimilaires. Les clusters sont souvent représentés par leurs centroïdes $\{\mu_j\}_{j=1}^{K}$, et de nombreux algorithmes de clustering, comme par exemple k-means, cherchent simultanément les centroïdes et l'affectation des points aux clusters.

3.1 Clustering avec des connaissances utilisateur

Les connaissances utilisateur sont par nature diverses. Comme mentionné précédemment, elles peuvent être issues d'étiquettes déjà connues sur un sous-ensemble d'objets, et souvent traduites sous forme de contraintes d'instances must-link ou cannot-link, qui respectivement indiquent si deux objets doivent être ou non dans le même cluster. Mais les connaissances données par un expert du domaine peuvent être plus complexes, au delà des contraintes d'instances comme illustré ci-dessous.

— Contrainte relative (contrainte *triplet*) $ab|c$, qui indique que a et b sont plus proches l'un de l'autre que de c, i.e. si a, c ou b, c sont dans le même cluster, alors a, b, c doivent être ensemble (Liu et al., 2011). Ce sont des connaissances plus faciles à exprimer pour un expert que des connaissances must-link ou cannot-link.

— Contrainte sur la taille des clusters, qui précise une taille minimale ou maximale des clusters. La contrainte de clustering équilibré exigeant que les clusters soient de taille équilibrée est un cas légèrement différent, puisque la satisfaire nécessite de considérer l'ensemble des clusters.

— Contrainte sur la taille de sous-parties des clusters, spécifiées par une propriété donnée, indiquant des valeurs minimales ou maximales sur le nombre d'éléments satisfaisant cette propriété dans chaque cluster (par exemple le nombre de femmes dans un groupe de personnes). Ce type de contraintes peut exprimer par exemple que le nombre d'éléments ayant cette propriété soit équilibré entre clusters.

— Contrainte géométrique, qui indique une valeur minimale/maximale sur le diamètre des clusters ou sur la marge entre les clusters. Une contrainte sur le diamètre peut s'exprimer par des contraintes cannot-link alors qu'une contrainte sur la marge peut s'exprimer par des contraintes must-link.

— Contrainte de couverture basée sur des attributs, qui restreint le nombre de clusters pouvant contenir des éléments d'une certaine propriété. Par exemple considérant des textes, un expert peut savoir que l'ensemble des documents contenant un ensemble de mots donnés peut être couvert par au plus p clusters.

Satisfaire tous ces types de contraintes est un problème NP-Difficile (Davidson et Ravi, 2005). A titre d'exemple, le problème de k-coloration de graphe, qui est NP-Complet, peut être reformulé par des contraintes cannot-link.

3.2 Clustering profond sous contraintes

Le clustering profond exploite des réseaux de neurones profonds pour apprendre une fonction non linéaire permettant d'engendrer une nouvelle représentation $f : X \to Z$, puis simultanément apprendre les centroides $\{\mu_j\}_{j=1}^{K}$ dans le nouvel espace Z et améliorer la fonction f.

Deep Embedded Clustering (DEC) (Xie et al., 2016) entraîne un auto-encodeur ($x_i = g(f(x_i))$), puis retire le décodeur, pour ne garder que l'encodeur ($z_i = f(x_i)$). Cet encodeur est ensuite peaufiné en optimisant une fonction objectif définissant la qualité du clustering :

$$L_c = KL(P||Q) = \sum_i \sum_k p_{ik} \log \frac{p_{ik}}{q_{ik}}$$

Plus précisément, k-means est appliqué sur l'espace des z_i pour chercher les centres μ_j, puis une matrice d'allocation (q_{ik}) est calculé, où q_{ik} donne la probabilité d'appartenance de l'élément i au cluster k :

$$q_{ik} = \frac{(1 + ||z_i - \mu_k||^2)^{-1}}{\sum_{k'}(1 + ||z_i - \mu_{k'}||^2)^{-1}}$$

Enfin la distribution cible (p_{ik}) est calculée de manière à amplifier la séparation entre les clusters :

$$p_{ik} = \frac{q_{ik}^2 / \sum_{i'} q_{i'k}}{\sum_{k'} q_{ik'}^2 / \sum_{i'} q_{i'k'}}$$

La fonction f et les centres des clusters μ_j sont mis à jour par rétro-propagation, cherchant à minimiser la différence entre q la matrice d'allocation obtenue et p une matrice plus "idéale" que q au sens du clustering (self-entraînement).

Improved Deep Embedded Clustering (IDEC) (Guo et al., 2017) préserve la structure locale de données en gardant le décodeur et en ajoutant un critère de qualité de la reconstruction $L_r = \sum_i ||x_i - g(z_i)||_2^2$. La fonction objectif est alors définie comme une combinaison linéaire de ces deux termes $L = L_r + \gamma L_c$.

Deep Constrained Clustering (Zhang et al., 2019) intègre quatre types de contraintes : contraintes d'instances, contraintes indiquant les instances difficiles, contraintes sur des triplets et contraintes de clusters équilibrés en taille. Pour chaque type de contraintes, un terme est défini pour représenter la satisfaction des contraintes. La fonction objectif finale est définie comme une combinaison de la qualité du clustering, de la qualité de reconstruction et de la satisfaction des contraintes.

Dans toutes les méthodes existantes, la partition finale est calculée de sorte que chaque point x_i soit affecté au cluster ayant la probabilité maximale $\arg\max_j q_{ij}$. Cette affectation peut ne pas satisfaire des contraintes représentant des connaissances.

4 Post-traitement de contraintes

Considérons une matrice P de taille $N \times K$, où P_{ik} indique la probabilité d'affecter le point x_i au cluster k. Cette matrice peut être le résultat d'une méthode de clustering probabiliste quelconque, comme par exemple DEC ou IDEC, avec ou sans contraintes utilisateur. Considérons un ensemble de contraintes $\mathcal{C}$. Comme expliqué dans la section 3.1, satisfaire les contraintes est un problème NP-difficile. Nous proposons dans notre approche de trouver le meilleur clustering construit à partir des probabilités P tout en satisfaisant toutes les contraintes de $\mathcal{C}$.

Le problème est alors de chercher une matrice booléenne Z de taille $N \times K$ avec comme fonction objectif

$$\arg\max \sum_{i=1}^{N} \sum_{k=1}^{K} P_{ik} Z_{ik}$$

satisfaisant les contraintes de $\mathcal{C}$. La matrice Z à valeurs dans $\{0, 1\}$ représente une affectation dure : $Z_{ik} = 1$ indique que le point x_i est affecté au cluster k. Pour assurer que chaque point soit dans un cluster, on impose $\forall i = 1, \ldots, N,\ \sum_{k=1}^{K} Z_{ik} = 1$. Les contraintes de $\mathcal{C}$ peuvent

être de différents types, comme introduit dans la section 3.1. Nous montrons ci-suit comment les différents types de contraintes peuvent être formulés.

Une contrainte must-link (resp. cannot-link) sur deux points i, j peut être formulée par $\forall k = 1, \ldots, K$, $Z_{ik} = Z_{jk}$ ($Z_{ik} + Z_{jk} \leq 1$, respectivement).

Une contrainte triplet $ab|c$ est formulée par :

$$\forall k = 1, \ldots, K,\ Z_{bk} \geq Z_{ak} + Z_{ck} - 1 \text{ et } Z_{ak} \geq Z_{bk} + Z_{ck} - 1$$

Cette formulation implique que $Z_{bk} = 1$ si $Z_{ak} = Z_{ck} = 1$ et que $Z_{ak} = 1$ si $Z_{bk} = Z_{ck} = 1$.

Une contrainte sur la taille des clusters s'exprime par :

$$\forall k = 1, \ldots, K,\ lb_k \leq \sum_{i=1}^{N} Z_{ik} \leq ub_k$$

Une contrainte sur la taille de sous-parties exprime que le nombre d'éléments possédant une propriété p dans chaque cluster est limité par des valeurs minimales lb_k et maximales ub_k. Soit $p(i) = 1$ si le point i possède la propriété p et 0 sinon. Cette contrainte est donc traduite en :

$$\forall k = 1, \ldots, K,\ lb_k \leq \sum_{i=1}^{N} p(i) Z_{ik} \leq ub_k$$

Une contrainte de couverture indique qu'un certain type d'éléments ne peut apparaître que dans un nombre limité de clusters. Soit $p(i) = 1$ si l'élément i est du type concerné et 0 sinon. Nous introduisons une variable $t_k \in \{0, 1\}$ pour chaque cluster k, telle que $t_k = 1$ si et seulement si le cluster k contient des éléments du type p. Cette relation est représentée par :

$$t_k \leq \textstyle\sum_i p(i) Z_{ik} \text{ et } \forall i = 1, \ldots, N,\ t_k \geq p(i) Z_{ik}$$
$$lb \leq \textstyle\sum_{k=1}^{K} t_k \leq ub$$

La première contrainte assure que $t_k = 0$ si aucun élément de type p est dans le cluster k. La seconde met $t_k = 1$ dès qu'un élément du type p est dans le cluster k. Les limites sur le nombre de ces clusters peuvent être alors imposées par la troisième contrainte : $lb \leq \sum_{k=1}^{K} t_k \leq ub$.

5 Expérimentations

5.1 Cadre des expérimentations

Données et clustering initial. Nous utilisons les mêmes bases de données que DCC (Zhang et al., 2019).

MNIST : Cette base est composée de 60.000 chiffres écrits à la main, représentés par des images de taille 28×28 pixels.

FASHION-MNIST : 60000 images dont les classes sont des entiers de 0 à 9.

REUTERS-10K : La base Reuters contient environ 810000 documents en anglais étiquetés par des catégories organisées en une arborescence (Lewis et al., 2004). Suivant la démarche expérimentale de DEC (Xie et al., 2016), nous considérons seulement les documents appartenant aux catégories `corporate/industrial`, `government/social`, `markets` et `economics`. Un échantillon de 10000 documents est choisi aléatoirement et la mesure TF-IDF est calculée sur les 2000 mots les plus fréquents.

Systèmes. Nous utilisons les configurations suivantes pour comparaison de nos résultats : **COP-Kmeans** (Wagstaff et al., 2001), utilisé sur les données initiales sans changement de représentation ; **IDEC-COP-Kmeans**, COP-Kmeans utilisé sur la représentation apprise par l'encodeur d'IDEC (Guo et al., 2017) ; **MSE-Kmeans**, une version modifiée de K-means, utilisant un calcul de flot de coût minimal pour satisfaire des contraintes sur la taille des clusters ; **IDEC-Post**, notre méthode appliquée au clustering probabiliste trouvé par IDEC ; **DCC**, Deep Constrained Clustering (Zhang et al., 2019) qui intègre les contraintes pendant la procédure de clustering ; **DCC-Post**, notre méthode appliquée au clustering probabiliste trouvé par DCC.

Parmi ces systèmes, COP-Kmeans n'intègre que des contraintes must-link et cannot-link. MSE-Kmeans ne considère que des contraintes sur la taille des clusters. DCC traite 4 types de contraintes comme mentionné dans la Section 2. Les méthodes génériques comme (Dao et al., 2017) permettant d'intégrer divers types de contraintes ne peuvent traiter ces jeux de données de grande taille et ne peuvent donc être utilisées dans nos expérimentations.

Afin d'être équitable et indépendant d'IDEC, nous l'appliquons une fois et fournissons la représentation apprise à toutes les méthodes utilisant le changement de représentation (IDEC-COP-Kmeans, DCC, IDEC-Post). De plus, nous initialisons DCC avec le réseau et les paramètres appris par IDEC. Tous les algorithmes sont implantés en Python. Gurobi est utilisé comme solveur de PLNE. Les expérimentations sont réalisées sur un processeur Intel Core i7 2.8 GHz avec 16Go de RAM.

Métriques d'évaluation. Pour ces données, nous disposons des classes réelles et nous considérons donc deux mesures : Normalized Mutual Information (NMI) et Clustering Accuracy (ACC), avec une fonction d'appariement des clusters aux classes calculée par l'algorithme hongrois (Kuhn, 1955). Les indices fondés sur une distance nous semblent non pertinents ici dans la mesure où les systèmes utilisés effectuent un changement de représentation.

5.2 Résultats et analyses

Contraintes d'instances. Dans ces premières expérimentations, nous souhaitons étudier le comportement des systèmes avec différents nombres de contraintes. Nous considérons donc les jeux de données MNIST et Fashion avec 3600, 30000 et 60000 contraintes. Pour chaque nombre de contraintes, nous générons cinq jeux de contraintes afin de prendre en compte la variabilité des contraintes. Pour chaque métrique, le résultat présenté est la moyenne et l'écart type sur les cinq jeux de contraintes.

Les tables 1 et 2 présentent les résultats pour MNIST et Fashion. En plus de NMI et ACC, nous présentons aussi le nombre de contraintes qui ne sont pas satisfaites, la différence en nombre de points modifiés entre IDEC-Post et IDEC, DCC et IDEC et DCC-Post et DCC.

Plusieurs points intéressants peuvent être observés dans ces deux tables. Tout d'abord, ils confirment qu'une représentation profonde peut améliorer le clustering (cf COP-Kmeans/IDEC-COP-Kmeans). Ensuite, dans nos expérimentations et avec notre méthode les contraintes d'instances améliorent toujours le clustering alors que des expérimentations menées avec d'autres systèmes avaient montré que l'ajout de contraintes pouvait dégrader les résultats. D'autres expérimentations seraient donc nécessaires pour confirmer ou non ce résultat. Finalement, notre méthode apparaît comme plus stable que DCC en nombre d'objets réaffectés, si on compare le clustering obtenu avec IDEC sans contraintes et le clustering avec contraintes (DCC/IDEC,

#Contraintes	Méthode	NMI	ACC	#Insatisfaites	Changements (%)
0	K-means	0.4908	0.5130	-	-
	IDEC	0.8539	0.8799	-	-
3600	COP-Kmeans	0.4904 ± 0.0053	0.5350 ± 0.0134	0	-
	IDEC-COP-Kmeans	0.8237 ± 0.0324	0.7372 ± 0.0630	0	-
	IDEC-Post	0.8547 ± 0.0002	0.8804 ± 0.0001	0	0.1917 ± 0.0084
	DCC	0.8637 ± 0.0012	0.8938 ± 0.0075	32.2000 ± 4.8332	2.6360 ± 0.5206
	DCC-Post	0.8640 ± 0.0013	0.8940 ± 0.0077	0	0.0533 ± 0.0079
30000	COP-Kmeans	0.5091 ± 0.0177	0.5837 ± 0.0405	0	-
	IDEC-COP-Kmeans	0.8477 ± 0.0195	0.8302 ± 0.0314	0	-
	IDEC-Post	0.8602 ± 0.0007	0.8839 ± 0.0005	0	1.4647 ± 0.0358
	DCC	0.9407 ± 0.0032	0.9786 ± 0.0013	68.0000 ± 13.7550	10.6480 ± 0.1762
	DCC-Post	0.9429 ± 0.0026	0.9796 ± 0.0011	0	0.1103 ± 0.0218
60000	COP-Kmeans	0.4856 ± 0.0144	0.5713 ± 0.0244	0	-
	IDEC-COP-Kmeans	0.8146 ± 0.0319	0.8039 ± 0.0644	0	-
	IDEC-Post	0.8668 ± 0.0005	0.8887 ± 0.0004	0	2.7600 ± 0.0690
	DCC	0.9549 ± 0.0029	0.9847 ± 0.0012	100.6000 ± 27.8898	11.2650 ± 0.0719
	DCC-Post	0.9581 ± 0.0021	0.9860 ± 0.0009	0	0.1543 ± 0.0391

TAB. 1 – *Résultats avec des contraintes d'instances sur MNIST*

#contraintes	Méthode	NMI	ACC	#Insatisfaites	Changements (%)
0	K-means	0.4738	0.5118	-	-
	IDEC	0.6320	0.5879	-	-
3600	COP-Kmeans	0.5124 ± 0.0048	0.5182 ± 0.0321	0	-
	IDEC-COP-Kmeans	0.6222 ± 0.0152	0.5808 ± 0.0092	0	-
	IDEC-Post	0.6315 ± 0.0003	0.5880 ± 0.0002	0	0.6013 ± 0.0244
	DCC	0.6403 ± 0.0192	0.6378 ± 0.0277	55.2000 ± 61.2614	30.1743 ± 5.5655
	DCC-Post	0.6402 ± 0.0191	0.6377 ± 0.0276	0	0.0933 ± 0.1039
30000	COP-Kmeans	0.5025 ± 0.0177	0.5513 ± 0.0234	0	-
	IDEC-COP-Kmeans	0.6175 ± 0.0043	0.5974 ± 0.0199	0	-
	IDEC-Post	0.6293 ± 0.0005	0.5905 ± 0.0003	0	4.5790 ± 0.0725
	DCC	0.7421 ± 0.0158	0.7989 ± 0.0279	965.2000 ± 200.6922	41.7050 ± 1.1691
	DCC-Post	0.7446 ± 0.0159	0.8019 ± 0.0282	0	1.5720 ± 0.3268
60000	COP-Kmeans	0.4933 ± 0.0150	0.5745 ± 0.0430	0	-
	IDEC-COP-Kmeans	0.6023 ± 0.0059	0.5853 ± 0.0175	0	-
	IDEC-Post	0.6276 ± 0.0008	0.5940 ± 0.0008	0	8.6477 ± 0.0994
	DCC	0.6430 ± 0.2882	0.7180 ± 0.2891	10373.2000 ± 17076.6703	47.9617 ± 17.1765
	DCC-Post	0.6624 ± 0.2705	0.7409 ± 0.2671	0	21.0657 ± 36.6327

TAB. 2 – *Résultats avec des contraintes d'instance sur Fashion*

IDEC-Post/IDEC). En effet, avec le même nombre de contraintes, notre méthode se focalise sur les points où les contraintes ne sont pas satisfaites, tandis que DCC change à la fois la représentation et les centres des clusters, ce qui implique un plus grand nombre de points concernés. DCC combine plusieurs termes dans sa fonction objectif et trouver un bon équilibre entre ces termes est difficile, entraînant un comportement instable.

Contraintes sur la taille des clusters. Nous montrons l'intérêt de notre méthode avec l'intégration d'autres types de contraintes, comme la contrainte sur la taille des clusters. A notre connaissance, seuls DCC (Zhang et al., 2019) et MSE (Tang et al., 2019) intègrent ce type de contraintes. DCC demande que les clusters soient de tailles plus ou moins identiques alors que MSE peut considérer des contraintes plus générales de taille. C'est la raison pour laquelle nous comparons notre méthode uniquement avec MSE. Pour avoir le même point de départ, nous fournissons à MSE comme à notre méthode la représentation apprise par IDEC.

Pour MNIST et Fashion, nous calculons les tailles minimale et maximale des classes et les utilisons comme limites pour les clusters. Les résultats sont montrés dans la table 3, illustrant ainsi la capacité d'intégration de ce type de contraintes à ces systèmes. Remarquons une baisse des indices NMI et ACC, qui peut être signe que les contraintes de taille imposées ne correspondent pas exactement à ces données.

Données - Méthode	NMI	ACC	Données - Méthode	NMI	ACC
MNIST - IDEC	0.8539	0.8799	Fashion - IDEC	0.6320	0.5879
MNIST - MSE	0.8536	0.8816	Fashion - MSE	0.5363	0.5387
MNIST - Post	0.8520	0.8796	Fashion - Post	0.5301	0.5425

TAB. 3 – *Résultats avec contraintes sur la taille des clusters*

Contraintes de couverture de clusters. Nous illustrons la flexibilité d'intégration de nouvelles contraintes de notre méthode, en considérant des contraintes de couverture. L'idée est que l'expert peut limiter l'étendue des éléments d'un certain type, en limitant le nombre de clusters pouvant les contenir. Par exemple, l'expert peut indiquer : *L'ensemble des documents qui contiennent un certain mot doit être couvert par moins de S groupes (clusters)*. Soit f l'attribut associé à ce mot et $act(f) = \{x|x_f > 0\}$ (l'ensemble des documents contenant ce mot). Cette contrainte est exprimée par :

$$\parallel \{C_h : \exists x \in act(f) \wedge x \in C_h\} \parallel \leq S \tag{1}$$

On peut étendre facilement la contrainte à un ensemble F de mots en posant $act(F) = \{x|\exists f \in F : x_f > 0\}$. Ce type de contraintes est complexe et les satisfaire nécessite souvent des modifications non locales.

Nous considérons la base Reuters-10K et des contraintes correspondant à 5 and 10 ensembles de 3 mots choisis aléatoirement, satisfaisant (1) avec $S = 2$. Nous rapportons les valeurs de NMI et ACC, la taille de $act(F)$, le nombre de contraintes non satisfaites par le clustering initial et le nombre de changements par rapport au clustering initial. L'effet de ces contraintes est assez important. Alors qu'une contrainte d'instance n'affecte que deux points, le nombre moyen de points affectés par une contrainte de couverture est d'environ 200. Le nombre de contraintes non satisfaites par le clustering initial est aussi élevé, environ 4,8 pour 5 contraintes et 9,8 pour 10. De même, le nombre de points réaffectés est aussi élevé. Tout ceci montre la généralité de ce type de contraintes.

#Contraintes	NMI	ACC	$\lvert act(F)\rvert$	#Insatisfaites	#Changements
0	0.5279	0.7452	-	-	-
5	0.5253 ± 0.0039	0.7474 ± 0.0070	1168.0000 ± 50.5332	4.8000 ± 0.4000	96.6000 ± 57.4895
10	0.5219 ± 0.0055	0.7499 ± 0.0088	2128.0000 ± 143.6760	9.8000 ± 0.4000	212.4000 ± 42.9679

TAB. 4 – *Résultats avec des contraintes de couverture sur Reuters*

Combinaisons de contraintes. Un intérêt de notre méthode est qu'elle permet d'intégrer différents types de contraintes. Ici nous considérons à la fois des contraintes d'instances (PW)

et des contraintes sur la taille des clusters (CS). Le nombre de contraintes d'instances est fixé à 30 000. Les tables 5 et 6 présentent les résultats sur MNIST et Fashion, respectivement.

Cas	NMI	ACC	#changements	#changt. positifs
Sans contrainte	0.8539	0.8799	-	-
CS	0.8520	0.8796	445	18
PW	0.8602 ± 0.0007	0.8839 ± 0.0005	878.8000 ± 21.4979	507.0000 ± 25.2745
PW + CS	0.8605 ± 0.0009	0.8843 ± 0.0006	1067.0000 ± 18.7723	596.2000 ± 23.2843

TAB. 5 – *Résultats de combinaisons de contraintes sur MNIST*

Cas	NMI	ACC	#changements	#changt. positifs
Sans contrainte	0.6320	0.5879	-	-
CS	0.5301	0.5425	8748	977
PW	0.6293 ± 0.0005	0.5905 ± 0.0003	2747.4000 ± 43.5137	977.4000 ± 13.8362
PW + CS	0.5368 ± 0.0006	0.5510 ± 0.0005	9600.8000 ± 38.2800	1580.2000 ± 28.0813

TAB. 6 – *Résultats de combinaisons de contraintes sur Fashion*

Remarquons une différence de comportement entre MNIST (Table 5) et Fashion (Table 6). Pour MNIST, ajouter des contraintes de taille améliore le résultat, ce qui n'est pas le cas avec Fashion. Il pourrait s'expliquer par un écart plus faible entre la taille maximale et la taille minimale pour Fashion que pour MNIST.

Pour MNIST, combiner les contraintes oblige la réaffectation de plus de points (environ 200) par rapport à des contraintes d'instances. En même temps, le nombre de points réaffectés correctement augmente aussi. Pour Fashion, entre PW et PW+CS, le nombre de changements ainsi que de changements positifs sont plus importants avec les contraintes CS. Cependant le ratio des changements positifs sur les changements totaux diminue (de 35,6% à 16,4%), ce qui mène à une diminution de la qualité du clustering.

Temps d'exécution. La table 7 présente le temps d'exécution en secondes des méthodes avec des contraintes d'instances. Pour avoir une comparaison équitable, pour COP-Kmeans et DCC nous prenons seulement le temps pour gérer les contraintes à partir du clustering initial donné par IDEC.

Données	#contraintes	Notre méthode	COP-Kmeans	DCC
MNIST	3600	1.0009 ± 0.0372	132.3833 ± 18.7203	1013.7586 ± 790.9106
	30000	3.8583 ± 0.3175	103.7654 ± 56.3909	1381.7345 ± 1067.0694
	60000	6.8137 ± 0.3919	70.9502 ± 38.4076	3277.3724 ± 1555.8282
Fashion	3600	0.9861 ± 0.0216	71.0007 ± 57.7661	5579.3029 ± 2761.3252
	30000	3.5111 ± 0.0155	103.7117 ± 35.2594	7359.3490 ± 3927.7909
	60000	6.5488 ± 0.4292	95.2828 ± 37.8964	3207.0003 ± 1057.7128

TAB. 7 – *Temps d'exécution avec des contraintes d'instances*

Le temps d'exécution dépend du nombre de contraintes. En moyenne, notre méthode est 10 fois plus rapide que COP-Kmeans et 500 fois plus rapide que DCC. En effet, COP-Kmeans doit calculer la matrice de distances après avoir mis à jour les centres des clusters, et DCC

applique la rétro-propagation pour mettre à jour les paramètres du modèle. De plus, le temps de calcul de notre méthode a une faible variance par rapport à COP-Kmeans et à DCC.

Notons que que ce soit pour les contraintes de taille des clusters ou les contraintes de couverture, notre méthode nécessite moins de 10 secondes pour ces données.

6 Conclusion

Des méthodes de clustering probabiliste intègrent des connaissances expertes sous forme de contraintes, mais elles sont en général limitées sur le type de contraintes et ne garantissent pas la satisfaction de toutes les contraintes même lorsque ces contraintes sont dures et devraient être satisfaites. Dans ce travail, nous proposons de forcer la satisfaction des contraintes a posteriori. Etant donnée une matrice d'affection probabiliste des points aux clusters, notre méthode retourne la meilleure affectation satisfaisant toutes les contraintes. Elle peut traiter de grands jeux de données, intégrer toutes les contraintes classiquement utilisées et elle peut être appliquée avec toute méthode de clustering probabiliste. Comme perspectives, nous envisageons de considérer d'autres types de connaissances, par exemples issues d'ontologies. Pour favoriser le passage à l'échelle, nous étudions des relaxations de la méthode comme par exemple, la Programmation Linéaire. Enfin nous souhaiterions intégrer des contraintes souples qui pourraient être utiles en cas de problèmes sur-contraints.

Références

Babaki, B., T. Guns, et S. Nijssen (2014). Constrained clustering using column generation. In *CPAIOR 2014*, pp. 438–454.

Bilenko, M., S. Basu, et R. J. Mooney (2004). Integrating constraints and metric learning in semi-supervised clustering. In *ICML 2004*, pp. 11–18.

Bradley, P., K. Bennett, et A. Demiriz (2000). Constrained k-means clustering. Technical Report MSR-TR-2000-65, Microsoft Research.

Dao, T.-B.-H., K.-C. Duong, et C. Vrain (2017). Constrained clustering by constraint programming. *Artificial Intelligence 244*, 70–94.

Dao, T.-B.-H., C. Vrain, K.-C. Duong, et I. Davidson (2016). A Framework for Actionable Clustering using Constraint Programming. In *ECAI 2016*, pp. 453–461.

Davidson, I. et S. S. Ravi (2005). Clustering with Constraints : Feasibility Issues and the k-Means Algorithm. In *SDM 2005*, pp. 138–149.

Davidson, I., S. S. Ravi, et L. Shamis (2010). A SAT-based Framework for Efficient Constrained Clustering. In *ICDM 2010*, pp. 94–105.

Ge, R., M. Ester, W. Jin, et I. Davidson (2007). Constraint-driven clustering. In *KDD 2007*, pp. 320–329.

Guo, X., L. Gao, X. Liu, et J. Yin (2017). Improved deep embedded clustering with local structure preservation. In *IJCAI 2017*, pp. 1753–1759.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly 2*(1-2), 83–97.

Lewis, D. D., Y. Yang, T. G. Rose, et F. Li (2004). Rcv1 : A new benchmark collection for text categorization research. *Journal of Machine Learning Research 5*(Apr), 361–397.

Liu, E. Y., Z. Zhang, et W. Wang (2011). Clustering with relative constraints. In *KDD 2011*, pp. 947–955.

Métivier, J.-P., P. Boizumault, B. Crémilleux, M. Khiari, et S. Loudni (2012). Constrained Clustering Using SAT. In *IDA 2012*, pp. 207–218.

Mueller, M. et S. Kramer (2010). Integer Linear Programming Models for Constrained Clustering. In *DS 2010*, pp. 159–173.

Ouali, A., S. Loudni, Y. Lebbah, P. Boizumault, A. Zimmermann, et L. Loukil (2016). Efficiently finding conceptual clustering models with integer linear programming. In *IJCAI 2016*, pp. 647–654.

Ren, Y., K. Hu, X. Dai, L. Pan, S. Hoi, et Z. Xu (2019). Semi-supervised deep embedded clustering. *Neurocomputing 325*, 121–130.

Song, C., F. Liu, Y. Huang, L. Wang, et T. Tan (2013). Auto-encoder based data clustering. In *CIARP 2013*, pp. 117–124.

Tang, W., Y. Yang, L. Zeng, et Y. Zhan (2019). Optimizing MSE for clustering with balanced size constraints. *Symmetry 11*(3).

Wagstaff, K. et C. Cardie (2000). Clustering with instance-level constraints. In *ICML 2000*, pp. 1103–1110.

Wagstaff, K., C. Cardie, S. Rogers, et S. Schrödl (2001). Constrained K-means Clustering with Background Knowledge. In *ICML 2001*, pp. 577–584.

Wang, X. et I. Davidson (2010). Flexible constrained spectral clustering. In *KDD 2010*, pp. 563–572.

Xie, J., R. Girshick, et A. Farhadi (2016). Unsupervised deep embedding for clustering analysis. In *ICML 2016*, pp. 478–487.

Zhang, H., S. Basu, et I. Davidson (2019). Deep constrained clustering - algorithms and advances. In *ECML 2019*.

Summary

Constrained clustering has received a lot of attention this last decade. It aims at integrating expert knowledge beyond the classic must-link and cannot-link constraints. Most probabilistic systems model this by integrating in the optimization criterion a second term penalizing the non satisfaction of constraints. They suffer from a lack of adaptability to various kinds of constraints and they do not guarantee the satisfaction of all the constraints, even if these constraints are hard and have to be satisfied. We propose a post-processing method that given a matrix assigning to each point their probability of belonging to a cluster, find the best assignment satisfying all the constraints. This method can be applied to any probabilistic algorithms, including deep clustering ones. Experiments show that when evaluated on a ground truth, our method is competitive in terms of clustering quality with the more recent approaches while being efficient.