Méthode ensemble de clustering profond

Séverine Affeldt*, Lazhar Labiod*, Mohamed Nadif*

*Université de Paris, CNRS, Centre Borelli, F-75006 Paris
<prénom.nom>@u-paris.fr

Résumé. Plusieurs stratégies de classification non supervisée combinant des algorithmes de *clustering* classiques et les auto-encodeurs ont été étudiées. Ces stratégies améliorent généralement les performances de clustering, mais dépendent des données et des hyperparamètres choisis qu'on ne peut fixer en amont pour un apprentissage non supervisé. Pour atténuer l'impact de la configuration des hyperparamètres sur la qualité du partitionnement, nous proposons une approche *ensemble*, ne nécessitant aucun pré-entraînement, qui exploite le clustering spectral et les forces de l'auto-encodeur. Notre approche génère plusieurs encodages profonds à partir des données d'origine, construit une matrice d'affinité *consensus* clairsemée et de faible dimension via la stratégie des *anchors* puis applique la classification spectrale pour obtenir l'espace commun partagé par les encodages. La stratégie des *anchors* assure une représentation efficace des encodages et leur fusion permet d'atténuer les problèmes d'hyperparamètres (Affeldt et al., 2020).

1 Introduction

Malgré leur succès, la plupart des méthodes de clustering sont mises à mal par les données générées avec les applications actuelles, qui sont généralement de grande dimension, hétérogènes et clairsemées. De nombreux chercheurs ont étudié de nouveaux modèles de clustering pour surmonter ces difficultés. Une catégorie prometteuse de ces modèles repose sur l'*embedding* des points (Allab et al., 2016, 2018; Labiod et Nadif, 2020). Dans l'apprentissage profond, les auto-encodeurs (DAE : Deep AutoEncoders) se sont avérés également utiles dans cette tâche (Hinton et Zemel, 1994) ; les auto-encodeurs peuvent transformer des données de manière non linéaire dans un espace latent. Un auto-encodeur se compose généralement d'un *encoder*, qui peut fournir un encodage des données d'origine en dimension inférieure, et d'un *decoder*, pour définir le coût de reconstruction des données. Dans le contexte du clustering, l'idée générale est d'intégrer les données dans un espace latent de faible dimension, puis d'effectuer le clustering dans ce nouvel espace.

Plusieurs travaux intéressants ont récemment combiné l'apprentissage profond de représentations et le clustering. Ces deux tâches peuvent être associées de manière séquentielle ou conjointe. Ainsi, Tian et al. (2014) utilisent un auto-encodeur *empilé* pour apprendre une représentation du graphe d'affinité, puis exécutent k-means sur les représentations apprises pour obtenir les clusters. Le processus d'embedding ne garantissant pas une représentation adaptée

à la tâche de clustering, plusieurs auteurs recommandent d'effectuer les deux tâches conjointement afin de laisser le clustering guider l'extraction des caractéristiques et *vice-versa*. Xie et al. (2016) proposent d'entraîner un réseau profond en minimisant de manière itérative une divergence de Kullback-Leibler (KL) entre une distribution de probabilité centroïde et une distribution cible auxiliaire. Plus récemment, Guo et al. (2017) ont intégré un auto-encodeur dans le cadre *Deep Embedded Clustering* (DEC) (Xie et al., 2016). L'approche effectue conjointement le clustering et l'apprentissage des représentations avec la préservation de la structure locale. Tian et al. (2017) proposent *DeepCluster*, un cadre général pour intégrer les méthodes de clustering traditionnelles dans des modèles d'apprentissage profond et adoptent l'algorithme des directions alternées [1] pour l'optimisation. Yang et al. (2017); Fard et al. (2020) proposent également une approche conjointe de réduction de dimension via un réseau profond et k-means.

Au-delà des approches conjointes ou séquentielles combinant clustering et représentations profondes, le lien entre l'auto-encodeur et l'apprentissage *ensemble* n'a pas encore été exploré. Nous avons pallié ce vide en proposant une approche robuste qui tire simultanément parti de plusieurs modèles profonds avec différents hyperparamètres discutées en détail dans (Affeldt et al., 2020).

2 Classification spectrale évolutive

Tout au long de l'article, les caractères gras majuscules désignent les matrices et gras minuscules désignent les vecteurs. Pour toute matrice $\mathbf{M}$, $\mathbf{m}_j$ désigne la j-ème colonne de $\mathbf{M}$, $\mathbf{y}_i$ désigne la i-ème ligne de $\mathbf{Y}$, m_{ij} la cellule $(i,j)-$ de $\mathbf{M}$ et $Tr[\mathbf{M}]$ est la trace de $\mathbf{M}$ si $\mathbf{M}$ est une matrice carrée ; $\mathbf{M}^\top$ désigne la matrice transposée de $\mathbf{M}$. On considère la norme de Frobenius d'une matrice $\mathbf{M} \in \mathbb{R}^{n\times d}$: $||\mathbf{M}||^2 = \sum_{i=1}^{n}\sum_{j=1}^{d} m_{ij}^2 = Tr[\mathbf{M}^\top\mathbf{M}]$. $\mathbf{I}$ la matrice d'identité de taille appropriée.

Plusieurs algorithmes de *spectral clustering* ont été proposés, chacun utilisant les vecteurs propres de manière légèrement différente (Shi et Malik, 2000; Meila et Shi, 2001). La partition des n points de $\mathbf{X} \in \mathbb{R}^{n\times d}$ en k clusters disjoints est basée sur une fonction objective qui favorise une faible similarité entre les classes et une forte similarité au sein des classes. Dans sa version normalisée, l'algorithme de *spectral clustering* exploite les k premiers vecteurs propres du graphe normalisé Laplacian qui sont les relaxations des vecteurs indicateurs fournissant des affectations de chaque point à une classe. Cela revient donc à maximiser,

$$\max_{\mathbf{B}\in\mathbb{R}^{n\times k}} Tr(\mathbf{B}^\top\mathbf{S}\mathbf{B}) \quad s.t. \quad \mathbf{B}^\top\mathbf{B} = \mathbf{I} \tag{1}$$

avec $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2} \in \mathbb{R}^{n\times n}$ la matrice de similarité normalisée, $\mathbf{K} \in \mathbb{R}^{n\times n}$ la matrice de similarité et $\mathbf{D} \in \mathbb{R}^{n\times n}$ la matrice diagonale dont l'élément (i,i)- de $\mathbf{X}$ est la somme des i-ème ligne de $\mathbf{X}$. La solution de (1) est de choisir $\mathbf{B} \in \mathbb{R}^{n\times k}$ égale aux k vecteurs propres correspondant aux k plus grandes valeurs propres de $\mathbf{S}$. Après renormalisation de chaque ligne de $\mathbf{B}$, k-means assigne chaque $\mathbf{x}_i$ de $\mathbf{X}$ à la classe d'affectation de la ligne $\mathbf{b}_i$.

Récemment, une approche de classification spectrale évolutive, *Landmark-based Spectral Clustering* (*LSC*) (Chen et Cai, 2011) ou *Anchor-Graph* (Liu et al., 2010), a été proposée. Elle permet de construire efficacement le graphe laplacien et de calculer la *décomposition propre*. Plus précisément, chaque point est représenté par une combinaison linéaire de p points

1. ADA ; en anglais ADMM pour Alternating Direction Method of Multipliers

de données représentatifs (ou *points de repère*), avec $p \ll n$. La matrice de représentation obtenue $\hat{\mathbf{Z}} \in \mathbb{R}^{p \times n}$, pour laquelle l'affinité/similarité est calculée entre n points de données et les p points de repère, est clairsemée assurant une décomposition propre plus efficace que la décomposition mentionnée ci-dessus de $\mathbf{S}$ (Eq. 1).

3 Auto-Encodeur et classification via *ensemble* DAE

Un auto-encodeur est un réseau neuronal qui implémente un algorithme d'apprentissage non supervisé dans lequel les paramètres sont appris de telle sorte que les valeurs de sortie ont tendance à copier l'échantillon d'apprentissage en entrée. Il se compose de deux parties, à savoir un *encoder*, f_θ, suivi d'un *decoder*, g_ψ. La première partie calcule un vecteur de caractéristiques, $\mathbf{y}_i = f_\theta(\mathbf{x}_i)$, pour chaque échantillon d'apprentissage d'entrée, fournissant ainsi l'encodage $\mathbf{Y}$ de l'ensemble de données d'entrée. La partie *decoder* reconvertit l'encodage dans sa représentation originale, $\hat{\mathbf{x}}_i = g_\psi(\mathbf{y}_i)$. Les ensembles de paramètres pour f_θ et g_ψ sont appris simultanément pendant la tâche de reconstruction tout en minimisant la perte, $\mathcal{J}_{AE}(\theta, \psi) = \sum_{i=1}^{n} \mathcal{L}(\mathbf{x}_i, g_\psi(f_\theta(\mathbf{x}_i)))$, où $\mathcal{L}$ est une fonction de coût qui mesure la divergence entre l'échantillon d'apprentissage d'entrée et les données reconstruites.

A partir de $\mathbf{X}$ de taille $n \times d$, le but est d'obtenir d'abord un ensemble de m encodages $\{\mathbf{Y}_\ell\}_{\ell \in [1,m]}$ en utilisant m DAEs ayant différents paramétrages. Dans un second temps, on construit une matrice de graphes $\mathbf{S}_\ell$ associée à chaque embedding $\mathbf{Y}_\ell$, puis on fusionne les m matrices de graphes dans une matrice de graphes consensuelle $\mathbf{S}$ (nommée de manière équivalente matrice d'*affinité* ou de *similarité*) qui contient des informations fournies par les m embeddings. Enfin, pour exploiter le sous-espace commun partagé par les m embeddings profonds, le clustering spectral est appliqué à $\mathbf{S}$. Les défis du problème sont triples, 1) générer m embeddings profonds, 2) placer le clustering dans un cadre d'apprentissage *ensemble*, 3) résoudre la tâche de clustering de manière très efficace.

La fonction de coût d'un auto-encodeur, avec un encodeur f_θ et un décodeur g_ψ, mesure l'erreur entre l'entrée $\mathbf{x} \in \mathbb{R}^{d \times 1}$ et sa reconstruction à la sortie $\hat{\mathbf{x}} \in \mathbb{R}^{d \times 1}$. L'encodeur f_θ et le décodeur g_ψ peuvent avoir plusieurs couches de largeurs différentes. Pour générer m représentations ou encodages profonds $\{\mathbf{Y}_\ell\}_{\ell \in [1,m]}$, le DAE est formé avec différents hyperparamètres (eg., initialisation, largeurs de couche) en optimisant la fonction de coût suivante.

$$||\mathbf{X} - g_{\psi_\ell}(f_{\theta_\ell}(\mathbf{X}))||^2 \tag{2}$$

où g_{ψ_ℓ} et f_{θ_ℓ} sont appris avec les hyperparamètres ℓ, et $\mathbf{Y}_\ell = f_{\theta_\ell}(\mathbf{X})$ (Fig. 1, (a)). Pour construire $\mathbf{S}_\ell$, nous utilisons une idée similaire à celle de *Landmark Spectral Clustering* (Chen et Cai, 2011) et des *Anchor-Graphs* (Liu et al., 2010) où une matrice de représentation $\mathbf{Z}_\ell \in \mathbb{R}^{n \times p}$, de taille réduite et clairsemée, est construite entre les points de repère $\{\mathbf{u}_j^\ell\}_{j \in [1,p]}$ et les points encodés $\{\mathbf{y}_i^\ell\}_{i \in [1,n]}$ (Fig. 1, (a)). Plus précisément, un ensemble de p points ($p \ll n$) est obtenu par k-means appliqué sur la matrice de embedding $\mathbf{Y}_\ell$. Puis, un mappage non linéaire des données au repère est calculé comme suit,

$$z_{ij}^\ell = \Phi(\mathbf{y}_i^\ell) = \frac{\mathcal{K}(\mathbf{y}_i^\ell, \mathbf{u}_j^\ell)}{\sum_{j' \in N_{(i)}} \mathcal{K}(\mathbf{y}_i^\ell, \mathbf{u}_{j'}^\ell)};\ j' \in N_{(i)} \tag{3}$$

où $N_{(i)}$ indique les points de repère les plus proches r ($r < p$) autour de $\mathbf{y}_i^\ell$. Comme proposé par Chen et Cai (2011), nous mettons z_{ij}^ℓ à zéro lorsque le repère $\mathbf{u}_j^\ell$ n'est pas parmi le plus

proche voisin de $\mathbf{y}_i^\ell$, ce qui conduit à une matrice d'affinité clairsemée $\mathbf{Z}_\ell$. La fonction $\mathcal{K}(.)$ est utilisée pour mesurer la similitude entre les données $\mathbf{y}_i^\ell$ et l'ancre $\mathbf{u}_j^\ell$ avec L_2 distance dans l'espace noyau gaussien $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2)$, et σ est le paramètre de bande passante. La matrice normalisée $\hat{\mathbf{Z}}_\ell \in \mathbb{R}^{n\times p}$ est ensuite utilisée pour obtenir un graphe de rang inférieur, $\mathbf{S}_\ell \in \mathbb{R}^{n\times n}, \mathbf{S}_\ell = \mathbf{Z}_\ell\Sigma^{-1}\mathbf{Z}_\ell^\top$ où $\Sigma = diag(\mathbf{Z}_\ell^\top \mathbb{1})$. Comme Σ^{-1} normalise la matrice construite, $\mathbf{S}_\ell$ est bi-stochastique, *ie* la somme de chaque colonne et ligne est égale à un, et le graph laplacien devient, $\mathbf{S}_\ell = \hat{\mathbf{Z}}_\ell\hat{\mathbf{Z}}_\ell^\top$, où $\hat{\mathbf{Z}}_\ell = \mathbf{Z}_\ell\Sigma^{-1/2}$.

3.1 Ensemble de matrices d'affinité et algorithme

Étant donné un ensemble de m encodages $\{\mathbf{Y}_\ell\}_{\ell\in[1,m]}$ obtenu en utilisant m DAEs formés avec ℓ différents hyperparamètres, l'objectif est de fusionner les m graphes des matrices de similarité $\mathbf{S}_\ell$ dans une matrice de similarité consensus qui contient des informations fournies par les m embeddings. Pour agréger les différentes matrices de similarité, nous utilisons une idée de clustering *ensemble* analogue à celle proposée dans (Strehl et Ghosh, 2003; Vega-Pons et Ruiz-Shulcloper, 2011) où une matrice de *co-association* est d'abord construite comme la somme de toutes les matrices de similarité de base, et où chaque matrice de partition de base peut être représentée comme une matrice diagonale en blocs. Ainsi, la matrice d'affinité *consensus* est construite comme la somme des m matrices de similarité de base en utilisant la formule suivante, $\bar{\mathbf{S}} = \frac{1}{m}\sum_{\ell=1}^m \mathbf{S}_\ell$. Notons que la matrice obtenue, $\bar{\mathbf{S}}$ est toujours bi-stochastique. Pour de nombreux problèmes, $\bar{\mathbf{S}}$ est approximativement une matrice stochastique en blocs, et donc les premiers k vecteurs propres de $\bar{\mathbf{S}}$ sont approximativement constants par morceaux sur les k sous-ensembles de lignes presque invariants. Dans la suite, nous cherchons à calculer, à moindre coût, $\mathbf{B}$ qui est partagée par les m matrices de graphes $\mathbf{S}_\ell$, et obtenue en optimisant le problème de maximisation de trace suivant $\max_{\mathbf{B}} Tr(\mathbf{B}^\top\bar{\mathbf{S}}\mathbf{B}) \quad s.t. \quad \mathbf{B}^\top\mathbf{B} = \mathbf{I}$. La solution de ce problème est de fixer la matrice $\mathbf{B}$ égale aux k vecteurs propres correspondant aux k plus grandes valeurs propres de $\bar{\mathbf{S}}$. Cependant, comme le calcul de la décomposition propre de $\bar{\mathbf{S}}$ de taille $(n \times n)$ est $O(n^3)$, en nous appuyant sur la proposition 3.1 de Affeldt et al. (2020), nous proposons de calculer les k vecteurs singuliers gauches de la matrice concaténée,

$$\bar{\mathbf{Z}} = \frac{1}{\sqrt{m}}[\hat{\mathbf{Z}}_1|\dots|\hat{\mathbf{Z}}_j|\dots|\hat{\mathbf{Z}}_m]. \tag{4}$$

Utiliser la matrice clairsemée $\bar{\mathbf{Z}} \in \mathbb{R}^{n\times\sum_{j=1}^m \ell_j}$ avec $\sum_{j=1}^m \ell_j \ll n$, à la place de $\bar{\mathbf{S}}$ dont la dimension est plus grande, induit naturellement une amélioration du coût de calcul de $\mathbf{B}$ (Fig. 1, (b)). Les étapes de notre algorithme se basent sur la proposition suivante,

Proposition 3.1. *Etant donné m matrices de similarités $\mathbf{S}_\ell$, tel que chaque matrice $\mathbf{S}_\ell$ peut être exprimé comme $\mathbf{Z}_\ell\mathbf{Z}_\ell^\top$. Soit $\bar{\mathbf{Z}} \in \mathbb{R}^{n\times\sum_{j=1}^m \ell_j}$, où $\sum_{j=1}^m \ell_j \ll n$, noté $\frac{1}{\sqrt{m}}[\mathbf{Z}_1|\dots|\mathbf{Z}_j|\dots|\mathbf{Z}_m]$, soit la concaténation des $\mathbf{Z}_\ell$'s, $\ell = 1,\dots,m$. Nous avons d'abord,*

$$\max_{\mathbf{B}^\top\mathbf{B}=\mathbf{I}} Tr(\mathbf{B}^\top\bar{\mathbf{S}}\mathbf{B}) \Leftrightarrow \min_{\mathbf{B}^\top\mathbf{B}=\mathbf{I},\mathbf{M}} ||\bar{\mathbf{Z}} - \mathbf{B}\mathbf{M}^\top||_F^2. \tag{5}$$

Etant donné $\texttt{SVD}(\bar{\mathbf{Z}})$, $\bar{\mathbf{Z}} = \mathbf{U}\Sigma\mathbf{V}^\top$ *et la solution optimale* $\mathbf{B}^*$ *est égale à* $\mathbf{U}$.

Les étapes de SC−EDAE sont résumées dans l'Algorithme 1 et illustrées par la Figure 1. Notre approche propose une manière unique de combiner les encodages DAE avec le *Spectral*

Algorithm 1 : SC−EDAE algorithm

Input : data matrix $\mathbf{X}$;
Initialize : m DAE with different hyperparameters setting ;
Do :
(a) Generate m deep embedding $\{\mathbf{Y}_\ell\}_{l\in[1,m]}$ *(Eq. 2)*
(b) Construct the ensemble sparse affinity matrix $\bar{\mathbf{Z}} \in \mathbb{R}^{n\times\sum_{j=1}^{m}\ell_j}$ *(Eq. 3, 4)*
(c) Compute $\mathbf{B}^* \in \mathbb{R}^{\mathbf{n}\times\mathbf{k}}$ by performing *sparse* SVD on $\bar{\mathbf{Z}}$ *(Eq. 5)*
Output : Run k-means on $\mathbf{B}^*$ to get the final clustering

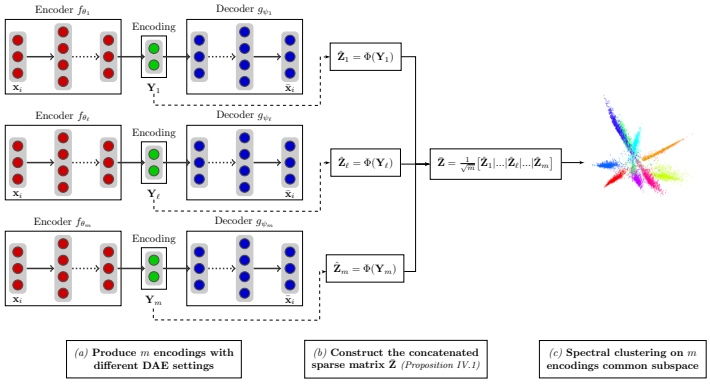

FIG. 1 – **Description de SC-EDAE.** *L'algorithme* SC−EDAE *calcule d'abord* m *encodages à partir des DAEs avec différents paramétrages* (a)*, puis génère* m *matrices d'affinité clairsemées,* $\{\hat{\mathbf{Z}}_\ell\}_{\ell\in[1,m]}$*, qui sont concaténées dans* $\bar{\mathbf{Z}}$ (b)*, et exécute finalement une* SVD *sur* $\bar{\mathbf{Z}}$ (c).

Clustering via une approche *ensemble*. SC−EDAE bénéficie de la faible complexité de la stratégie des *anchors* pour la construction de la matrice d'affinité du graphe et la décomposition.

L'originalité et l'efficacité de notre méthode *ensemble* reposent sur le remplacement d'une décomposition coûteuse sur $\bar{\mathbf{S}} \in \mathbb{R}^{n\times n}$ par une décomposition sur un matrice de dimension réduite et clairsemée $\bar{\mathbf{Z}} \in \mathbb{R}^{n\times\sum_{j=1}^{m}\ell_j}$, avec $\sum_{j=1}^{m}\ell_j \ll n$ (Alg. 1, étape (c)). En particulier, la *sparsité* de $\bar{\mathbf{Z}}$ permet d'utiliser une décomposition itérative rapide moins coûteuse.

4 Expériences

Les auto-encodeurs sont entièrement connectés avec un encodeur f_θ de trois couches cachées de taille 500, 750 ou 1000 comme suggéré par Bengio et al. (2007), dans tous les ordres possibles. La partie décodeur g_ψ reflète l'encodeur f_θ. Pour chaque architecture DAE, 5 encodages sont générés avec 50, 100, 150, 200 et 250 époques. L'initialisation des poids suit l'approche de Glorot et tous les encodeur/décodeur utilisent des unités linéaires rectifiées (ReLU), à l'exception de la couche de sortie qui nécessite une fonction sigmoïde. Les données de l'auto-encodeur sont normalisées L_2. SC−EDAE exploite des encodages $\{\mathbf{Y}_\ell\}_{\ell\in[1,m]}$ qui sont générés avec soit *(i)* m initialisations différentes ou m nombres d'époques différents pour une même structure de DAE, ou *(ii)* m DAE avec des structures différentes pour le même nombre de points de repère et d'époques. Dans les deux cas, SC−EDAE permet de calculer les différentes matrices d'affinités creuses m $\{\hat{\mathbf{Z}}_\ell\}_{\ell\in[1,m]}$ (Eq. 3) pour générer la matrice d'affinité $\bar{\mathbf{Z}}$ (Eq. 4).

Notre algorithme SC−EDAE (Alg. 1) est évalué sur **MNIST** et **USPS** [2] et leurs encodages DAE. La base de données **MNIST** est chargée à partir du package Python Keras. Les ensembles d'entraînement et de test contiennent respectivement 60000 et 10000 images de taille 28×28 d'entiers compris entre 0 et 9. Les images sont en niveaux de gris redimensionnés entre $[0, 1]$. La base de données **USPS** est préparée comme proposé dans Guo et al. (2017) et contient 9298 images de taille 16×16 pixels d'entiers compris entre 0 et 9.

Nous évaluons d'abord k-means et LSC (Chen et Cai, 2011) sur les deux ensembles de données réels. L'approche $\texttt{kmeans}_{++}$ correspond à l'implémentation de k-means du package Python scikit−learn avec les paramètres par défaut et $\texttt{kmeans}_{++}$ pour l'initialisation (Arthur et Vassilvitskii, 2007). Nous avons implémenté la méthode LSC en Python, en suivant l'implémentation Matlab proposée par Chen et Cai (2011), et avons conservé les mêmes paramètres par défaut. Suivant les résultats de Banijamali et Ghodsi (2017), nous avons initialisé les repères de LSC avec k-means, et non aléatoirement. Le nombre de points de repère est pris entre 100 et 1000, par pas de 100. La précision rapportée pour LSC et $\texttt{k−means}_{++}$ correspond à la moyenne pour 10 répliques de clustering sur les ensembles de données d'origine, sur tous les nombres d'époque et de repères. La précision rapportée pour DAE−LSC et $\texttt{DAE−kmeans}_{++}$ correspond à une moyenne sur 50 répliques (10 répliques sur chacun des 5 encodages par structure DAE), sur tous les nombres d'époque et de repères.

Plusieurs stratégies utilisant un algorithme d'apprentissage profond et des approches de type k-means, séquentiellement ou conjointement, ont démontré une amélioration de la précision de la tâche de clustering. Parmi ces méthodes, deux approches peuvent désormais être considérées comme des méthodes populaires, à savoir IDEC (*Improved Deep Embedded Clustering*) (Guo et al., 2017) et DCN (*Deep Clustering Network*) (Yang et al., 2017). Très récemment, l'algorithme DKM (*Deep k-means*), qui applique k-means dans un espace d'embedding via AE, a surpassé ces approches (Fard et al., 2020). Nous comparons SC−EDAE à ces trois

Model	MNIST		USPS	
	ACC	NMI	ACC	NMI
baselines				
$\texttt{kmeans}_{++}$	55.13 ±0.05	52.89 ±0.02	68.36 ±0.08	65.67 ±0.10
LSC	68.55 ±2.25	70.54 ±0.83	77.20 ±1.49	79.48 ±0.90
$\texttt{DAE+kmeans}_{++}$	78.40 ±6.09	71.97 ±4.13	73.17 ±3.27	70.48 ±1.84
DAE+LSC	89.78 ±5.14	83.06 ±4.38	81.62 ±6.25	80.44 ±3.39
no pretraining required				
SC−EDAE Ens.Init.	92.91 ±0.24	87.65 ±0.18	81.46 ±1.48	82.88 ±0.59
SC−EDAE Ens.Ep.	92.33 ±2.77	87.72 ±2.42	**81.88** ±3.62	83.03 ±1.88
SC−EDAE Ens.Struct.	**93.23** ±2.84	**87.93** ±2.27	81.78 ±3.61	**83.17** ±1.96
Deep clustering approaches without pretraining *(Fard et al. 2020) Fard et al. (2020)*				
$\texttt{DCN}^{np}$	34.8 ±3.0	18.1 ±1.0	36.4 ±3.5	16.9 ±1.3
$\texttt{IDEC}^{np}$	61.8 ±3.0	62.2 ±1.6	53.9 ±5.1	50.0 ±3.8
$\texttt{DKM}^{a}$	82.3 ±3.2	78.0 ±1.9	75.5 ±6.8	73.0 ±2.3
Deep clustering approaches with pretraining *(Fard et al. 2020) Fard et al. (2020)*				
$\texttt{DCN}^{p}$	81.1 ±1.9	75.7 ±1.1	73.0 ±0.8	71.9 ±1.2
$\texttt{IDEC}^{p}$	85.7 ±2.4	86.4 ±1.0	75.2 ±0.5	74.9 ±0.6
$\texttt{DKM}^{p}$	84.0 ±2.2	79.6 ±0.9	75.7 ±1.3	77.6 ±1.1

TAB. 1 – *Comparaisons de différentes méthodes de classification non supervisée.*

méthodes et résumons ces évaluations dans la Table 1. Les six dernières lignes de cette table sont directement extraites de l'étude de Fard et al. (2020). La précision et les valeurs NMI de ces six lignes sont une moyenne sur 10 essais. Les autres valeurs correspondent à nos évaluations. Les résultats de base sont donnés dans les quatre premières lignes et correspondent

2. **MNIST** : Modified National Institute of Standards and Technology ; **USPS** : US Postal Service

à la tâche de clustering via k−means$_{++}$ ou LSC (résultats moyens sur 10 exécutions), et via une combinaison de DAE et k−means ou LSC (résultats moyens sur 10 exécutions pour chacun des 5 encodages différents). Les lignes SC−EDAE donnent la précision et les résultats NMI pour notre méthode d'ensemble, avec un ensemble sur plusieurs initialisations (SC−EDAE Ens.Init.), nombres d'époque (SC−EDAE Ens .Ep.) et architectures DAE (SC−EDAE Ens.Struct.). Comme le montre Table 1, alors que notre approche SC−EDAE ne nécessite aucun pré-entraînement, elle surpasse les méthodes DCN et IDEC dans leur version pré-entrainée (Table 1, DCN^p et IDEC^p résultats). La méthode DKM fonctionne bien avec et sans pré-entrainement. Pourtant, notre approche SC−EDAE atteint une précision et des résultats NMI plus élevés que l'approche DKM avec et sans pré-entrainement.

5 Conclusion

Pour améliorer le partitionnement de grands ensembles de données, plusieurs auteurs ont proposé d'associer, séquentiellement ou conjointement, une architecture profonde et des méthodes de clustering classiques. Cependant, ces méthodes sont généralement confrontées à des problèmes importants liés à des défis bien connus de l'apprentissage profond, tels que l'initialisation des poids ou les paramètres de l'architecture. SC−EDAE atténue ces problèmes en exploitant une approche *ensemble* combinant plusieurs modèles profonds avant d'appliquer un *spectral clustering* ; elle est assez simple et peut être résumé en trois étapes : générer m embeddings profonds à partir des données d'origine, construire une matrice d'affinité consensus clairsemée et de faible dimension basée sur la stratégie des *anchors*, et appliquer une classification spectrale pour obtenir l'espace commun partagé par les m les encodages. Pour plus de détails concernant les développements des étapes d'optimisation et les performances de cette méthode, le lecteur peut consulter (Affeldt et al., 2020).

Références

Affeldt, S., L. Labiod, et M. Nadif (2020). Spectral clustering via ensemble deep autoencoder learning (sc-edae). *Pattern Recognition 108*, 107522.

Allab, K., L. Labiod, et M. Nadif (2016). A semi-NMF-PCA unified framework for data clustering. *IEEE Transactions on Knowledge and Data Engineering 29*(1), 2–16.

Allab, K., L. Labiod, et M. Nadif (2018). Simultaneous spectral data embedding and clustering. *IEEE Transactions on Neural Networks and Learning Systems 29*(12), 6396–6401.

Arthur, D. et S. Vassilvitskii (2007). k-means++ : The advantages of careful seeding. In *Proceedings of the* 18^{th} *annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035.

Banijamali, E. et A. Ghodsi (2017). Fast spectral clustering using autoencoders and landmarks. In *International Conference Image Analysis and Recognition*, pp. 380–388. Springer.

Bengio, Y., P. Lamblin, D. Popovici, et H. Larochelle (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pp. 153–160.

Chen, X. et D. Cai (2011). Large scale spectral clustering with landmark-based representation. In 25^{th} *Conference on Artificial Intelligence (AAAI)*.

Fard, M. M., T. Thonet, et E. Gaussier (2020). Deep k-means : Jointly clustering with k-means and learning representations. *Pattern Recognition Letters 138*, 185–192.

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

Hinton, G. E. et R. S. Zemel (1994). Autoencoders, minimum description length and Helmholtz free energy. In *Advances in neural information processing systems*, pp. 3–10.

Labiod, L. et M. Nadif (2020). Efficient regularized spectral data embedding. *Advances in Data Analysis and Classification*, 1–21.

Liu, W., J. He, et S.-F. Chang (2010). Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th ICML*.

Meila, M. et J. Shi (2001). Learning segmentation by random walks. In *Advances in neural information processing systems*, pp. 873–879.

Shi, J. et J. Malik (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence 22*(8), 888–905.

Strehl, A. et J. Ghosh (2003). Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res. 3*, 583–617.

Tian, F., B. Gao, Q. Cui, E. Chen, et T. Liu (2014). Learning deep representations for graph clustering. In *AAAI 2014*, pp. 1293–1299.

Tian, K., S. Zhou, et J. Guan (2017). Deepcluster : A general clustering framework based on deep learning. In *Machine Learning and Knowledge Discovery in Databases*.

Vega-Pons, S. et J. Ruiz-Shulcloper (2011). A survey of clustering ensemble algorithms. *International Journal of Pattern Recognition and Artificial Intelligence 25*(03), 337–372.

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

Yang, B., X. Fu, N. D. Sidiropoulos, et M. Hong (2017). Towards k-means-friendly spaces : Simultaneous deep learning and clustering. In *Proceedings of the 34th ICML*, pp. 3861–3870.

Summary

Several works have studied clustering strategies that combine classical clustering algorithms and deep learning methods. These strategies generally improve clustering performance, however deep autoencoder setting issues impede the robustness of these approaches. To alleviate the impact of hyperparameters setting, we propose a model which combines spectral clustering and deep autoencoder strengths in an ensemble framework. Our proposal does not require any pretraining and includes the three following steps: generating various deep embeddings from the original data, constructing a sparse and low-dimensional ensemble affinity matrix based on anchors strategy and applying spectral clustering to obtain the common space shared by multiple deep representations. While the anchors strategy ensures an efficient merging of the encodings, the fusion of various deep representations enables to mitigate the deep networks setting issues (Affeldt et al., 2020).