UNIVERSITÉ DU QUÉBEC À MONTRÉAL

UNE APPROCHE SCALABLE DU CLUSTERING DES SOUS-ESPACES DANS UN CONTEXTE D'APPRENTISSAGE PROFOND

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

SIHEM SAMI

AOÛT 2022

REMERCIEMENT

Tout d'abord, je tiens à remercier mon directeur de recherche, Mohamed Bouguessa, pour son encadrement, son assistance et sa rigueur, qui m'ont beaucoup aidé dans la réalisation de ce travail.

Je tiens également à remercier Nairouz Mrabah pour son accompagnement et ses conseils, qui m'ont beaucoup aidé tout au long de cette maîtrise.

Je remercie également la faculté des sciences de l'UQAM pour les bourses d'exemption des frais de scolarité majorés que j'ai reçues durant mes études de maîtrise.

J'adresse enfin une reconnaissance particulière à ma famille et mes amis qui m'ont apporté leur support moral tout au long de mes études et à tous ceux qui ont contribué de près ou de loin à l'accomplissement de ce travail.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

RÉSUMÉ

Les méthodes de clustering des sous-espaces vectoriels (*Subspace Clustering*) basées sur la propriété d'autoreprésentation sont efficaces et ont connu un grand succès. Cependant, les deux étapes du subspace clustering, à savoir, la construction de la matrice d'affinité et le clustering spectral, souffrent d'une complexité temporelle et spatiale élevée, ce qui rend difficile le regroupement de grands ensembles de données. De ce fait, des chercheurs se sont intéressés à la scalabilité du subspace clustering en essayant de réduire cette complexité. Bien que de nouvelles techniques ont été proposées, ces techniques souffrent des problèmes suivants : (1) le coût de calcul reste élevé, car elles sont résolues avec une manière d'optimisation itérative, (2) certaines se basent sur des hypothèses trop restrictives, (3) la complexité peut être réduite au détriment de la précision du clustering, (4) ces techniques se concentrent que sur le clustering des sous-espaces linéaires (les données ne sont pas forcément représentées à partir de sous-espaces linéaires). Récemment, d'autres recherches se sont intéressées aux réseaux de neurones profonds pour le subspace clustering. Les recherches les plus récentes utilisent les auto-encodeurs (AEs) en introduisant une nouvelle couche d'autoreprésentation (self-expression) entre l'encodeur et le décodeur afin d'imiter la propriété d'autoreprésentation. Toutefois, nous avons remarqué qu'il n'y avait pas de travaux traitant la scalabilité du subspace clustering dans le contexte de l'apprentissage profond.

Pour palier aux limites des approches existantes, nous présentons dans le cadre de ce mémoire une nouvelle approche scalable du subspace clustering profond. L'approche proposée utilise une couche d'autoreprésentation entre l'encodeur et le décodeur pour apprendre une matrice plus petite, en se basant sur des points de repère, pour construire la matrice d'affinité. Notre solution permet de réduire la complexité relative à la construction de la matrice d'affinité et à celle du clustering spectral de $O(n^3)$ vers une complexité linéaire $O(n)$. L'efficacité de l'approche proposée est comparée à d'autres méthodes récentes sur différents ensembles de données. Les résultats des expérimentations montrent une amélioration significative de la précision du clustering par rapport aux solutions scalables existantes, et confirment la capacité de notre approche à identifier effectivement les sous-espaces.

Mots clés : Subspace clustering profond, Couche d'autoreprésentation (*self-expression*), Points de repère.

CHAPITRE I

INTRODUCTION

1.1 Mise en contexte :

De nos jours, la quantité de données qui ne cessent d'augmenter a fait naître un besoin essentiel et nécessaire de les exploiter et les analyser. Par conséquent, plusieurs et différentes techniques ont vue le jour, comme le partitionnement des données, appelé aussi clustering en anglais. Cette approche sert à grouper les données en *clusters* de sorte que les données d'un même cluster répondent aux mêmes critères et sont plus similaires les unes aux autres que les données des autres clusters. Par exemple, partitionner une base de données d'image selon une distribution de pixel similaire ou un arrière-plan similaire.

Les algorithmes de clustering sont importants dans la science de données. Ils sont largement utilisés dans de nombreux domaines, comme les statistiques, la biologie et diverses applications d'analyses de données. Par exemple, pour analyser des données biologiques complexes (comme les données génomiques), les chercheurs visent généralement à identifier certains motifs qui coexistent sous forme de clusters. Par ce fait, le clustering devient un outil indispensable pour diverses tâches de découverte de connaissances dans le domaine de la biologie. Les algorithmes de clustering peuvent être aussi utilisés dans d'autres applications. Par exemple, pour grouper des maladies en science médicale, grouper des clients dans le do-

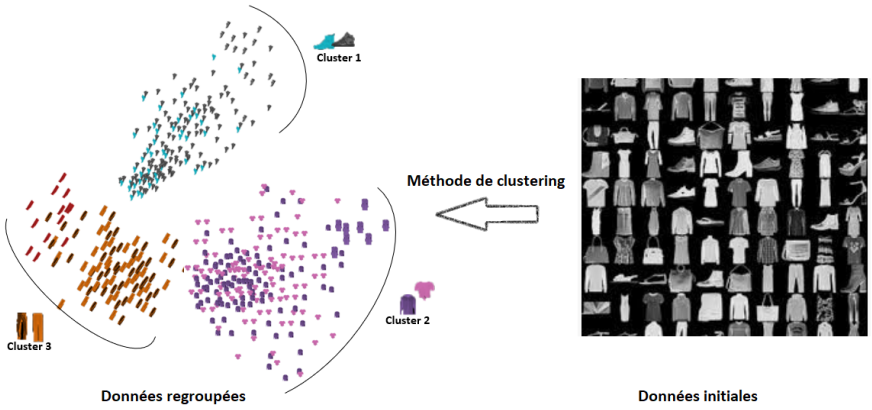


FIGURE 1.1 – Exemple de clustering.

maine des études de marché ou encore la détection des objets pour améliorer la conduite autonome des véhicules en utilisant le clustering des images.

Les méthodes de clustering standards (Rokach et Maimon, 2005) donnent des résultats satisfaisants sur certains types de données. Ces méthodes essayent de trouver les caractéristiques ou attributs, appelés aussi dimensions, les plus pertinents pour chaque cluster. Ainsi, tous les attributs sont considérés avec le même degré d'importance lors du calcul des similarités entre deux points de données. Cependant, les données du monde réel sont multiples et souvent représentées avec des quantités importantes d'attributs qui peuvent être redondants ou corrélés. De plus, en présence de données bruitées, d'autres attributs peuvent être inutiles et n'apportent aucune information pertinente. Par exemple, les données textuelles explorées à partir d'internet sont bruyantes. Ces textes sont généralement écrits de manière informelle et souffrent d'erreurs grammaticales, de fautes d'orthographe et de ponctuation incorrecte. Par conséquent, la présence d'attributs non pertinents réduit la qualité du clustering, occupe plus de mémoire et nécessite davantage de temps. Bien que, un ensemble de données est de grandes dimensions, les dimensions intrinsèques (réels) sont souvent beaucoup plus petites que les dimensions

de l'espace ambiant. Par exemple, le nombre de pixels dans une image peut être assez grand, mais la plupart des modèles de vision par ordinateur n'utilisent que quelques paramètres pour décrire l'apparence, la géométrie et la dynamique d'une scène.

Afin de répondre aux problématiques de la haute dimensionnalité, une solution serait de réduire la dimensionnalité des données à grande dimension. Cela a motivé le développement d'un certain nombre de techniques de réduction de dimensions pour trouver une représentation à faible dimension (Ye *et al.*, 2007), (De la Torre et Kanade, 2006), (Van Der Maaten *et al.*, 2009). Néanmoins, avec la croissance de la taille des données, ces méthodes deviennent difficiles à appliquer et la mise à l'échelle des algorithmes utilisés est compromise. De plus, ces techniques supposent que les données sont représentées à partir d'un seul sous-espace de faibles dimensions. Cependant, en pratique, les données sont de haute dimensionnalité et peuvent être représentées par une union de sous-espaces de faibles dimensions. Par exemple, en bio-informatique avec les données de puces à ADN, des cellules d'une population peuvent être similaires à d'autres parce qu'elles produisent de la chlorophylle. Donc, elles peuvent être regroupées en fonction des niveaux d'expression d'un certain ensemble de gènes liés à la chlorophylle. Cependant, une autre population pourrait être similaire parce que les cellules sont régulées par le mécanisme de l'horloge circadienne de l'organisme. Dans ce cas, elles seraient regroupées sur un ensemble différent de gènes. Ces deux relations représentent des clusters dans deux sous-ensembles distincts de gènes (Parsons *et al.*, 2004). De plus, peu d'informations sont disponibles sur les clusters recherchés, ce qui nécessite des algorithmes utilisant un apprentissage non supervisé. Ce dernier devient une tâche plus difficile avec ce type de données. En effet, les algorithmes utilisés doivent simultanément regrouper les données dans plusieurs sous-espaces et trouver un sous-espace de faible dimension pour chaque cluster. Dans ce cas,

le problème du clustering est réduit au problème d'assigner chaque donnée à son propre sous-espace vectoriel.

Pour répondre à ces problématiques, les méthodologies de clustering des sous-espaces vectoriels (*Subspace Clustering*) ont été proposées. Le subspace clustering est largement étudié dans la littérature. Il est basé sur l'hypothèse que les points de données de haute dimensionnalité sont répartis autour de plusieurs sous-espaces linéaires de faible dimension. Par exemple, sous une réflectance lambertienne, les images de visage d'un sujet obtenues avec une pose fixe et des conditions d'éclairage variables se situent dans un sous-espace de faible dimension avec le nombre de dimensions proches de neuf (Basri et Jacobs, 2003). Par conséquent, le subspace clustering peut être utilisé pour regrouper des images selon leurs sujets respectifs (Ji *et al.*, 2017). Le but de ces méthodologies est de trouver les sous-espaces vectoriels, leurs dimensions et la segmentation des données selon ces sous-espaces. Les méthodologies existantes peuvent être divisées en quatre catégories principales : méthodes itératives, algébriques, statistiques et celles basées sur le clustering spectral. Les méthodes itératives, telles que k-subspace (Ho *et al.*, 2003), alternent entre la classification des points dans leurs sous-espaces le plus proches et la réestimation des nouveaux sous-espaces. Cette dernière est faite en appliquant la décomposition en valeurs singulières (SVD) sur les points de chaque cluster. Les méthodes algébriques, telles que celles basées sur la factorisation (Costeira et Kanade, 1998; Kanatani, 2001), utilisent une matrice de similarité obtenue à partir de la factorisation de la matrice des données. Les méthodes statistiques, telles que les approches itératives (Tipping et Bishop, 1999; Sugaya et Kanatani, 2004; Gruber et Weiss, 2004), supposent que la distribution des données à l'intérieur de chaque sous-espace est gaussienne et alternent entre la segmentation des données et l'estimation des sous-espaces. Les méthodes basées sur le clustering spectral, telles que (Yan et Pollefeys, 2006; Goh et Vidal, 2007; Zhang *et al.*, 2012; Zelnik-

Manor et Irani, 2003), utilisent les informations autour de chaque point pour construire les similarités entre chaque paire de points de données. La segmentation des données est alors obtenue en appliquant le clustering spectral sur la matrice de similarité.

Les méthodes de subspace clustering itératives et statistiques nécessitent généralement de connaître le nombre et les dimensions des sous-espaces, et peuvent être sensibles à une étape d'initialisation. Les approches algébriques sont manifestement correctes lorsque les sous-espaces sont indépendants, mais échouent lorsque cette hypothèse est violée (Costeira et Kanade, 1998; Kanatani, 2001). De plus, ils sont sensibles au bruit et aux valeurs aberrantes (Vidal *et al.*, 2005). Les approches basées sur le clustering spectral ont des difficultés à traiter les points proches de l'intersection de deux sous-espaces. En effet, le voisinage d'un point peut contenir des points appartenant à différents sous-espaces. De plus, ils sont sensibles aux choix du nombre des voisins pour construire les similarités entre les points de données (Yan et Pollefeys, 2006; Goh et Vidal, 2007; Zhang *et al.*, 2012; Zelnik-Manor et Irani, 2003).

Les méthodes récentes de subspace clustering, basées sur le clustering spectral, reposent principalement sur l'autoreprésentativité des observations (Chen et Lerman, 2009; Vidal, 2009; Elhamifar et Vidal, 2010; Soltanolkotabi et Candes, 2012; Liu *et al.*, 2010; Favaro *et al.*, 2011). Ces méthodes utilisent une nouvelle approche qui vise à trouver une représentation éparse ou avec un faible rang des données à partir des données elles mêmes. Un algorithme est utilisé pour construire un graphe de similarité à partir duquel la segmentation des données est obtenue. L'avantage avec ces méthodes est qu'elles n'ont pas besoin de connaître les dimensions et le nombre de sous-espaces à priori. L'algorithme Sparse Subspace clusetring [1] (SSC)

1. Dans ce mémoire, pour les besoins de la discussion, nous utilisons les nominations en anglais, des modèles de subspace clustering, comme elles sont indiquées dans la littérature étudiée.

(Elhamifar et Vidal, 2013), se base sur les techniques de représentation éparse des données. L'idée derrière cet algorithme est appelée la propriété d'autoreprésentation (*self-expression*) des données. Cette propriété permet d'exprimer chaque point dans son sous-espace correspondant en tant qu'une combinaison linéaire des autres points dans le même sous-espace. Dans le cadre de ce mémoire, nous utilisons cette propriété pour identifier les sous-espaces. Étant donné n instances de données X non étiquetées avec une dimensionnalité d, $X = [x_1, ..., x_n] \in \mathbb{R}^{d \times n}$ tirées de k sous-espaces $S_1, ..., S_k \subset \mathbb{R}^{d'}$, l'objectif est de trouver ces k sous-espaces en deux étapes. La première étape, la plus importante, consiste au calcul d'une matrice d'affinité de taille n^2 entre tous les points de données. Telle que seuls les points du même sous-espace vectoriel sont connectés. La deuxième étape consiste à appliquer un clustering spectral sur cette matrice d'affinité.

Au-delà du fait que plusieurs approches de subspace clustering ont été proposées, leurs applicabilités à des données réelles sont très limitées. En pratique, les données ne sont pas forcément représentées à partir de sous-espace linéaire. Dans l'exemple du regroupement d'images de visage, la réflectance est généralement non lambertienne et la pose du sujet varie souvent. Dans ces conditions, les images de visage d'un sujet se trouvent plutôt dans un sous-espace non linéaire.

Récemment, d'autres recherches se sont intéressées aux réseaux de neurones profonds. Ces derniers permettent de prendre en considération des données qui résident dans des sous-espaces non linéaires de faible dimension. Ils ont montré un succès remarquable dans diverses applications telles que la reconnaissance vocale, le traitement du langage naturel et la classification d'image (Dahl *et al.*, 2011; Krizhevsky *et al.*, 2012; Collobert *et al.*, 2011). Différentes couches d'unité de traitement non linéaire sont utilisées pour l'extraction des attributs caractérisant une donnée. Au niveau de chaque couche, une nouvelle représentation des données est obtenue. Chaque couche utilise une fonction d'activation non linéaire et prend

en entrée la sortie de la couche précédente. Ainsi, en empilant ces couches non linéaires ensemble, l'apprentissage profond donne de meilleures représentations des données. Différents travaux se sont intéressés à apprendre de nouvelles et meilleures représentations des données pour le clustering (Xie *et al.*, 2016; Huang *et al.*, 2014), en proposant des méthodes qui simultanément projettent les données dans un espace de faible dimension et optimisent la précision du clustering. Aussi, des solutions sont proposées en combinant l'apprentissage profond avec le subspace clustering (*Deep Subspace Clustering*) (Ji *et al.*, 2017; Peng *et al.*, 2020). Par ce fait, le subspace clustering peut bénéficier du succès des réseaux de neurones profonds. De plus, les performances du subspace clusetring peuvent être améliorées en combinant les avantages des méthodes existantes du subspace clustering avec celles de l'apprentissage profond.

Les recherches les plus récentes utilisent les auto-encodeurs (AEs). Ces derniers ont prouvé leurs efficacités dans la réduction de dimension. Plus précisément, l'extraction des caractéristiques les plus pertinentes dans le but de mieux représenter les données originales. L'encodeur permet de générer, dans un espace latent, une représentation compressée Z d'une donnée X, en ne conservant que les informations pertinentes et nécessaires pour l'analyse et l'apprentissage. Le décodeur essaye, à partir de la représentation Z, d'avoir la meilleure reconstruction possible $\widehat{X}$ de la donnée d'entrée X. Le but des auto-encodeurs est d'optimiser la qualité de la reconstruction en minimisant la fonction coût : $\left\|X - \hat{X}\right\|_2^2$. Dans le cadre du subspace clustering, les auto-encodeurs sont utilisés en introduisant une nouvelle couche d'autoreprésentation (self-expression) entre l'encodeur et le décodeur, comme illustrer dans la figure 1.2. L'idée est d'imiter la propriété d'autoreprésentation qui s'est avérée efficace dans le clustering traditionnel des sous-espaces (Ji *et al.*, 2017).

Différentes techniques et méthodologies de subspace clustering ont évoluées à tra-

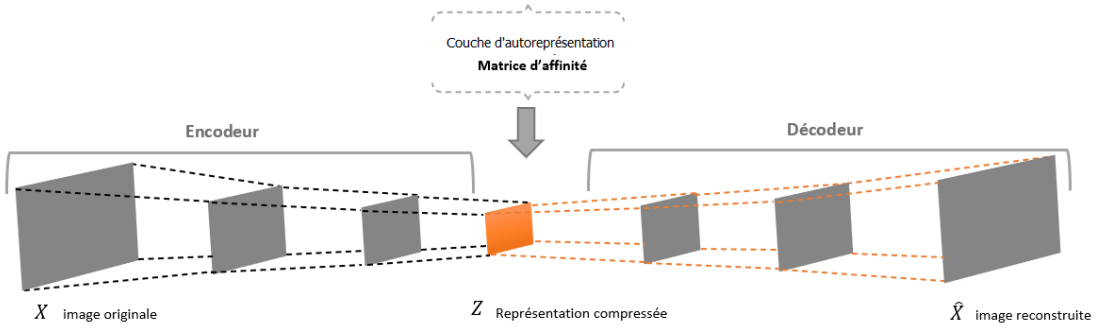


FIGURE 1.2 – Architecture du clustering des sous-espaces profond.

vers plusieurs travaux. En effet, de nouvelles techniques ont été proposées afin de prendre en considération des données plus complexes (avec plus d'instances et de dimensions). L'utilisation de la technique d'autoreprésentation et la matrice d'affinité pour le clustering donne de bons résultats. Cependant, le calcul reste très coûteux avec la construction et le traitement de grandes matrices. Dans le cadre de notre travail, nous nous intéressons à la problématique de la scalabilité du subspace clustering. Les méthodes de subspace clustering ne doivent pas seulement identifier les clusters, mais également trouver les sous-espaces pour chaque cluster. Dans les données de haute dimensionnalité, le nombre des sous-espaces possibles est très grand, nécessitant des algorithmes de recherche efficaces. Les méthodes qui se basent sur la technique d'autoreprésentation ont une complexité $O(n^3)$. Ces méthodes apprennent une matrice d'affinité et effectuent ensuite un clustering spectral. Les deux étapes souffrent d'une complexité temporelle et spatiale élevée, ce qui rend difficile le regroupement de grands ensembles de données. Les méthodes standards, comme le k-subspace clustering, possèdent une complexité $O(n^2)$ pour le calcul de la décomposition en valeurs singulières (SVD) mais sont généralement moins performantes que les méthodes qui se basent sur la technique d'autoreprésentation. C'est pour cela que les articles les plus récents utilisent la technique d'autoreprésentation bien qu'elle soit moins scalable en ce qui concerne

le temps et la mémoire.

De ce fait, dans le cadre de notre travail, nous nous intéressons à traiter la problématique de la scalabilité du subspace clustering en réduisant ce coût. Plusieurs travaux de recherches existent afin de répondre à cette problématique. Toutefois, nous avons remarqué qu'il n'y avait pas de travaux traitant la scalabilité du subspace clustering dans le contexte de l'apprentissage profond. Le seul travail trouvé, intitulé Scalable Deep k-Subspace Clustering (Zhang *et al.*, 2018), contourne la construction de la matrice d'affinité. Il propose un réseau de neurones profond adapté à la méthode de k-subspace clustering. Cette approche possède une complexité quadratique nécessaire pour calculer la décomposition en valeurs singulières (SVD). Dans le cadre de notre travail, nous nous intéressons à proposer une approche capable d'effectuer le subspace clustering profond en un temps linéaire par rapport à la taille des données. Principalement, réduire la complexité relative au calcul de la matrice d'affinité et à celle du clustering spectral.

1.2 Motivation :

Dans la littérature, plusieurs techniques existent afin de répondre à la problématique de la scalabilité du subspace clustering. Parmi elles, les techniques d'échantillonnage. Ces dernières se basent sur l'idée qu'avec un grand nombre de données il y a beaucoup de redondance. Par conséquent, un petit nombre d'instances sont suffisantes pour identifier les sous-espaces. Ces techniques tentent de sélectionner un sous-ensemble plus petit de données, de telle sorte qu'il soit le plus représentatif possible de l'ensemble de données originale. Théoriquement, pour chaque sous-espace linéaire, il suffit de fournir un nombre d'échantillons indépendants (formant une famille indépendante) égale à la dimension de ce sous-espace. Dans certains travaux, la sélection du sous-ensemble de données est ciblée et doit répondre à

certains critères. Ces techniques souffrent d'un coût de calcul élevé, car elles sont résolues avec une manière d'optimisation itérative. De plus, certaines supposent que les sous-espaces sont indépendants (Wang *et al.*, 2014; Matsushima et Brbic, 2019; You *et al.*, 2018). Cette hypothèse est trop restrictive et ne peut pas être garantie pour les données du monde réel (avec des sous-espaces pas suffisamment séparés et avec des données pas forcement bien distribuées) (Parsons *et al.*, 2004). Dans d'autres travaux (Chen et Cai, 2011; Kang *et al.*, 2020; Li et Zhao, 2017), cette sélection est faite aléatoirement. Toutefois, ce choix aléatoire des données ne garantit pas d'avoir, pour un sous-espace donné, un nombre d'échantillons indépendants égale à la dimension de ce sous-espace (la taille de dimension de chaque sous-espace étant inconnu). Par conséquent, le sous-ensemble sélectionné peut ne pas être représentatif des données originales pour un meilleur clustering. Ainsi, le coût peut être réduit au détriment de la précision du clustering.

Dans d'autres travaux, d'autres techniques sont utilisées pour réduire la complexité du subspace clustering (You *et al.*, 2016b,a; Chen *et al.*, 2020; Lim *et al.*, 2020). Également, garantir la préservation des sous-espaces et traiter le problème de connectivité dans le graphe d'affinité. Certaines arrivent à atteindre ces garanties, mais sous certaines conditions théoriques. Toutefois, ces conditions ne reflètent pas les données du monde réel. De plus, les algorithmes utilisés souffrent toujours d'une complexité temporelle élevée, car ils sont résolus par des méthodes d'optimisation itératives.

Les travaux les plus récents sur la scalabilité du subspace clustering se concentrent sur le regroupement des sous-espaces linéaires. Cependant, en pratique, les données ne résident pas nécessairement dans des sous-espaces linéaires (Parsons *et al.*, 2004). Pour cette raison, nous nous intéressons au subspace clustering profond où les données sont transformées de manière non linéaire. Les solutions existantes introduisent la couche d'autoreprésentation entre l'encodeur et le décodeur. Cette

couche fournit dans un espace latent un moyen simple et efficace pour apprendre la matrice d'affinité entre tous les points de données. Néanmoins, en s'appuyant sur l'ensemble de données pour créer la matrice d'affinité, cette méthode ne réduit pas le coût du subspace clustering. Au mieux de nos connaissances, le seul travail essayant d'améliorer la complexité relative à la construction de la matrice d'affinité dans le contexte d'apprentissage profond est (Kheirandishfard *et al.*, 2020). Ce modèle nécessite beaucoup moins de paramètres de réseau et permet d'incorporer certaines normes de régularisation, par exemple la norme nucléaire (la somme des valeurs singulières), avec un coût de calcul très inférieur au modèle standard (Ji *et al.*, 2017; Peng *et al.*, 2020). Cependant, la complexité de cette solution n'est pas linéaire.

Les résultats prometteurs des travaux existants dans le contexte d'apprentissage profond, ainsi que le manque de travaux traitant la scalabilité dans ce contexte à susciter notre intérêt. Dans le cadre de notre travail, nous nous intéressons à réduire le coût du subspace clustering. À cette fin, nous utilisons les auto-encodeurs ainsi que la couche d'autoreprésentation. Notre objectif est de réduire la complexité relative au calcul de la matrice d'affinité ainsi que celle du spectral clustering vers une complexité linéaire par rapport à la taille des données.

1.3 Contributions :

Dans ce travail nous traitons la scalabilité du subspace clustering dans le contexte d'apprentissage profond. Le modèle présenté dans ce mémoire est la première proposition à cette problématique en utilisant la propriété d'autoreprésentation du subspace clustering. Ci-dessous, les éléments importants de notre solution :

1. Nous proposons un algorithme efficace pour réduire la complexité du subspace clustering. Les deux étapes du subspace clustering souffrent d'une

complexité temporelle et spatiale élevée de $O(n^3)$, ce qui rend difficile le regroupement de grands ensembles de données. Nous réduisons la complexité relative à la construction de la matrice d'affinité ainsi que celle du clustering spectral de $O(n^3)$ vers une complexité linéaire $O(n)$.

2. Notre algorithme garantit une convergence vers les sous-espaces optimaux. Nous considérons un sous-ensemble de points de repère plus petit que l'ensemble de données initiales pour apprendre les similarités entre tous les points de données. Ce sous-ensemble est beaucoup plus petit que la taille des données initiales. Ce qui nous permet d'apprendre les similarités à moindre coût.

3. Nous proposons une nouvelle méthode basée sur l'apprentissage profond. Ce dernier permet de prendre en considération des données complexes qui ne résident pas initialement dans des sous-espaces linéaires. Nous utilisons l'auto-encodeur et la couche d'autoreprésentation. Ainsi, nous définissons une nouvelle fonction objective permettant de réduire la complexité du subspace clustering.

4. Nous avons mené des expériences sur des données synthétiques et réelles. Les résultats ont démontré que notre méthode fournit une amélioration significative de la précision du clustering par rapport aux solutions scalables existantes. De plus, les expériences sur les données synthétiques confirment la capacité de notre approche à identifier effectivement les sous-espaces. Également, nous constatons que nos résultats sont compétitifs aux résultats obtenus par les travaux du deep subspace clustering. Ces derniers, il faut le rappeler, utilisent la similarité entre chaque point et tous les autres points de données lors de la construction de la matrice d'affinité.

1.4 Structure et organisation du mémoire :

Le reste de ce document est organisé comme suit : le chapitre 2 présente une revue des principaux travaux traitant la scalabilité du subspace clustering. Premièrement, nous présentons les différents algorithmes proposés qui n'utilisent pas un apprentissage profond. Par la suite, nous présentons le seul travail (Deep k-Subspace Clustering) traitant la scalabilité du subspace clustering dans le contexte d'apprentissage profond. Également, nous présentons quelques modèles proposés du subspace clustering profond non scalable. Le chapitre 3 présente notre solution à la problématique de la scalabilité du subspace clustering dans le contexte d'apprentissage profond. Plus précisément, nous présentons notre nouvelle fonction objective ainsi que la méthode utilisée pour le regroupement des données. Enfin, nous décrivons la stratégie d'entraînement utilisée pour notre modèle. Le chapitre 4 présente une évaluation empirique de notre approche sur des données synthétiques et réelles. La performance de notre approche est comparée aux différentes méthodes de subspace clustering scalable ainsi qu'à quelques méthodes du subspace clustering profond. En dernier, le chapitre 5 conclut ce mémoire.

CHAPITRE II

REVUE DE LA LITTÉRATURE

Ce chapitre présente un aperçu de quelques travaux existants traitant la problématique de la scalabilité du subspace clustering. En premier lieu, nous présenterons quelques notions sur les concepts du clustering et le subspace clustering. Par la suite, nous présenterons quelques techniques scalables pour le subspace clustering. Plus précisément, nous catégorisons les techniques existantes selon leur mode de fonctionnement en deux catégories : (1) scalabilité du subspace clustering linéaire, et (2) scalabilité du subspace clustering profond. Premièrement, nous présenterons les différents algorithmes proposés qui n'utilisent pas un apprentissage profond. Pour la deuxième catégorie, nous présenterons le seul travail (Deep k-Subspace Clustering) traitant la scalabilité du subspace clustering dans le contexte d'apprentissage profond. Par la suite, nous présenterons quelques techniques non scalables du subspace clustering profond. Par ce fait, nous montrons comment le subspace clustering peut bénéficier du succès des réseaux de neurones profonds. De plus, nous montrons comment les performances du subspace clusetring peuvent être améliorées en combinant les avantages des méthodes existantes du subspace clustering avec celles de l'apprentissage profond.

2.1 Concepts et notions :

Plusieurs algorithmes de clustering existent dans la littérature. Le plus populaire est l'algorithme k-moyennes (k-means). L'objectif des algorithmes de clustering, d'une façon générale, est de grouper des éléments similaires dans des clusters. L'algorithme k-means par exemple, compare le degré de similarité entre les différentes données. Ainsi, deux données similaires auront une petite distance de similarité, alors que deux données différentes auront une distance plus grande. La littérature déborde de définition de distance. Parmi les plus connus, la distance euclidienne. Toutefois, avec un nombre très grand de variables, le calcul devient très long. Par conséquent, ce type de méthode à un impacte négatif sur les performances. De plus, les métriques de distance euclidienne prennent une forme convexe par rapport aux clusters sous-jacents. De toute évidence, ceci peut avoir un impact sur la qualité du clustering dans des ensembles de données arbitraires.

(a) Forme non convexe. (b) Forme convexe.

FIGURE 2.1 – Exemple de forme de clusters.

Contrairement aux méthodes traditionnelles basées sur la distance, les méthodes de spectral clustering sont moins restrictives. Ces dernières ne font pas d'hypothèses sur les formes des clusters. Elles peuvent être appliquées dans des scénarios plus complexes, tels que des formes spirales entrelacées ou d'autres formes non

linéaires arbitraires comme illustrer dans la figure 2.1. Les méthodes de spectral clustering se basent sur un graphe de similarité entre tous les points de données. Pour décrire le graphe de similarité, une matrice d'adjacence (matrice d'affinité) est calculée. Parmi les méthodes pour calculer cette matrice d'affinité, celles qui se basent sur les k plus proches voisins. Une autre méthode consiste à connecter tous les points avec une similarité positive, en utilisant une fonction de similarité gaussienne. Par la suite, comme une deuxième étape, le graphe Laplacien est calculé en utilisant la matrice d'affinité pour le clustering spectral final.

Bien que les performances du spectral clustering dépassent celles des méthodes traditionnelles, elles se dégradent avec le calcul de grandes matrices d'affinités. Les algorithmes de spectral clustering doivent bien évoluer en ce qui concerne le nombre d'instances et de dimensions dans de grands ensembles de données. De plus, les méthodes de spectral clustering supposent que les données sont représentées à partir d'un seul sous-espace. Cependant, en pratique, les données pourraient être tirées de plusieurs sous-espaces, et l'appartenance à ces sous-espaces peut être inconnue. Dans ce cas, le problème du clustering est réduit au problème d'assigner chaque donnée à son propre sous-espace. Pour répondre à ces problématiques, les méthodes de subspace clustering ont été proposées pour trouver des sous-ensembles de dimensions pertinentes pour chaque sous-espace (Parsons *et al.*, 2004; Kriegel *et al.*, 2009; Vidal, 2011; Elhamifar et Vidal, 2013; Vidal et Favaro, 2014). La figure 2.2 montre un exemple de données avec quatre sous-espaces, chacun appartient à deux dimensions.

Les méthodes récentes de subspace clustering se basent sur l'hypothèse que les points de données de haute dimensionnalité sont répartis autour de plusieurs sous-espaces linéaires de faible dimension. Spécifiquement, ces méthodes consistent, dans une première étape, à construire d'abord une matrice d'affinité telle que seuls les points du même sous-espace sont connectés. Par la suite, comme une

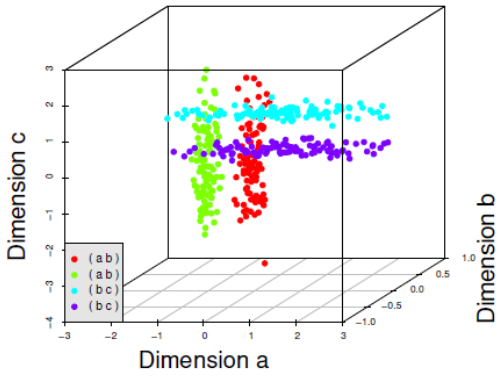


FIGURE 2.2 – Exemple de sous-espaces vectoriel (Parsons *et al.*, 2004).

deuxième étape, le spectral clustering est appliqué sur cette matrice d'affinité. La clé du succès de cette approche réside dans la matrice d'affinité. Soit les données $\{X_i\}_{i=1,..,n}$ formées à partir de plusieurs sous-espaces linéaires $\{S_i\}_{i=1,..K}$, un point peut être exprimé dans un sous-espace en tant qu'une combinaison linéaire des autres points dans le même sous-espace. Dans la littérature, cette propriété est appelée autoreprésentation (Self-expression). Cette dernière est représentée par l'équation $X = XC$, où C est la matrice de coefficient. Sous l'hypothèse que les sous-espaces sont indépendants, $C_{ij} \neq 0$ si le point x_i et le point x_j se trouvent dans le même sous-espace. La matrice de coefficient est utilisée pour construire la matrice d'affinité $A = (|C| + |C|^{\intercal})/2$ pour le spectral clustering. Cette idée est formulée comme un problème d'optimisation comme suit (Ji *et al.*, 2017) :

$$\min_{C} \|C\|_p \quad s.t. \quad X = XC, \quad (diag(C) = 0) \tag{2.1}$$

où $\|.\|_p$ représente une norme matricielle quelconque, et la contrainte diagonale sur C empêche des solutions triviales pour les normes induisant la sparsité, telles que la norme l_1. Diverses normes pour C ont été proposées dans la littérature, la

norme l_1 dans Sparse Subspace Clustering (SSC) (Elhamifar et Vidal, 2013), la norme nucléaire dans Low Rank Representation (LRR) (Liu *et al.*, 2010, 2012) et Low Rank Subspace Clustering (LRSC) (Favaro *et al.*, 2011; Vidal et Favaro, 2014), et la norme de Frobenius dans Least-Squares Regression (LSR) (Lu *et al.*, 2012) et Efficient Dense Subspace Clustering (EDSC) (Ji *et al.*, 2014). Le problème d'optimisation précédent, dans l'équation (2.1), est souvent assoupli en minimisant la fonction objective suivante :

$$\min_{C} \|C\|_p + \frac{\lambda}{2} \|X - XC\|_F^2 \quad s.t. \quad (dig(C) = 0) \tag{2.2}$$

Comme nous pouvons le constater, les techniques citées ci-dessus se basent principalement sur un calcul de similarité entre tous les points de données. Différentes techniques et méthodes ont évolué à travers plusieurs travaux. Par conséquent, de nouvelles techniques ont été proposées afin de prendre en considération des données plus complexes (avec plus d'instances et de dimensions). Dans le cadre de notre travail, nous nous intéressons à la problématique de la scalabilité du subspace clustering. En effet, les deux étapes du subspace clustering (le calcul de la matrice d'affinité, et le clustering spectral) souffrent d'une complexité temporelle et spatiale élevée, ce qui rend difficile le regroupement de grands ensembles de données. Précisément, comme mentionnées précédemment, les méthodes de subspace clustering ne doivent pas seulement identifier les clusters, mais également les sous-espaces pour chaque cluster. Dans les données de haute dimensionnalité, le nombre des sous-espaces possibles est énorme, nécessitant des algorithmes de recherche efficaces. Ainsi, les méthodes de subspace clustering doivent être mises à l'échelle par rapport à la taille de données et à la dimensionnalité des sous-espaces où se trouvent les clusters.

Nous présenterons ci-dessous différentes techniques et méthodologies traitant la

scalabilité du subspace clustering.

2.2 Techniques de scalabilité du subspace clustering linéaire :

Nous catégorisons les techniques existantes selon leur mode de fonctionnement en deux catégories : (1) celles basées sur des techniques d'échantillonnage, (2) celles qui n'utilisent pas l'échantillonnage.

2.2.1 Techniques d'échantillonnage :

Les techniques d'échantillonnage assument qu'il y a beaucoup de redondance dans les données. Par conséquent, un petit sous-ensemble de données est suffisant pour reconstruire les sous-espaces sous-jacents. Ce petit ensemble de données est choisi selon un type de sélection. Cette dernière peut se faire : (1) aléatoirement, ou (2) avec une sélection ciblée selon certains critères.

Techniques de sélection aléatoire :

Dans un contexte de spectral clustering, **Large Scale Spectral Clustering with Landmark-Based Representation** [1] **(LSC-R)** (Chen et Cai, 2011) sélectionnent p points de repère à partir de n points de données ($p \ll n$). La sélection est faite aléatoirement ou en utilisant k-means. Par la suite, une plus petite matrice d'affinité $Z \in \mathbb{R}^{p \times n}$ est calculée entre tous les points de données et les points sélectionnés. En ayant la matrice $U \in \mathbb{R}^{d \times p}$ des points fixes sélectionnés, la matrice Z doit vérifier l'équation suivante : $\hat{x_i} = \sum_{j=1}^{p} z_{ij} u_j$, permettant de reconstruire le point x_i à partir des points sélectionnés et la matrice Z calculée. Avec l'hypothèse

1. Nous rappelons que dans ce mémoire, nous utilisons les nominations en anglais, des modèles de subspace clustering, comme elles sont indiquées dans la littérature étudiée.

que z_{ij} devrait être plus grand si x_i est plus proche de u_j ; z_{ij} est mis à zéro si u_j n'est pas parmi les $r(\leq p)$ voisins les plus proches de x_i. Avec cette restriction, la matrice d'affinité Z devient éparse. De ce fait, en utilisant une sous-matrice $U_{<i>} \in \mathbb{R}^{d\times r}$ composée des r plus proches points de x_i, z_{ij} est calculée avec une complexité d'ordre $O(pn)$ comme suit :

$$z_{ij} = \frac{K_h(x_i, u_j)}{\sum_{j'\in U_{<i>}} K_h(x_i, u_{j'})} \quad , \quad j \in U_{<i>}. \tag{2.3}$$

où $K_h(x_i, u_j) = exp(-\left\|x_i - u_j\right\|^2 / 2h^2)$ est une fonction noyau (kernel en anglais), avec un paramètre de lissage h qui régit le degré de lissage de l'estimation.

Pour réduire la complexité du clustering, la décomposition en valeurs singulières (SVD) est appliquée sur Z, avec une complexité d'ordre $O(p^3 + p^2 n)$.

Large-scale Subspace Clustering by Fast Regression Coding (FRC) (Li et Zhao, 2017) indiquent que pour des données $X \in \mathbb{R}^{d\times n}$, il existe une fonction non linéaire $f(.;\theta) \in \mathbb{R}^{m\times n}$ telle que $X = Yf(X;\theta)$, où $Y \in \mathbb{R}^{d\times m}$ est un petit ensemble de données de taille m sélectionné aléatoirement à partir de X, et le paramètre θ est appris à partir des données Y comme suit : $Y = YZ$, $Z = f(Y, \theta)$. Le but est d'apprendre une fonction qui permet de calculer une matrice d'affinité Z des données du sous-ensemble Y. Pour apprendre θ un réseau de neurones composé de trois couches est utilisé avec l'optimisation suivante :

$$\min_{Z,W_1,W_2} \left\|Y - YZ\right\|_F^2 + \alpha \left\|Z\right\|_F^2 \quad s.t. \quad Z = W_2 g(W_1 Y) \tag{2.4}$$

où $g(.)$ est une fonction d'activation non linéaire, W_1 et W_2 sont les poids du réseau de neurones.

Une fois le réseau entraîné et les poids W_1 et $W2$ obtenus, la matrice Z_X pour tous les points de données X est alors calculée, avec une complexité totale d'ordre

$O(nm)$, en utilisant la fonction suivante : $Z_X = f(X, \theta) = W_2 g(W_1 X)$. Pour le clustering des données, la même méthode que celle dans (Chen et Cai, 2011) est appliquée.

Dans un contexte de clustering des sous-espaces dans des vues multiples [2] (Multi-Vue Subspace Clustering (MVSC)), **Large-Scale Multi-View Subspace Clustering in Linear Time (LMVSC)** (Kang *et al.*, 2020) appliquent le même principe de sélection des points de repère dans (Chen et Cai, 2011) cité ci-dessus. Contrairement à (Chen et Cai, 2011), qui utilisent une technique à base de distance, les similarités sont obtenues à partir des données, en minimisant la fonction suivante :

$$\min_{Z^i} \sum_{i=1}^{v} \left\| X^i - A^i (Z^i)^\intercal \right\|_F^2 + \alpha \left\| Z^i \right\|_F^2 \quad s.t \quad 0 \leq Z^i, (Z^i)^\intercal 1 = 1 \tag{2.5}$$

où A^i représente les m points de repère fixes obtenus en appliquant k-means sur les points X^i de la vue i. Z^i représente la matrice de similarité de la vue i entre tous les points X^i et les points A^i. La matrice d'affinité $\overline{Z} \in \mathbb{R}^{n \times mv}$ pour les v vues est calculée avec cette équation : $\overline{Z} = \frac{1}{\sqrt{v}}[Z^1, ..., Z^i, ..., Z^v]$. Pour le clustering des données, la même méthode que celle dans (Chen et Cai, 2011) est appliquée.

Structured Graph Learning for Scalable Subspace Clustering : From Single View to Multiview (SGL) (Kang *et al.*, 2021a) utilisent la même technique utilisée dans (Kang *et al.*, 2020). Dans le but d'améliorer le résultat du clustering, son objectif est de pouvoir apprendre la matrice $Z \in \mathbb{R}^{n \times m}$ ($m \ll n$) avec k composants disjoints connectés (k nombre de clusters). Pour obtenir cette représentation de la matrice Z, les graphes bipartis sont utilisés de telle sorte que

2. Un article peut être écrit dans différentes langues. Par conséquent, il est intéressant de développer une méthode d'apprentissage sur des vues multiples à fin d'incorporer les informations utiles provenant des autres langues

Z soit définie par $S = \begin{bmatrix} & \mathrm{Z} \\ \mathrm{Z}^\intercal & \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}$. Ainsi, le Laplacien normalisé est exprimé sous la forme : $L = I - D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$. Selon la théorie de graphe spectral, pour que les n points de données et les m points de repère soient groupés en k clusters le $rank(L)$ doit être égal à $(m+n)-k$. Cette contrainte est exprimée dans le problème (2.5) comme suit :

$$\begin{aligned} &\min_{Z^i} \sum_{i=1}^{v} \left\| X^i - A^i (Z^i)^\intercal \right\|_F^2 + \alpha \left\| Z^i \right\|_F^2 \\ s.t. \quad &0 \leq Z^i, (Z^i)^\intercal 1 = 1, rank(L) = (m+n) - k \end{aligned} \tag{2.6}$$

Pour rendre la contrainte plus facile à résoudre, en se basant sur (Kang *et al.*, 2021b), le problème dans (2.6) est remplacé par :

$$\begin{aligned} \min_{Z^i} \sum_{i=1}^{v} \left\| X^i - A^i (Z^i)^\intercal \right\|_F^2 + \alpha \left\| Z^i \right\|_F^2 +_r (F^\intercal L F) \\ s.t \quad 0 \leq Z^i, (Z^i)^\intercal 1 = 1, F^\intercal F = I \end{aligned} \tag{2.7}$$

où $F \in \mathbb{R}^{(n+m)\times k}$. Ce problème est résolu par la stratégie d'optimisation alternée. Z et F sont résolus de manière alternée, en fixant l'un d'eux puis mettre à jour l'autre.

Ces techniques d'échantillonnage tentent de sélectionner un sous-ensemble plus petit de données, de telle sorte qu'il soit le plus représentatif possible de l'ensemble de données originales. Théoriquement, pour chaque sous-espace linéaire, il suffit de fournir un nombre d'échantillons indépendants (formant une famille indépendante) égal à la dimension de ce sous-espace. L'utilisation d'un petit sous-ensemble de données, au lieu de toutes les données, a permis de réduire la complexité relative au calcul de la matrice d'affinité ainsi qu'à celle du spectral clustering. Toutefois,

ce choix aléatoire des données ne garantit pas d'avoir, pour un sous-espace donné, un nombre d'échantillons indépendants égal à la dimension de ce sous-espace (la taille de dimension de chaque sous-espace étant inconnu). Par conséquent, le sous-ensemble sélectionné peut ne pas être représentatif des données originales pour un meilleur clustering. Ainsi, le coût est réduit au détriment de la précision du clustering.

Techniques de sélection ciblée :

Exact Subspace Clustering in Linear Time (Wang *et al.*, 2014) proposent de traiter la problématique de la scalabilité en trois étapes : (1) : sélection d'un sous-ensemble de données, (2) : clustering du sous-ensemble sélectionné, (3) : classification de toutes les données non sélectionnées dans les clusters trouvés à la deuxième étape. Cette proposition repose sur deux hypothèses : (1) les k sous-espaces $S_1, ..., S_k$ sont linéairement indépendants tel que $S_i \cap S_j = 0$ pour tout $i \neq j$, (2) et pour tout sous-espace $S_{i \in k}$, l'ensemble des instances appartenant à un sous-espace ne peut pas être partitionné en deux ensembles distincts. Ces hypothèses sont appelées respectivement, indépendance des sous-espaces et dépendance interne d'un sous-espace. Un algorithme de sélection est proposé en prenant en considération ces deux hypothèses. Le but de cet algorithme est de faire en sorte que les deux hypothèses sont respectées par le sous-ensemble sélectionné. L'algorithme sélectionne r instances, r est la somme des dimensions des sous-espaces, avec une complexité d'ordre $O(nr^2)$ (n est le nombre d'instances totales). Par la suite, l'algorithme forme k ensembles distincts à partir des données sélectionnées. Pour l'étape du clustering du sous-ensemble sélectionné, les auteurs se basent sur le théorème qui garantit que si des données X satisfont les hypothèses 1 et 2, alors les instances de données sélectionnées satisfont également les mêmes hypothèses. Par conséquent, ils proposent d'appliquer une méthode de subspace clustering,

Sparse Subspace Clustering (SSC) (Vidal, 2009; Adler *et al.*, 2012) où Low-Rank Representation (LRR) (Liu *et al.*, 2010; Lin *et al.*, 2011), sur les données sélectionnées pour identifier les clusters. Comme une dernière étape, les données non sélectionnées sont affectées à l'un des clusters trouvés en utilisant une régression linéaire.

Scalable Exemplar-based Subspace Slustering on Class-Imbalanced Data (ESC) (You *et al.*, 2018) et **Selective Sampling-based Scalable Sparse Subspace Clustering (S^5C)** (Matsushima et Brbic, 2019) proposent un algorithme de sélection d'un sous-ensemble de données de telle sorte que ce sous-ensemble contient suffisamment d'instances de chaque sous-espace. Par la suite, un calcul de similarités est fait entre tous les points de données et le sous-ensemble sélectionné. Pour garantir la préservation des sous-espaces [3], les auteurs supposent que les sous-espaces sont indépendants. L'algorithme sélectionne un sous-ensemble $|X_0| = k$ qui doit représenté au mieux toutes les données $|X| = n$, avec une complexité d'ordre $O(kn)$. Un point $x \in X$ est choisi et rajouté itérativement à l'ensemble X_0 de telle sorte qu'il minimise la nouvelle fonction d'autoreprésentation suivante :

$$\min_{c_j \in \mathbb{R}^n} \|c_j\|_1 + \frac{\lambda}{2} \left\| x_j - \sum_{i:x_i \in X_0} c_{ij} x_i \right\|_2^2 \tag{2.8}$$

Cette nouvelle fonction permet d'écrire chaque point de données comme une combinaison linéaire des autres points dans X_0. Une fois l'ensemble sélectionné par l'algorithme, il est utilisé pour calculer la matrice de coefficient C entre tous les points et l'ensemble de points sélectionnés. Plus précisément, pour chaque

3. Il n'y a pas de connexions entre les points de différents sous-espaces, c'est-à-dire que, pour une matrice d'affinité C, $c_{ij} \neq 0$ seulement si x_i et x_j sont dans le même sous-espace.

$x_j \in X$, c_j est calculé en optimisant la nouvelle fonction d'autoreprésentation ci-dessus (2.8). Pour le spectral clustering, en utilisant l'approche du plus proche voisin, la matrice d'affinité A entre tous les points de données est calculée comme suit : $A_{ij} = 1$ si c_j est t-plus proche voisin de c_i ; $A_{ij} = 0$ sinon. Les auteurs dans (Matsushima et Brbic, 2019), quant à eux, ne font aucune supposition sur l'indépendance des sous-espaces et proposent un algorithme de sélection basé sur un calcul stochastique du sou-gradient. Pour l'étape de clustering, ils effectuent une décomposition des valeurs propres en utilisant une itération orthogonale.

La sélection ciblée permet d'avoir un sous-ensemble de données qui représente mieux les données originales que la sélection aléatoire. Cependant, les algorithmes de sélection souffrent toujours d'une complexité temporelle élevée, car ils sont résolus par des méthodes d'optimisation itératives. De plus, certains supposent que les données provenant de différents sous-espaces satisfont certaines conditions de séparation et les données d'un même sous-espace sont bien réparties.

2.2.2 Techniques qui ne se basent pas sur l'échantillonnage :

Scalable Sparse Subspace Clustering by Orthogonal Matching Pursuit (SSC-OMP) (You *et al.*, 2016b) utilisent *orthogonal matching pursuit (OMP)* (Pati *et al.*, 1993) à l'étape d'autoreprésentation pour trouver une représentation éparse des données. OMP résout le problème $min_c \|Ac - b\|_2^2 \quad s.t. \quad \|c\|_0 \leq k$ graduellement, avec une complexité linaire, en sélectionnant une colonne à la fois de $A = [a_1, ...a_m]$ et calcule les coefficients pour la colonne sélectionnée. Pour le subspace clustering le vecteur de coefficient $C_j \in \mathbb{R}^N$ de x_j est calculé en utilisant $OMP(X_{-j}, x_j) \in \mathbb{R}^{N-1}$, avec $C[j] = 0$ où X_{-j} représente les données X sans la données x_j. L'idée d'utiliser OMP pour le subspace clustering avait déjà été envisagée dans (Dyer *et al.*, 2013; You et Vidal, 2015). En revanche, les

principales contributions dans (You *et al.*, 2016b) est de trouver des conditions théoriques sous lesquelles la matrice d'affinité produite préserve les sous-espaces. Les auteurs déterminent que les sous-espaces sont plus préservés, dans le cas où : (1) les sous-espaces sont indépendants, ou (2) ils sont suffisamment séparés et les données sont bien distribuées. Dans le cas contraire où les sous-espaces sont aléatoires, les auteurs supposent que la taille des dimensions des sous-espaces est très inférieure à la taille des dimensions de l'ensemble des données originales.

Oracle Based Active Set Algorithm for Scalable Elastic net Subspace Clustering (EnSC) (You *et al.*, 2016a) proposent une approche pour avoir un équilibre entre la préservation du sous-espace et la propriété de connectivité [4] (Nasihatkon et Hartley, 2011). Une solution serait d'utiliser un mélange de deux normes de régularisation l_1 et l_2 [5] (Wang *et al.*, 2013; Panagakis et Kotropoulos, 2014; Fang *et al.*, 2014) donné par : $r(.) = \lambda \|c\|_1 + \frac{1-\lambda}{2} \|c\|_1^2$. Néanmoins, l'utilisation de ces deux normes se fait au prix d'une complexité de calcul très élevée. Comme solution, (You *et al.*, 2016a) utilisent $r(.)$ et proposent un algorithme itératif pour le subspace clustering. Dans la littérature statistique, le programme d'optimisation utilisant cette régularisation s'appelle le réseau élastique (Elastic Net en anglais). Ce dernier est utilisé pour la sélection de variables dans les problèmes de régression (Zou et Hastie, 2005). L'idée de base derrière l'algorithme est de résoudre une séquence de sous-problèmes à une échelle réduite. L'algorithme proposé exploite le fait que les entrées non nulles de la solution de Elastic Net ($c_j \neq 0$) tombent dans une région appelée oracle. En effet, à chaque itération, un sous-ensemble de données plus petit est utilisé pour le calcul de la matrice

4. Les points de données du même sous-espace forment un composant connecté du graphe d'affinité.

5. La norme l_1 garantit la préservation des sous-espaces sous des conditions théoriques générales. Alors que la norme l_2 améliore la connectivité, mais suppose que les sous-espaces sont préservés que lorsqu'ils sont indépendants.

d'affinité. Ce sous-ensemble grandit à chaque itération, en utilisant une sélection basée sur la région oracle, jusqu'à ce qu'il n'y ait plus de points à rajouter. Pour contrôler la taille de ce sous-ensemble, seul un petit nombre de nouveaux points sont ajoutés.

Stochastic Sparse Subspace Clustering (SSSC) (Chen *et al.*, 2020) exploitent le dropout, une technique utilisée dans l'apprentissage profond pour éviter le surapprentissage (Srivastava *et al.*, 2014; Wan *et al.*, 2013). Dans (Chen *et al.*, 2020), le dropout fait référence à l'élimination ou l'abandon de quelques colonnes de la matrice de données X uniformément au hasard lors du calcul de la matrice d'affinité. Ce qui permet de résoudre un problème d'optimisation impliquant un sous-ensemble de données très petit de l'ensemble de données originales. Plus précisément, un taux d'abandon $0 \leq \delta \leq 1$ est défini, ainsi que des variables aléatoires indépendantes et identiquement distribuées de Bernoulli $\{\xi_i\}_{i=1}^{N}$, avec une distribution de probabilité donnée par :

$$\xi_i = \begin{cases} \frac{1}{1-\delta} & \text{avec une probabilité} \quad 1\text{-}\delta, \\ 0 & \text{avec une probabilité} \quad \delta. \end{cases} \tag{2.9}$$

En multipliant les N variables aléatoires de Bernoulli $\{\xi_i\}_{i=1}^{N}$ aux colonnes de la matrice de données X, l'élimination des colonnes avec la probabilité δ est introduite dans la fonction d'autoreprésentation comme suit :

$$\min_{c_j} \left\| x_j - \sum \xi_i c_{ij} x_i \right\|_2^2 \quad s.t. \quad c_{ij} = 0 \tag{2.10}$$

L'algorithme proposé sélectionne T sous-ensembles $\{S^{(t)}\}_{t=1}^{\intercal}$ en utilisant la probabilité δ. Par la suite, le calcul du vecteur de coefficient c_j de x_j est effectué par la résolution de T sous problèmes parallèlement en utilisant $OMP(\{S^{(t)}\}_{t=1}^{\intercal}, x_j)$

(Pati *et al.*, 1993).

Doubly Stochastic Subspace Clustering (A-DSSC)(Lim *et al.*, 2020) proposent une méthode de subspace clustering qui calcule la matrice d'affinité A en même temps que la matrice de coefficient C. Le but est d'avoir une matrice d'affinité éparse et bien normalisée sans passer par le post-traitement de la matrice de coefficient pour le spectral clustering. Pour ce faire, (Lim *et al.*, 2020) exigent que la matrice A doit être dans un ensemble convexe des matrices doublement stochastiques (Horn et Johnson, 2012). La normalisation doublement stochastique a démontré qu'elle améliore considérablement les performances du spectral clustering (Wang *et al.*, 2016; Nie *et al.*, 2016). Pour ce faire, les deux contraintes suivantes sont appliquées sur la matrice A : (1)$A \geq 0$, (2) contraindre les lignes et les colonnes à avoir la norme l_1. La première contrainte est tirée du fait que la matrice d'affinité doit être positive pour le spectral clustering. La deuxième contrainte est définie selon une forme de normalisation bien établie dans la littérature sur le spectral clustering (Von Luxburg, 2007). Par conséquent, le modèle prend la forme d'un problème d'optimisation sur C et A comme suit :

$$\begin{aligned} \min_{C,A} \quad & \frac{1}{2}\left\|X - XC\right\|_F^2 + \frac{\eta_1}{2}\left\|C\right\|_F^2 \\ & +\eta_3\left\|C\right\|_1 - \langle \eta_1|C|, \eta_2 A\rangle + \frac{\eta_1\eta_2^2}{2}\left\|A\right\|_F^2 \\ s.t.\ & A \geq 0, A1 = 1, A^\intercal 1 = 1, diag(c) = 0 \end{aligned} \tag{2.11}$$

Le problème d'optimisation ci-dessus est résolu en deux étapes. La première étape consiste à calculer la matrice de coefficient C. En initialisant $A = I$, le problème d'optimisation est résolu par l'algorithme proposé dans (Scalable Elastic net Subspace Clustering) (You *et al.*, 2016a) :

$$\min_{C} \quad \frac{1}{2}\left\|X - XC\right\|_F^2 + \frac{\eta_1}{2}\left\|C\right\|_F^2 + \eta_3 + \left\|C\right\|_1 \quad s.t. \quad diag(c) = 0 \tag{2.12}$$

La deuxième étape consiste à calculer la matrice A. En fixant C, la matrice A devient un cas particulier du problème de régularisation quadratique (Blondel *et al.*, 2018; Lorenz *et al.*, 2021) :

$$\min_{A} \quad \langle -|C|, A\rangle + \frac{\eta_2}{2}\left\|A\right\|_F^2 \quad s.t. \quad A \geq 0, A1 = 1, A^\intercal 1 = 1 \tag{2.13}$$

Pour assurer un calcul plus efficace et développer une méthode scalable, la projection doublement stochastique à travers le dual est utilisée (Rontsis et Goulart, 2020).

Les techniques citées ci-dessus, qui ne se basent pas sur l'échantillonnage, sont proposées pour réduire la complexité du subspace clustering. Également, garantir la préservation des sous-espaces et traiter le problème de connectivité dans le graphe d'affinité. Certaines arrivent à atteindre ces garanties, mais sous certaines conditions théoriques. Toutefois, ces conditions ne reflètent pas les données du monde réel (avec des sous-espaces pas suffisamment séparés et avec des données pas forcément bien distribuées). De plus, les algorithmes utilisés souffrent toujours d'une complexité temporelle élevée, car ils sont résolus par des méthodes d'optimisation itératives.

Bien que les techniques présentées dans cette section proposent des solutions à la problématique de la scalabilité du subspace clustering, leurs applicabilités sur des données réelles sont très limitées. Elles ne considèrent que la relation linéaire entre les points de données. Cependant, en pratiques, les données ne sont pas forcément représentées à partir de sous-espaces linéaires. D'autres recherches se sont intéres-

sées à utiliser les réseaux de neurones profonds pour le subspace clustering. Ce qui permet de prendre en considération des données qui ne résident pas initialement dans des sous-espaces linéaires.

2.3 Techniques de scalabilité du subsapce clustering dans le contexte d'apprentissage profond :

Des approches ont été proposées en combinant l'apprentissage profond avec le subspace clustering (*Deep Subspace Clustering*). Les recherches les plus récentes utilisent les auto-encodeurs (AEs). Ces derniers ont prouvé leurs efficacités dans la réduction des dimensions. Plus précisément, l'extraction des caractéristiques les plus pertinentes afin de mieux représenter et discriminer les données originales. Les auto-encodeurs se composent principalement d'un encodeur et d'un décodeur. L'encodeur génère, dans un espace latent, une représentation compressée Z d'une donnée X. Différentes couches d'unité de traitement non linéaire sont utilisées pour l'extraction des dimensions caractérisant une donnée. Au niveau de chaque couche, en utilisant des filtres, une nouvelle représentation des données est obtenue. Chaque couche prend en entrée la sortie de la couche précédente. Ainsi, en empilant ces couches non linéaires ensemble, l'apprentissage profond donne de meilleures représentations des données. Le décodeur quant à lui, essaye à partir de la représentation Z d'avoir la meilleure reconstruction possible $\widehat{X}$ de la donnée d'entrée X, voir figure 2.3. Le but des auto-encodeurs est d'optimiser la qualité de la reconstruction en minimisant la fonction coût L_{ae} suivante :

$$L_{ae(\Theta)} = \sum_{j} \|X_j - g_{\Theta_d}(f_{\Theta_e}(X_j))\|_2^2 \tag{2.14}$$

où $f_{\Theta_e}(X_j) = Z_j$ est la fonction de transformation de l'encodeur, $g_{\Theta_d}(Z_j) = \hat{X}_j$ est la fonction de reconstruction du décodeur, Θ_e et Θ_d sont les paramètres de

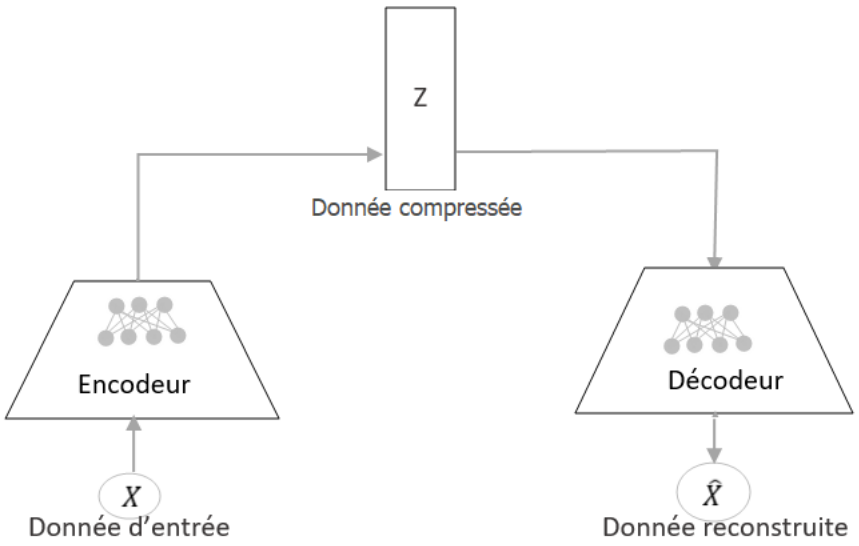


FIGURE 2.3 – Architecture d'un Auto-encodeur.

l'encodeur et du décodeur respectivement.

Dans le cadre du subspace clustering, **Deep Subspace Clustering Networks (DSCNets)** (Ji *et al.*, 2017) constitue la première tentative à avoir utilisé l'apprentissage profond non supervisé. Précisément, une couche complètement connectée, appelée autoreprésentation (Self-expression), est ajoutée entre l'encodeur et le décodeur, comme illustrer dans la figure 2.4. Cette couche vise à apprendre les similarités entre les données dans l'espace latent. La nouvelle fonction coût est définie comme suit :

$$L(\Theta) = \frac{1}{2}\left\|X - \hat{X}_\Theta\right\|_F^2 + \lambda_1 \left\|C_{\Theta s}\right\|_p + \frac{\lambda_2}{2}\left\|Z_{\Theta_e} - Z_{\Theta_e} C_{\Theta s}\right\|_F^2 \quad s.t.\ (diag(C) = 0) \tag{2.15}$$

où $\|Z_{\Theta_e} - Z_{\Theta_e} C_{\Theta_s}\|_F^2$ représente le terme de la couche d'autoreprésentation. Les poids de cette couche Θ_s correspondent à la matrice d'affinité $C \in \mathbb{R}^{n \times n}$. Les paramètres Θ correspondent aux paramètres Θ_e de l'encodeur, Θ_s de la couche autoreprésentation, et Θ_d du décodeur.

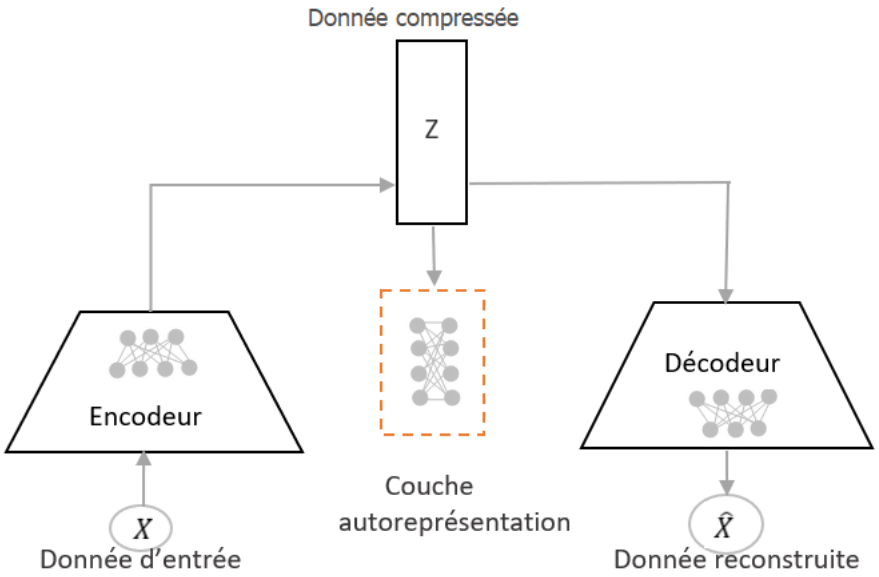


FIGURE 2.4 – Architecture du Deep Subspace Clustering Networks (DSCNets).

Les paramètres des couches encodeur et décodeur sont initialisés avec les paramètres obtenus dans une étape de préentraînement de l'auto-encodeur, sur toutes les données, sans la couche d'autoreprésentation. Par la suite, en introduisant la couche d'autoreprésentation, la fonction objective (2.15) est minimisée.

Les expérimentations ont démontré que DSCNets (Ji *et al.*, 2017) fournit une amélioration significative par rapport aux méthodes de subspace clustering en termes de précision du clustering sur plusieurs ensembles de données. Cependant, la complexité du calcul de la matrice d'affinité reste élevée ; d'ordre $O(n^2)$, n étant la taille de l'ensemble de données. Suite au modèle DSCNets, d'autres modèles se basant sur cette couche d'autoreprésentation ont été proposés. Quelques travaux sont présentés dans la Section 2.4. Toutefois, nous avons constaté qu'il n'y avait pas de travaux traitant la scalabilité du subspace clustering tout en utilisant la propriété d'autoreprésentation. Dans la section suivante, nous présenterons le seul travail, Deep k-Subspace Clustering, traitant la scalabilité du subspace clustering

dans le contexte d'apprentissage profond sans passer par la propriété d'autoreprésentation.

2.3.1 Scalable Deep k-Subspace Clustering (SDk-SC) :

(Zhang *et al.*, 2018) proposent, SDk-SC, une méthode scalable de subspace clustering profond mais sans l'utilisation de la matrice d'affinité. L'approche consiste à utiliser une variante de k-Subspace Clustering (k-SC) (Tseng, 2000; Agarwal et Mustafa, 2004). Cette dernière est une méthode itérative et peut être considérée comme une généralisation de l'algorithme k-means. Le k-SC cherche à minimiser la somme des résidus des points vers leurs sous-espaces les plus proches. En Considérant un ensemble de points $\{x_1, ..., x_n\} \in \mathbb{R}^d$ appartenant à une union de k sous-espaces $S_1, ..., S_k$ de dimensionnalité $p_1 = p_2 =, ..., = p_k$ respectivement, la fonction de minimisation pour le k-subspace clustering s'écrit comme suit :

$$\begin{aligned} \min_{\{S_i\},\{w_{ij}\}} \sum_{j}^{n} \sum_{i}^{k} w_{ij} \left\| X_j - S_i S_i^{\intercal} X_j \right\|_2^2 \\ s.t.\ w_{ij} \in \{o, 1\}, \quad \sum_{i=1}^{k} w_{ij} = 1 \end{aligned} \tag{2.16}$$

La valeur optimale pour w_{ij} est écrite comme suit :

$$w_{ij} = \begin{cases} 1 & i = \arg\min_m \left\| x_j - S_m S_m^{\intercal} x_j \right\|_2^2, \\ \\ 0 & \text{otherwise.} \end{cases} \tag{2.17}$$

avec $W \in \mathbb{R}^{k \times n}$ une matrice d'appartenance où $w_{ij} = 1$ signifie que le point x_j appartient au subspace S_i. En commençant par une initialisation de k sous-espaces candidats, le k-SC met à jour en alternance les w_{ij} et les sous-espaces S_i. L'équation (2.17) permet de classer les points dans leurs sous-espaces le plus

proches. Par la suite, les nouveaux sous-espaces sont réestimés en appliquant un SVD sur les points de chaque cluster (les colonnes de W où la ième ligne est égale à 1).

Le SDk-SC introduit le module k-subspace clustering entre l'encodeur et le décodeur, voir figure 2.5. L'objectif est d'appliquer la méthode k-SC sur les données Z de l'espace latent. La fonction coût de SDk-SC est définie comme suit :

$$L(\theta, \{S_i\}, W) = L_{ae}(\theta) + \lambda L_{ksc}(\{S_i\}, W) \tag{2.18}$$

Le terme $L_{ae}(\theta)$ représente la fonction coût de la reconstruction de l'auto-encodeur. Le terme $L_{ksc}(\{S_i\}, W)$ représente le coût du k-subspace clustering, il est défini comme suit :

$$\begin{aligned} L_{ksc}(\{S_i\}, W) &= \sum_{i,j} w_{ij} \left\| f_{\theta e}(x_j) - S_i S_i^{\intercal} f_{\theta e}(x_j) \right\|_2^2 \\ s.t. \quad S_i \in \mathcal{G}(d,p), w_{ij} &\in \{0,1\}, \sum_{i=1}^{k} w_{ij} = 1, \forall_{ij} \end{aligned} \tag{2.19}$$

où $\mathcal{G}(d,p)$ désigne la variété de Grassmann (Absil *et al.*, 2004) composée de p sous-espaces avec une dimensionnalité ambiante d .

S_i et W sont initialisés à partir des paramètres du préentraînement de l'auto-encodeur sans le module k-SC. Une fois le modèle SDk-SC entraîner avec le module k-SC, la représentation dans l'espace latent est recalculée, $Z = f_{\theta e}(X)$. Par la suite, l'appartenance aux sous-espaces pour chaque z_i est attribuée en utilisant l'équation (2.17).

Le modèle SDk-S propose de réduire la complexité du subspace clustering en contournant la construction de la matrice d'affinité. Cependant, la méthode nécessite que la taille des dimensions soit connue à priori et qu'elle soit la même pour

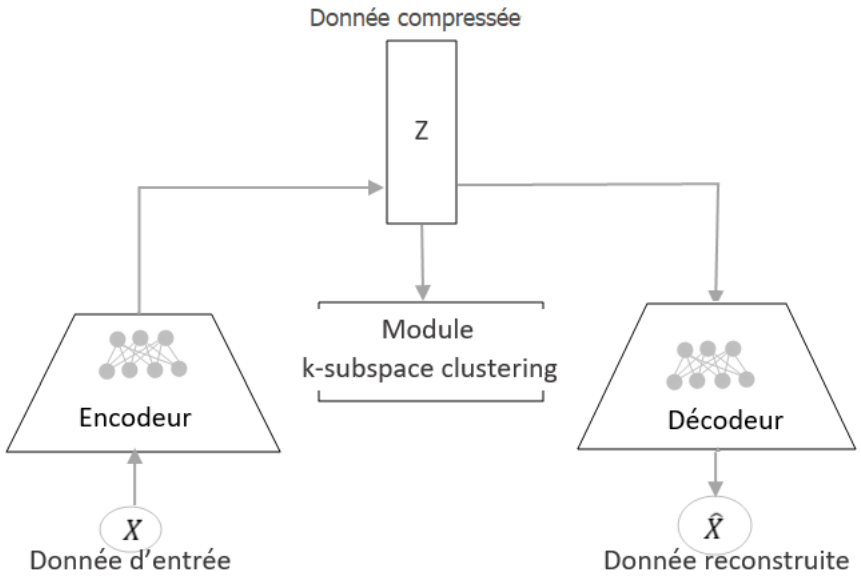


FIGURE 2.5 – Architecture du Scalable Deep k-Subspace Clustering (SDk-SC).

tous les sous-espaces, ce qui est une contrainte très restrictive. De plus, les méthodes k-SC souffrent toujours d'une complexité temporelle élevée, égale à $O(n^2)$, pour le calcul de la décomposition en valeurs singulières (SVD). Aussi, elles nécessitent une bonne initialisation et semblent fragiles aux données aberrantes (Zhang *et al.*, 2019b).

2.4 Subsapce clustering dans le contexte de l'apprentissage profond :

Nous présenterons les techniques récentes du subspace clustering utilisées dans le contexte d'apprentissage profond. Par ce fait, nous montrons comment le subspace clustering peut bénéficier du succès des réseaux de neurones profonds. De plus, nous montrons comment les performances du subspace clusetring peuvent être améliorées en combinant les avantages des méthodes existantes du subspace clustering avec celles de l'apprentissage profond.

(Zhang *et al.*, 2019a) proposent un modèle appelé **Self-Supervised Convolutional Subspace Clustering Network (S^{2}ConvSCN)**. Ce modèle propose de superviser la transformation des données dans l'espace latent, et l'apprentissage des similarités entre les points de données. Cette supervision est constituée principalement de deux modules : (1) module de couche complètement connectée d'autosupervision *(FC Layers)*, qui supervise la projection des données dans l'espace latent, (2) module de spectral clustering, qui fournit des informations pour l'étape d'autoreprésentation et pour le module *(FC Layers)*, comme montré dans la figure 2.6.

Le module spectral clustering utilise la matrice de coefficient C de la couche autoreprésentation pour regrouper les données avec la fonction coût suivante :

$$\sum_{i,j} |C_{ij}| \frac{\|q_i - q_j\|_2^2}{2} = \|C\|_Q \quad s.t. \quad QQ^\intercal = I \tag{2.20}$$

où $Q \in \{0,1\}^{k \times n}$ est la matrice de segmentation en k clusters, q_i et q_j sont les colonnes i et j de Q indiquant l'appartenance de chaque point de données aux clusters.

La fonction (2.20) mesure l'écart entre la matrice de coefficients C et la matrice de segmentation Q. Lorsque Q est fourni, la minimisation du coût de $\|C\|_Q$ implique que $c_{ij} \neq 0$ seulement si x_i et x_j appartiennent au même cluster. Par conséquent, l'ajout de cette fonction à celle du subspace clustering profond améliore l'apprentissage de la matrice d'affinité. Le résultat du module spectral clustering est aussi utilisé dans le module *(FC Layers)* pour superviser la transformation des données dans l'espace latent avec la fonction suivante :

$$\frac{1}{N} \sum_{j=1}^{N} (ln(1 + e^{-\tilde{\mathrm{y}}_j^\intercal q_j}) + \tau \left\| \mathrm{y}_j - \mu_{\pi(\mathrm{y}_j)} \right\|_2^2) \tag{2.21}$$

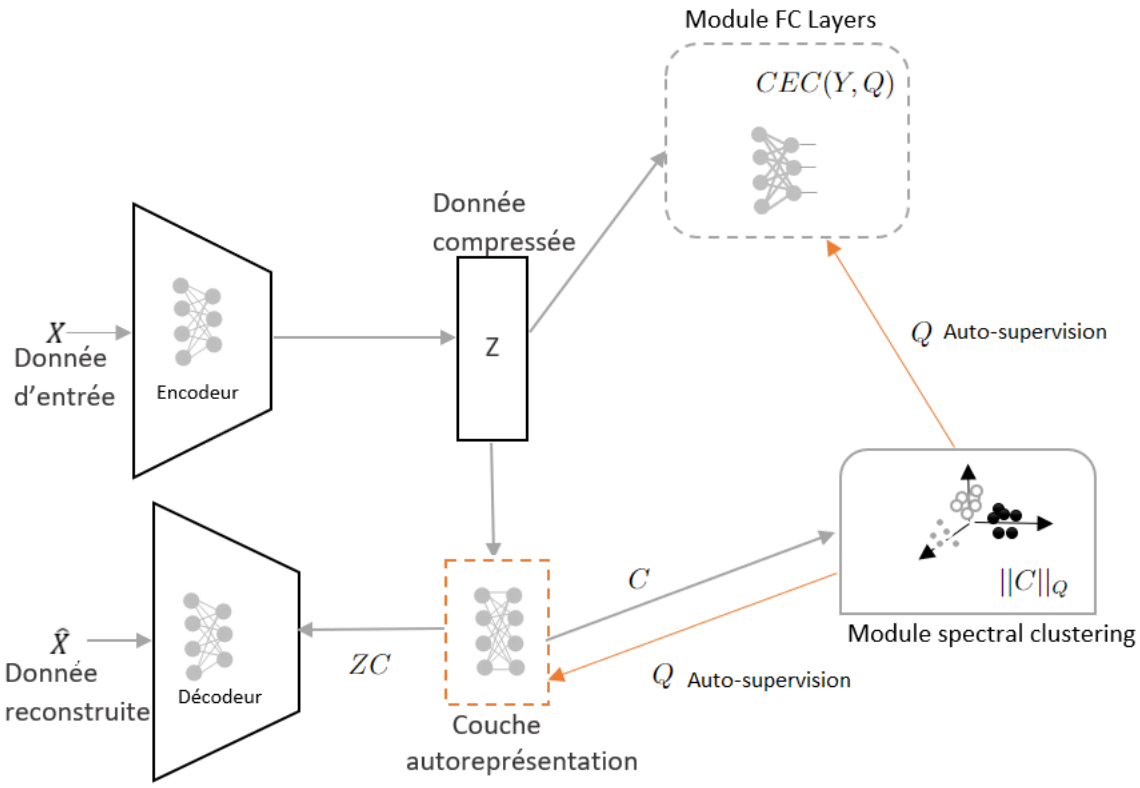


FIGURE 2.6 – Architecture du Self-Supervised Convolutional Subspace Clustering Networks (S^{2}ConvSCN).

où $\mathrm{y} \in \mathbb{R}^k$ est la sortie à k dimensions des couches du module *(FC Layers)*, $\mu_{\pi(\mathrm{y}_j)}$ représente le centre du cluster correspondant à y_j ; $\pi(\mathrm{y}_j)$ sert à prendre l'indice de y_j à partir du résultat du spectral clustering.

En combinant ces deux nouveaux modules à la fonction objective du subspace clustering profond, la nouvelle fonction objective est la suivante :

$$\begin{aligned} &\frac{1}{2}\left\|X-\hat{X}\right\|_F^2+\frac{1}{2}\left\|Z-ZC\right\|_F^2+\lambda\left\|C\right\|_l+\left\|C\right\|_Q \quad + \\ &\frac{1}{N}\sum_{j=1}^{N}(ln(1+e^{-\tilde{\mathrm{y}}_j^{\intercal}q_j})+\tau\left\|\mathrm{y}_j-\mu_{\pi(\mathrm{y}_j)}\right\|_2^2) \quad s.t.\ (diag(C)=0) \end{aligned} \tag{2.22}$$

Selon les expérimentations, la supervision fournie par le spectral clustering amé-

liore l'apprentissage de la nouvelle représentation des données ainsi que l'apprentissage des similarités entre les points de données, conduisant ainsi à de meilleures performances de clustering.

Deep Subspace Clustering (DSC) (Peng *et al.*, 2020) proposent de développer une extension de Sparse Subspace clustering (SSC) (Elhamifar et Vidal, 2013) dans un contexte d'apprentissage profond. Le modèle proposé, Deep Subspace Clustering (DSC), utilise un réseau de neurones pour l'extraction des dimensions et l'apprentissage simultané de la matrice d'affinité C. Le réseau de neurones est constitué de M couches non linéaires, où la sortie de chaque couche est représentée comme suit :

$$h^{(m)} = g(W^{(m)}h^{(m-1)} + b^{(m)}) \tag{2.23}$$

avec $m = 1, 2, ..., M$, $g(.)$ une fonction d'activation non linéaire, d^m la dimension de la sortie de la couche m, et $W^{(m)} \in \mathbb{R}^{d^{(m)} \times d^{(m-1)}}$ et $b^{(m)} \in \mathbb{R}^{d^{(m)}}$ représentent respectivement le poids et le biais relatifs à la couche m.

Ainsi, pour n instances de données, $H^{(M)}$ est défini comme une collection des sorties de la couche M : $H^{(M)} = [h_1^{(M)}, h_2^{(M)}, ..., h_n^{(M)}]$. En utilisant la nouvelle représentation des données $H^{(M)}$, et la norme l_1 pour que la matrice d'affinité soit éparse, la fonction coût suivante est minimisée :

$$\begin{aligned} \min_{\Theta, C} = \frac{1}{2}\left\|H^{(M)} - H^{(M)}C\right\|_F^2 + \gamma\left\|C\right\|_1 + \frac{\lambda}{4}\sum_{i=1}^{n}\left\|(h_i^{(M)})^\intercal h_i^{(m)} - 1\right\|_2^2 \\ s.t.\ diag(C) = 0 \end{aligned} \tag{2.24}$$

où $\Theta = \{W^{(m)}, b^{(m)}\}_{m=1}^{M}$ sont les paramètres du réseau de neurones, $\|.\|_1$ est la norme l_1, et le terme $\frac{\lambda}{4}\sum_{i=1}^{n}\left\|(h_i^{(M)})^\intercal h_i^{(m)} - 1\right\|_2^2$ est rajouté pour supprimer un facteur d'échelle arbitraire dans l'espace latent. Il a été noté que sans ce dernier terme, le réseau de neurones proposé peut tomber dans une solution triviale telle

que $H^{(M)} = 0$. Les auteurs ont mentionné qu'en utilisant un auto-encodeur ce terme peut être supprimé.

Pour résoudre la fonction (2.24) une minimisation alternée est utilisée.

(Kheirandishfard *et al.*, 2020) proposent un modèle appelé **Deep Low-Rank Subspace Clustering (DLRSC)**. L'objectif de ce modèle est d'apprendre une matrice d'affinité C à faible rang[6]. Une contrainte est ajoutée à la matrice C tel que $rank(C) = m$, avec $(m \ll n)$. Pour ce faire, la couche autoreprésentation pour la matrice $C \in \mathbb{R}^{n \times n}$ est remplacée par une couche avec une plus petite matrice $c^{n \times m}$ et sa transposée $c^{m \times n}$. La figure 2.7 décrit l'architecture du DLRSC. La matrice C est considérée comme une matrice symétrique de la forme $C = cc^{\intercal}$, où c est calculée par la nouvelle fonction objective suivante :

$$\left\| X - \hat{X} \right\| + \lambda_1 \left\| Z - Zcc^{\intercal} \right\|_F^2 + \lambda_2 \left\| cc^{\intercal} \right\|_2 \tag{2.25}$$

La matrice c est de taille $n \times m$ $(m \ll n)$ avec $\text{rang}(c) = \min(m, n)$. Par conséquent, $\text{rang}(C) = m$ est garantie. Ainsi, le modèle DLRSC permet d'apprendre une matrice d'affinité $C \in \mathbb{R}^{n \times n}$ à faible rang avec un coût de calcul inférieur au modèle standard (Liu *et al.*, 2012; Elhamifar et Vidal, 2013; Vidal et Favaro, 2014; Ji *et al.*, 2017).

Les travaux cités ci-dessus utilisent l'apprentissage profond pour apprendre une meilleure représentation des données pour le subspace clustering. Également, pour prendre en considération les données représentées par des sous-espaces non linéaires. Les principales différences entre les travaux du Subspace Clustering profond résident au niveau de la structure du réseau de neurones et la fonction coût

6. Le rang correspond au nombre maximal de vecteurs colonnes linéairement indépendants dans la matrice.

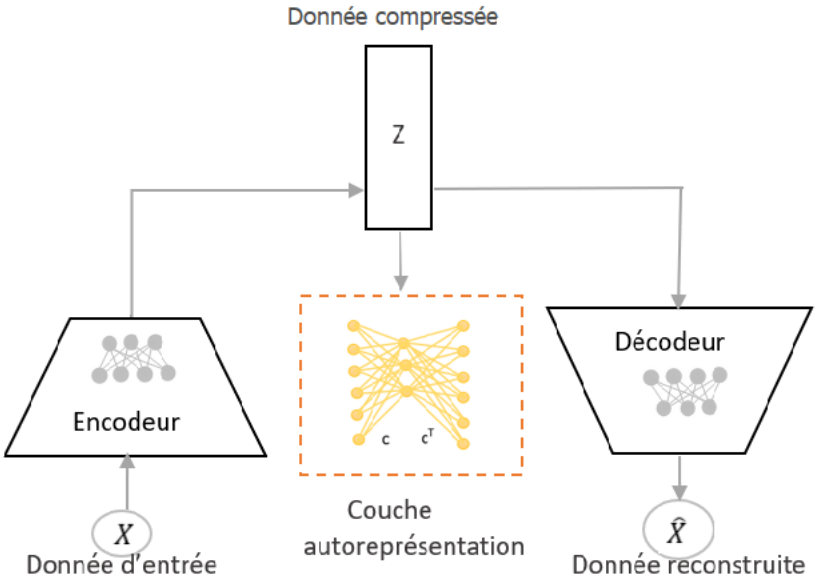


FIGURE 2.7 – Architecture du Deep Low-Rank Subspace Clustering (DLRSC).

utilisée selon un objectif bien défini.

Les réseaux de neurones ont attiré une attention considérable en raison de nombreuses applications réussies dans différents domaines. Néanmoins, l'utilisation des réseaux de neurones pour le subspace clustering a été critiquée dans (Haeffele *et al.*, 2020). Les critiques se basent sur une analyse théorique des modèles du subspace clustering profond utilisant les auto-encodeurs. Les auteurs critiquent l'efficacité du terme d'autoreprésentation incorporé entre l'encodeur et le décodeur. Ils formulent que ce terme, tel qu'il est utilisé, n'est pas suffisant pour apprendre une représentation latente exprimée par une union de sous-espaces linéaire. Spécifiquement, que sans aucune régularisation sur les paramètres réseau de l'auto-encodeur (Θ_e, Θ_d) ou normalisation des données Z dans l'espace latent, la formulation dans (2.15) est généralement mal posée. De telle sorte que la valeur du terme d'autoreprésentation peut être rendue arbitrairement petite sans vraiment changer la valeur de la fonction coût de l'auto-encodeur.

2.5 Conclusion :

Dans ce chapitre, nous avons présenté différents travaux traitant la problématique de la scalabilité du subspace clustering linéaire. Également, nous avons présenté quelques travaux du subspace clustering dans le contexte d'apprentissage profond. Nous avons constaté qu'il n'y avait pas de travaux traitant la scalabilité du subspace clustering, en utilisant la propriété d'autoreprésentation, dans le contexte d'apprentissage profond. De ce fait, dans le chapitre suivant, nous présenterons une nouvelle méthode de subspace clustering scalable basée sur l'apprentissage profond.

CHAPITRE III

APPROCHE PROPOSÉE

Ce chapitre présente la première proposition de solution à la problématique de la scalabilité du subspace clustering dans le contexte d'apprentissage profond. Notre modèle se base sur la propriété d'autoreprésentation du subspace clustering. Cette propriété permet d'apprendre les similarités entre les points de données et s'est avérée efficace dans le clustering des sous-espaces. Le modèle proposé se base également sur l'apprentissage profond en utilisant l'auto-encodeur. Ce dernier permet de réduire la dimensionnalité des données et de prendre en considération des données qui résident dans des sous-espaces non linéaires. L'approche proposée consiste à considérer un sous-ensemble de points de repère plus petit que l'ensemble de données initiales pour identifier les sous-espaces. Une nouvelle fonction objective est définie pour la propriété d'autoreprésentation. Cette dernière permet d'apprendre une matrice plus petite pour la construction de la matrice d'affinité. Ce qui permet de réduire la complexité relative à la construction de la matrice d'affinité ainsi que celle du clustering spectral de $O(n^3)$ vers une complexité linéaire $O(n)$. Dans ce qui suit, nous donnons premièrement quelques notations et définitions. Ensuite, nous présentons notre nouvelle fonction objective. Par la suite, nous présentons la méthode utilisée pour le regroupement des données. Enfin, nous présentons la stratégie d'entraînement utilisée par notre modèle que nous avons nommé Subspace Clustering Scalable Profond (SCS-P) .

3.1 Notations :

Nous présenterons dans cette section quelques notations liées au sujet traité. $\mathbf{X} = \{x_1, x_2, ..., x_n\} \in \mathbb{R}^{n \times D}$ désigne la matrice des données, où n est la taille des données, et D la taille des dimensions. La matrice des données $\mathbf{X}$ est formée à partir de plusieurs sous-espaces vectoriels de faible dimension $\{S_i\}_{i=1,...,k}$, $S_i \subset \mathbb{R}^{d_i^{'}}$ avec $\sum_{i=1}^{k} d_i^{'} \leq D$. L'objectif de ce travail est de regrouper les données $\mathbf{X}$ en k clusters, où chaque cluster appartient à un ou plusieurs sous-espaces vectoriels. Notre approche se base sur l'apprentissage profond, en utilisant l'auto-encodeur. Permettant ainsi de réduire la dimensionnalité des données et de prendre en considération des données qui résident dans des sous-espaces non linéaires de faible dimension. Pour l'identification des sous-espaces, nous utilisons la propriété de l'autoreprésentation des données. Cette dernière consiste principalement à construire une matrice d'affinité entre tous les points de données $C = \{C_{ij}\}_{i,j=1,....n}$, avec $C_{ij} = 0$ lorsque les données x_i et x_j appartiennent à deux sous-espaces différents. Dans notre solution, nous proposons une nouvelle approche scalable du subspace clustering. Notre objectif est de reformuler le problème du subspace clustering, qui a une complexité cubique, pour avoir un autre problème qui a une complexité linéaire par rapport à la taille des données.

3.2 Modèle proposé - Subspace Clustering Scalable Profond :

Le modèle que nous proposons utilise un auto-encodeur (pour l'extraction des attributs (dimensions)) et une couche d'autoreprésentation entre l'encodeur et le décodeur (pour apprendre les similarités entre les points de données).

3.2.1 Extraction des attributs :

Nous utilisons un auto-encodeur convolutif. L'encodeur est composé de plusieurs couches convolutives $\{h^{(\ell)}\}_{\ell=1}^{L}$. Chaque couche possède un ensemble de poids $\mathbf{w}^{(\ell)}$ et les biais $\mathbf{b}^{(\ell)}$ associés, ainsi l'entrée $\mathbf{X}$ est transformée à travers les différentes couches comme suit :

$$\begin{aligned} h^{(0)} &= f_{\mathrm{ReLU}}(\mathbf{X}, \mathbf{w}^{(0)}) \\ h^{(1)} &= f_{\mathrm{ReLU}}(h^{(0)}, \mathbf{w}^{(1)}) \\ &\vdots \\ h^{(L)} &= f_{\mathrm{ReLU}}(h^{(L-1)}, \mathbf{w}^{(L)}) \end{aligned} \tag{3.1}$$

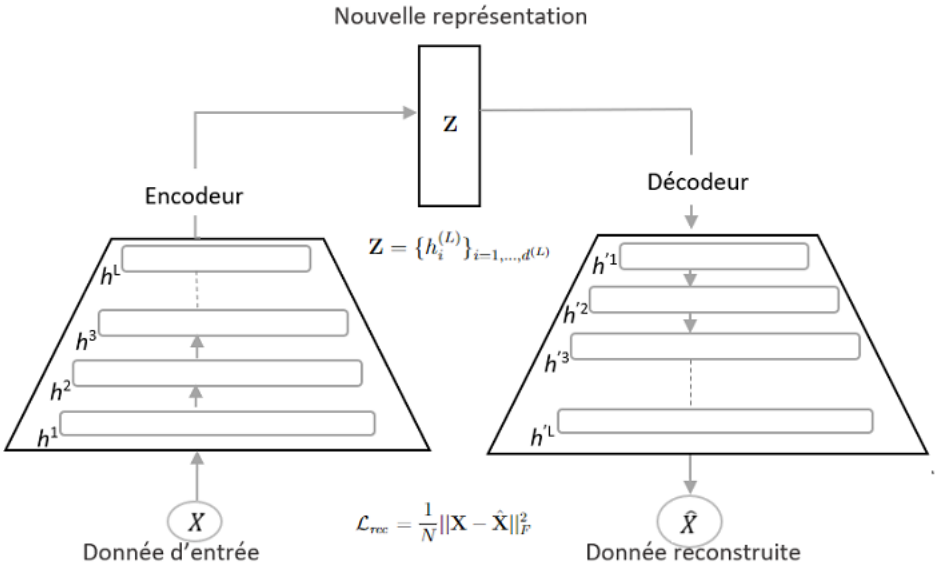


FIGURE 3.1 – Architecture de l'auto-encodeur utilisé.

avec $f_{\mathrm{ReLU}}(h^{(\ell)}, \mathbf{w}^{(\ell)}) = \mathrm{ReLU}(h^{(\ell-1)}\mathbf{w}^{(\ell)} + \mathbf{b}^{(\ell)})$ est la fonction de transformation de la couche ℓ, et $h^{(\ell)}$ représente la sortie de la couche convolutive ℓ. Au niveau de chaque couche, $d^{(\ell)}$ attributs $\{h_i^{(\ell)}\}_{i=1}^{d^{(\ell)}}$ sont produits. Les attributs $d^{(L)}$ pro-

duits par la dernière couche L sont utilisés pour former la nouvelle représentation compressée $\mathbf{Z} \in \mathbb{R}^{n \times d}$, dans l'espace latent, des données initiales :

$$\mathbf{Z} = \{h_i^{(L)}\}_{i=1,...,d^{(L)}} = [h_1^{(L)}, h_2^{(L)}, ..., h_{d^{(L)}}^{(L)}] \tag{3.2}$$

avec $\{h_i^{(L)}\}_{i=1,...,d^{(L)}}$ représentent les vecteurs d'attributs de la couche L.

À l'inverse de l'encodeur, le décodeur décompresse la nouvelle représentation $\mathbf{Z}$ (voir figure 3.1) . Il essaye de produire la meilleure reconstruction $\hat{\mathbf{X}}$ de la données d'entrée $\mathbf{X}$:

$$\begin{aligned} \hat{\mathbf{X}} &= Dec(\mathbf{Z}, \hat{\mathbf{w}}) \\ &= Dec(Enc(\mathbf{X}, \mathbf{w}), \hat{\mathbf{w}}) \end{aligned} \tag{3.3}$$

où Enc est la fonction de transformation des données $\mathbf{X}$, Dec est la fonction de reconstruction des données transformées, et $\hat{\mathbf{w}}$ représente les poids du décodeur.

L'auto-encodeur est ainsi entraîné, à travers plusieurs itérations, pour minimiser l'erreur de reconstruction suivante :

$$\begin{aligned} \mathcal{L}_{rec} &= \frac{1}{N} \sum_{j=1}^{N} \|\mathbf{x_j} - \hat{\mathbf{x}_j}\|_F^2 \\ &= \frac{1}{N} \left\|\mathbf{X} - \hat{\mathbf{X}}\right\|_F^2 \\ &= \frac{1}{N} \|\mathbf{X} - Dec(Enc(\mathbf{X}))\|_F^2 \end{aligned} \tag{3.4}$$

avec $\|.\|_F$ représente la norme Frobenius (Lu *et al.*, 2012).

3.2.2 Étape d'autoreprésentation :

Nous utilisons une couche d'autoreprésentation complètement connectée pour apprendre les similarités entre tous les points de données. La couche autoreprésen-

tation standard apprend une matrice d'affinité $C \in \mathbb{R}^{n \times n}$ en minimisant le coût de la fonction suivante pour le subspace clustering profond :

$$\mathcal{L}_{scp} = \mathcal{L}_{rec} + \mathcal{L}_{ar} \tag{3.5}$$

avec $\mathcal{L}_{rec}$ représente le coût de la reconstruction de l'auto-encodeur, et $\mathcal{L}_{ar}$ représente le coût de l'autoreprésentation défini comme suit :

$$\mathcal{L}_{ar} = \lambda \left\| Z - ZC \right\|_F^2 + \left\| C \right\|_1 \tag{3.6}$$

L'objectif du problème (3.5) est de capturer la non-linéarité et réduire la dimensionnalité des données. Également, apprendre une nouvelle représentation des données qui satisfait la propriété d'autoreprésentation (c'est-à-dire, $Z \approx ZC$), avec une complexité $O(n^3)$. Dans ce qui suit, nous allons présenter notre approche capable d'effectuer le subspace clustering profond en un temps linéaire par rapport à la taille des données.

Le problème du subspace clustering peut être formulé comme suit (Elhamifar et Vidal, 2013) :

$$\min_C \left\| C \right\|_1 \quad s.t. \quad X = CX \tag{3.7}$$

avec $X \in \mathbb{R}^{n \times D}$ est la matrice de données initiales, $C \in \mathbb{R}^{n \times n}$ est la matrice d'affinité, et $\left\| . \right\|_1$ représente la norme l_1 utilisée pour avoir une matrice éparse. La solution au problème (3.7) est formulée comme suit :

$$C^* = (XX^\intercal + \lambda I)^{-1} XX^\intercal \tag{3.8}$$

La matrice C^* à deux propriétés, elle est semi-définies positives (PSD) et symétrique. Dans ce qui suit nous allons démontrer que $C^* = (XX^\intercal + \lambda I)^{-1} XX^\intercal$ est

semi-définies positives :

$$\begin{cases} XX^\intercal & \text{est PSD} \\ \lambda I & \text{est PSD} \end{cases} \implies (XX^\intercal + \lambda I) \quad \text{est PSD } (\textit{somme de deux matrices PSD})$$

$$(\mathrm{XX}^\intercal + \lambda I) \quad \text{est PSD} \quad \implies \quad (XX^\intercal + \lambda I)^{-1} \quad \text{est PSD}$$

Le produit de deux matrices PSD est PSD, si et seulement si le produit est symétrique. Dans ce qui suit nous allons démontrer que $(XX^\intercal + \lambda I)^{-1}XX^\intercal$ est symétrique :

$$\mathrm{C}^* = (XX^\intercal + \lambda I)^{-1}XX^\intercal$$
$$C^* = (XX^\intercal + \lambda I)^{-1}(XX^\intercal + \lambda I - \lambda I)$$
$$C^* = [(XX^\intercal + \lambda I)^{-1}(XX^\intercal + \lambda I)] - (XX^\intercal + \lambda I)^{-1}\lambda I$$
$$\boxed{C^* = I - \lambda(XX^\intercal + \lambda I)^{-1}}$$

$$(\mathrm{C}^*)^\intercal = XX^\intercal(XX^\intercal + \lambda I)^{-1}$$
$$(C^*)^\intercal = (XX^\intercal + \lambda I - \lambda I)(XX^\intercal + \lambda I)^{-1}$$
$$(C^*)^\intercal = [(XX^\intercal + \lambda I)^{-1}(XX^\intercal + \lambda I)] - (XX^\intercal + \lambda I)^{-1}\lambda I$$
$$\boxed{(C^*)^\intercal = I - \lambda(XX^\intercal + \lambda I)^{-1}}$$

$$\boxed{C^* = (C^*)^\intercal \quad \implies \quad C^* \quad \text{est symétrique}}$$

Avec $[XX^\intercal] \in \mathbb{R}^{n\times n}$, la complexité pour calculer C^* est $O(n^3)$. Pour réduire la

complexité du subspace clustering nous construisons un ensemble de m points de repère, avec m plus petit que la cardinalité de l'ensemble de données n. En effet, un petit nombre d'instances sont suffisantes pour identifier les sous-espaces. Théoriquement, pour chaque sous-espace, il suffit de fournir un nombre d'échantillons indépendants (formant une famille indépendante) égale à la dimension de ce sous-espace.

Notre but initial est de décomposer la matrice d'affinité de taille $n \times n$ sous la forme d'un produit de deux matrices de taille $n \times m$ et $m \times n$ respectivement. Pour arriver à cette fin, nous nous basons sur la décomposition de cholesky suivante :

une matrice $M \in \mathbb{R}^{n \times n}$ est semi-definie positive (PSD) si est seulement si M peut être décomposée et écrite sous la forme $BB^\intercal$. Avec $B \in^{n \times k}$, $rank(B) = rank(M) = k$.

C^* étant PSD, elle peut être décomposée en deux matrices et s'écrire comme suit :

$$C^* = PP^\intercal \quad s.t. \quad P^\intercal P = I_m \tag{3.9}$$

En remplaçant C avec (3.9) dans la fonction (3.6), cette dernière devient :

$$\mathcal{L}_{ar} = \|Z - PP^\intercal Z\|_F^2 + \|PP^\intercal\|_1 \quad s.t. \quad P^\intercal P = I_m \tag{3.10}$$

Nous pouvons voir que le problème dans (3.10) satisfait la propriété d'autoreprésentation suivante :

$$Z \approx PP^\intercal Z \tag{3.11}$$

La technique de factorisation matricielle tente de compresser les données en trouvant une représentation, par rapport à un ensemble de vecteurs de base, pour chaque point de données. La factorisation matricielle de $Z \in \mathbb{R}^{n \times d}$ peut être

mathématiquement définie par l'identification de deux matrices $A \in \mathbb{R}^{n \times m}$ et $L \in \mathbb{R}^{m \times d}$ dont le produit peut être plus approximatif à Z :

$$Z \approx AL \tag{3.12}$$

A partir de (3.12) et (3.11), nous pouvons voir que le problème dans (3.10) satisfait la propriété d'autoreprésentation suivante :

$$\begin{aligned} Z \approx & PL, \quad \text{avec} \\ & A = P, \\ & L = P^\intercal Z \end{aligned} \tag{3.13}$$

Ce qui implique que le problème d'optimisation dans (3.10) peut s'écrire comme suit :

$$\mathcal{L}_{ar} = \|Z - PL\|_F^2 + \|PP^\intercal\|_1 \quad s.t. \quad P^\intercal P = I_m \tag{3.14}$$

La contrainte $P^\intercal P = I_m$ implique que $\|PP^\intercal\|_1 = m$. Cette dernière est une constante qui peut être supprimée de la fonction objective (3.14). Notre fonction coût de l'autoreprésentation pour une solution scalable est formulée comme suit :

$$\mathcal{L}_{ars} = \|Z - PL\|_F^2 \quad s.t. \quad P^\intercal P = I_m \tag{3.15}$$

En introduisant le terme d'autoreprésentation (3.15), la fonction coût pour le subspace clustering scalable profond est comme suit :

$$\mathcal{L}_{scps} = \mathcal{L}_{ars} + \mathcal{L}_{rec} \tag{3.16}$$

Nous proposons de résoudre le problème du subspace clustering scalable profond par la stratégie d'optimisation alternée. En minimisant la fonction coût (3.17), P

et L sont résolus alternativement, en fixant l'un d'eux puis mettre à jour l'autre.

$$\mathcal{L}_{scps} = \min_{P,L} \left(\frac{1}{N} \left\| \mathbf{X} - Dec(Enc(\mathbf{X})) \right\|_F^2 + \lambda \left\| Enc(\mathbf{X}) - PL \right\|_F^2 \right), s.t. \quad P^\intercal P = I_m \tag{3.17}$$

Calcul de P :

En fixant L, le calcul de P revient à minimiser la fonction coût suivante :

$$\min_P \left\| X - \hat{X} \right\|_F^2 + \lambda \left\| Z - PL \right\|_F^2 \quad s.t. \quad P^\intercal P = I_m \tag{3.18}$$

Le deuxième terme dans (3.18) représente un problème appelé *Orthogonal Procrustes*. Ce dernier est un problème d'approximation de matrice dans l'algèbre linéaire. Étant donné deux matrices X, L, la question est de trouver une matrice semi-orthogonale Q tel que :

$$\min_Q \left\| X - QL \right\|_F^2 \quad s.t. \quad Q^\intercal Q = I \tag{3.19}$$

Dans notre cas, on peut considérer $P = Q$. Néanmoins, puisque P n'est pas une matrice carrée, le problème dans (3.18) n'est pas un problème Procruste classique. Cependant, selon (Gan *et al.*, 2020) le problème (3.18) peut être équivalent au problème Procruste standard en complétant L par des zéros :

$$\bar{L} = \begin{bmatrix} L \\ 0 \end{bmatrix}$$

où le bloc supplémentaire de 0 rend $\bar{L}$ de taille $n \times d$. Le problème dans (3.19) est équivalent à :

$$\min_Q \left\| X - Q\bar{L} \right\| \quad s.t. \quad Q^\intercal Q = I \tag{3.20}$$

avec $Q \in \mathbb{R}^{n \times n}$ une matrice semi-orthogonale, la solution à ce problème est premièrement de trouvez la décomposition en valeurs singulières (SVD) de la matrice $XL^\intercal$:

$$XL^\intercal = U\Sigma V^\intercal \tag{3.21}$$

Par la suite, la solution optimale pour Q est :

$$Q = UV^\intercal \tag{3.22}$$

Nous pouvons voir que $PL = Q\bar{L}$, car les $(n-m)$ dernières lignes de $\bar{L}$ sont à 0. En remplaçant X par Z dans (3.20) nous pouvons résoudre P en utilisant *Orthogonal Procrustes* comme suit :

$$\min_P \|Z - PL\|_F^2 \quad s.t. \quad P^\intercal P = I \tag{3.23}$$

La solution au problème (3.23) est premièrement de trouvez la décomposition en valeurs singulières (SVD) de la matrice $ZL^\intercal \in \mathbb{R}^{n \times m}$. Avec $m \ll n$, nous utilisons la version économique de SVD qui utilise beaucoup moins d'espace et de temps de calcul. La décomposition économique en valeurs singulières est appliquée, avec une complexité linéaire, comme suit :

$$ZL^\intercal = U\Sigma V^\intercal \tag{3.24}$$

où $U \in \mathbb{R}^{n \times m}$, $\Sigma \in \mathbb{R}^{m \times m}$, et $V \in \mathbb{R}^{m \times m}$. Par la suite, nous calculons $P \in \mathbb{R}^{n \times m}$, avec $m \ll n$, avec une complexité linéaire $O(nm^2)$:

$$P = UV^\intercal \tag{3.25}$$

Calcul de L :

La matrice $L \in \mathbb{R}^{m \times d}$ a la même dimensionnalité que les points de données dans l'espace latent. Nous pouvons considérer les vecteurs colonnes de cette matrice comme des points de repère de l'ensemble de données. L'objectif est de pouvoir identifier les sous-espaces vectoriels à partir de ce sous-ensemble de données représenté par la matrice L. Plusieurs méthodes ont été proposées pour la sélection de ces points de repère. Cependant, beaucoup de ces méthodes sont coûteuses en termes de calcul et ne s'adaptent pas aux grands ensembles de données. De plus, certaines se basent sur des hypothèses trop restrictives et ne peuvent pas être garanties dans les problèmes des données du monde réel (avec des sous-espaces pas suffisamment séparés et avec des données pas forcement bien distribuées).

En fixant P, le calcul de L revient à minimiser la fonction coût suivante :

$$\min_{L} \left\| X - \hat{X} \right\|_F^2 + \lambda \left\| Z - PL \right\|_F^2 \quad s.t. \quad P^\intercal P = I_m \tag{3.26}$$

Pour notre solution, puisque le problème pour résoudre L est convexe, nous pouvons utiliser un calcul exact, en fixant P, L peut être calculé comme suit :

$$L = P^\intercal Z \tag{3.27}$$

Le calcul de la solution exacte pour L a une complexité $O(n^2)$. Pour cette raison, nous proposons d'apprendre l'ensemble de données représentées par la matrice L avec une complexité linéaire. En effet, comme le problème dans (3.26) est convexe, nous le résolvant en utilisant la descente de gradient :

$$\min_{L} \left(\frac{1}{N} \left\| \mathbf{X} - Dec(Enc(\mathbf{X})) \right\|_F^2 + \lambda \left\| Enc(\mathbf{X}) - PL \right\| \right) \tag{3.28}$$

En appliquant la descente de gradient, notre approche converge vers le minimum global. Ainsi, nous pouvons éviter le calcul exact pour L.

3.2.3 Étape du clustering

Dans cette étape, il est question de regrouper les données originales en k clusters. La méthode communément utilisée est le spectral clustering. À partir de la matrice P obtenue dans l'étape précédente, et en se basant sur l'équation (3.9), nous pouvons calculer une matrice d'affinité $C = PP^\intercal \in \mathbb{R}^{n \times n}$ entre tous les points de données. Ainsi, le spectral clustering peut être appliqué sur la matrice C pour regrouper les données. Le spectral clustering se base sur la décomposition en valeur propre du graphe Laplacien comme suit :

1. À partir de la matrice C, calculer le Laplacien.
2. Calculer les k vecteurs propres $u_1, u_2, .., u_k$ de Laplacien, k étant le nombre de clusters.
3. Soit $U \in \mathbb{R}^{n \times k}$ la matrice des vecteurs propres, appliquer k-means sur U pour avoir les clusters.

La complexité du spectral clustering appliqué sur la matrice $C \in \mathbb{R}^{n \times n}$ est élevée d'ordre $O(n^3)$. C'est pourquoi, en utilisant uniquement la matrice $P \in \mathbb{R}^{n \times m}$, nous proposons une méthode scalable pour regrouper les données.

La décomposition en valeurs singulières (SVD) peut être interprétée comme une décomposition en valeurs propre. Pour trouver la matrice des vecteurs propres U de la matrice C, il suffit d'appliquer la décomposition en valeurs singulières (SVD) tel que :

$$SVD(C) = U\Sigma V^\intercal \tag{3.29}$$

où $U \in \mathbb{R}^{n \times n}$, $\Sigma \in \mathbb{R}^{n \times n}$, et $V \in \mathbb{R}^{n \times n}$. En se basant sur l'équation (3.9), nous avons :

$$\begin{aligned} C &= PP^\intercal \\ &= (U\Sigma V^\intercal)(U\Sigma V^\intercal)^\intercal \\ &= U\Sigma(V^\intercal V)\Sigma U^\intercal \\ &= U\Sigma^2 U^\intercal \end{aligned} \tag{3.30}$$

Puisque C est une matrice carrée et semi-définie positive, elle peut s'écrire comme suit :

$$C = U\Sigma U^\intercal \tag{3.31}$$

Σ est une matrice diagonale avec des éléments σ_i le long de la diagonale. Lorsque C est semi-défini positive, σ_i sont des nombres réels non négatifs de sorte que la décomposition $C = U\Sigma U^\intercal$ est aussi une décomposition en valeurs singulières.

Ainsi, les vecteurs singuliers gauches de P sont les mêmes que les vecteurs propres de C. De ce fait, nous réduisons la complexité du spectral clustering en appliquant la décomposition en valeurs singulières sur la matrice P, plus petite, de taille $n \times m$ avec $m \ll n$. Nous utilisons la version économique du SVD, qui utilise beaucoup moins d'espace et de temps de calcul, avec une complexité linéaire :

$$SVD(P) = U\Sigma V^\intercal \tag{3.32}$$

où $U \in \mathbb{R}^{n \times m}$, $\Sigma \in \mathbb{R}^{m \times m}$, et $V \in \mathbb{R}^{m \times m}$. En ne gardant que les k plus grandes valeurs singulières, nous obtenons une matrice $U \in \mathbb{R}^{n \times k}$ des k vecteurs propres. Par la suite nous appliquons k-means sur U pour obtenir les k clusters.

3.2.4 Entraînement du modèle Subspace Clustering Scalable Profond :

Notre modèle **SCS-P** (Subspace Clustering Scalable Profond) se compose de trois parties, un encodeur, une couche d'autoreprésentation et un décodeur. L'architecture globale du modèle proposé est illustrée par la figure 3.2.

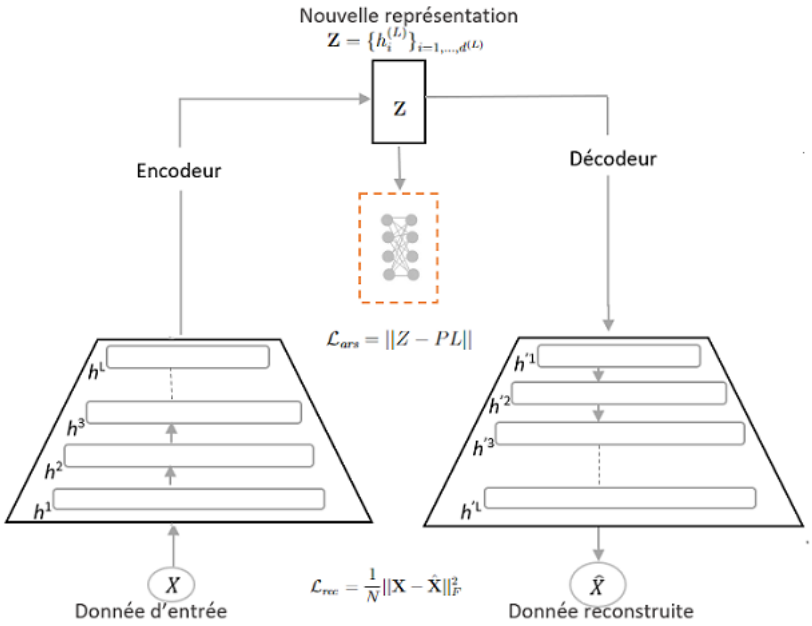


FIGURE 3.2 – Architecture du modèle subspace clustering scalable profond.

Nous présentons la procédure pour l'entraînement de notre modèle **SCS-P** dans l'Algorithme 1. Avant de commencer l'entraînement de notre modèle, nous effectuons un préentraînement de l'auto-encodeur profond sans la couche d'autoreprésentation sur toutes les données. À cette fin, le préentraînement est fait sur toutes les données en utilisant l'architecture illustrée dans la figure 3.1. Ainsi, à travers plusieurs itérations, les données sont transformées et compressées par l'encodeur. Les données dans l'espace latent produites par l'encodeur sont reconstruites par le décodeur en minimisant la fonction coût (3.4). Nous utilisons ensuite les paramètres de l'auto-encodeur préentraîné pour initialiser les couches d'encodeur

et de décodeur de notre réseau présenté par la figure 3.2. Pour la matrice P et la matrice des points de repère L, nous adoptons la même initialisation, égale à 1.0e-4, utilisée par les méthodes de subspace clustering profond (Peng *et al.*, 2020; Ji *et al.*, 2017; Zhang *et al.*, 2019a; Kheirandishfard *et al.*, 2020).

Algorithme 1 : Modèle Subspace Clustering Scalable Profond

Entrée : La matrice de données $\mathcal{X} \in \mathbb{R}^{n \times D}$, le nombre de clusters k, le nombre de points de repère $m \ll n$, l'encodeur Enc, le nombre d'itérations T, $t < T$;

Résultat : k clusters ;

début

- Préentraîner l'auto-encodeur via l'équation (3.4) ;
- Initialisation des paramètres réseaux de l'auto-encodeur du modèle par les paramètres obtenus par le préentraînement ;
- Initialisation de la matrice P et de la matrice des points de repère L ;

pour $i = 0$ **à** T **faire**

En fixant P, mettre à jour $L \in \mathbb{R}^{m \times d}$ par la descente de gradient via l'équation (3.28) ;

si $i \,\%\, t = 0$ **alors**

En fixant L, calculer SVD($ZL^\intercal$), avec $Z = Enc(\mathcal{X})$;

Mettre à jour $P \in \mathbb{R}^{n \times m}$ par l'équation (3.25) ;

fin

fin

- Calculer SVD(P) pour avoir les vecteurs singuliers gauches $U \in \mathbb{R}^{n \times m}$;
- Obtenir les k vecteurs propres $U \in \mathbb{R}^{n \times k}$, en ne gardant que les k plus grandes valeurs singulières ;
- Appliquer k-means sur U pour avoir les k clusters.

fin

Par la suite, nous utilisons l'architecture présentée dans la figure 3.2 pour la phase d'entraînement de notre modèle. Nous entraînons toutes les données avec une méthode de descente de gradient. En fixant P, à travers plusieurs itérations, la matrice des points de repère L est mise à jour en minimisant la fonction coût (3.28). Arrivée à $t < T$ itérations, nous fixons L et nous mettons à jour P par le calcul exact de l'équation (3.25). Comme une dernière étape, à la fin de l'entraînement, nous appliquons k-means sur la matrice des vecteurs propres obtenue en appliquant le SVD sur la matrice P.

3.3 Analyse de la complexité :

La complexité de SCS-P dépend de la complexité du préentraînement de l'auto-encodeur et les deux étapes du subspace clustering, à savoir, la construction de la matrice d'affinité et le clustering spectral.

Pour l'étape de construction de la matrice d'affinité, en utilisant notre fonction objective (3.17), la complexité : (1) du calcul de la matrice P est égale à $O(nm^2)$, n est la taille de données et $m \ll n$ est le nombre de points de repère, (2) du calcul de la matrice L est égale à $O(nmd)$, $d \ll n$ est la dimension des données dans l'espace latent, (3) de la transformation des données est égale à $O(lnd^2)$, l est le nombre de couches de l'encodeur, (4) de la minimisation de l'erreur de reconstruction des données est égale à $O(dn)+O(2lnd^2) = O(n(d+2ld^2))$. En rajoutant la complexité du préentraînement qui est égale à $O(n(d+2ld^2))$, la complexité totale pour cette étape est égale à $O(nm^2+nmd+lnd^2+2n(d+2ld^2) = O(n(m^2+md+ld+d+2ld^2))$. Avec $n \gg m, d$ et l, la complexité pour le calcul de la matrice d'affinité est linéaire par rapport à la taille des données.

Dans la deuxième étape, la décomposition en valeurs singulières de la matrice P entraîne une complexité $O(nm^2)$. De plus, pour le regroupement final, l'algorithme k-means est appliqué avec une complexité $O(nk^2t)$, ou k est le nombre de clusters, et t nombre d'itérations de k-means. La complexité totale pour cette étape est égale à $O(nm^2 + nk^2t) = O(n(m^2 + k^2t))$, avec $n \gg m, k$ et t, la complexité du spectral clustering est également réduite vers une complexité linéaire.

3.4 Conclusion :

Dans ce chapitre, nous avons présenté notre modèle SCS-P (Subspace Clustering Scalable Profond). Ce dernier est la première proposition à la problématique

de la scalabilité du subspace clustering dans le contexte d'apprentissage profond. Notre modèle se base sur la propriété d'autoreprésentation du subspace clustering. Également, sur l'apprentissage profond en utilisant l'auto-encodeur. Comme une première étape, en utilisant les données dans l'espace latent, une nouvelle fonction objective est définie pour la propriété d'autoreprésentation. Cette dernière permet d'apprendre une matrice plus petite pour la construction de la matrice d'affinité. Dans la deuxième étape, cette matrice est utilisée pour le regroupement des données. L'apprentissage d'une matrice plus petite à permis de réduire la complexité relative à la construction de la matrice d'affinité ainsi que celle du clustering spectral de $O(n^3)$ vers une complexité linéaire $O(n)$. Dans le prochain chapitre, nous présenterons les résultats obtenus des expérimentations de notre modèle sur des données réelles et synthétiques.

CHAPITRE IV

ÉVALUATION DE L'APPROCHE PROPOSÉE

Ce chapitre présente une évaluation empirique de notre approche **SCS-P** (Subspace Clustering Scalable Profond) sur des données synthétiques et réelles. La performance de notre approche est comparée à sept méthodes de subspace clustering scalable. À noter que, dans la littérature courante, il n'y a aucune méthode de subspace clustering scalable dans un contexte d'apprentissage profond à laquelle notre approche peut être comparée. Pour cette raison, la performance de notre approche en termes d'exactitude du clusering est également comparée à celle de quatre autres méthodes de subspace clustering profond. Dans ce qui suit, nous commençons par décrire le cadre expérimental ainsi que les cirières d'évaluation. Par la suite, nous présentons les expérimentations et les résultats obtenus. Pour les résultats, dans un premier temps, nous analysons la qualité du résultat du clustering. Ensuite, nous analysons la scalabilité pour déterminer comment le temps d'exécution évolue par rapport à la taille des données. En dernier, nous analysons le comportement de convergence de la matrice d'affinité.

4.1 Cadre expérimental :

4.1.1 Modèles sélectionnés pour la comparaison :

Afin d'illustrer les performances de l'approche proposée, nous avons sélectionné plusieurs méthodes décrites dans l'état de l'art. Premièrement, nous comparons notre approche aux méthodes de subspace clustering scalable. La comparaison est faite avec les méthodes qui se basent sur la technique d'échantillonnage, tel que **SGL** (Kang *et al.*, 2021a), **S^{5}C** (Matsushima et Brbic, 2019), **LMVSC** (Kang *et al.*, 2020). Également, avec les méthodes qui n'exploitent pas l'échantillonnage, tel que **SSC-OMP** (You *et al.*, 2016b), **EnSC** (You *et al.*, 2016a), **SSSC** (Chen *et al.*, 2020), **A-DSSC** (Lim *et al.*, 2020).

Dans l'absence des travaux traitant la scalabilité du subspace clustering profond, en utilisant la propriété d'autoreprésentation, nous considérons également les méthodes de subspace clustering profond telles que, **DSC** (Peng *et al.*, 2020), **DSC-Net** (Ji *et al.*, 2017), **S^{2}ConvScn** (Zhang *et al.*, 2019a), **DLRSC** (Kheirandishfard *et al.*, 2020).

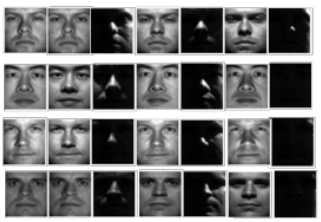
4.1.2 Ensemble de données utilisées pour la comparaison :

La comparaison des méthodes décrites dans la section précédente est faite sur cinq ensembles de données citées et utilisées dans la littérature, ainsi que sur des données synthétiques. Une description sommaire est fournie dans les tableaux 4.1, 4.2 pour chaque ensemble de données.

Dans ce qui suit, nous donnons une description un peu plus détaillée des données utilisées :

TABLEAU 4.1 – Description des ensembles de données réelles.

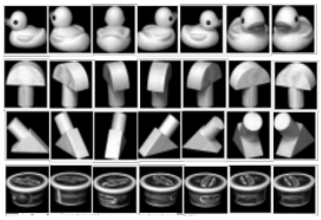
Données	Taille	Groupes	Dimensions
YaleB	2 432	38	48 x 42
ORL	400	40	32 x 32
Coil100	7 200	100	32 x 32
UMIST	480	20	32 x 32
Fashion-MNIST	60 000	10	28 x 28



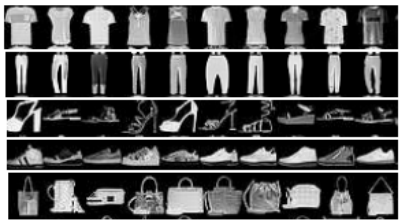
(a) Yaleb.

(b) ORL.

(c) UMIST.



(d) Coil100.



(e) Fashion-MNIST.

FIGURE 4.1 – Exemples d'images tirées des ensembles de données réelles.

1. **YaleB** (Lee *et al.*, 2005) est une référence populaire pour le subspace clustering avec 2 432 images. Les données se composent de 38 sujets, chacun étant représenté par 64 images de visage obtenues dans différentes conditions d'éclairage (voir la figure 4.1a pour des exemples d'images de ce jeu de données).
2. **ORL** (Samaria et Harter, 1994) est composé de 400 images de visages, avec 40 sujets ayant chacun 10 images. Pour chaque sujet, les images ont été prises dans des conditions d'éclairage variables avec différentes expressions faciales (yeux ouverts, ou fermés, et avec ou sans sourire). Également,

avec d'autres détails du visage (avec ou sans lunettes) (voir Figure 4.1b pour des exemples d'images). Cet ensemble de données est plus difficile à regrouper en raison que : (1) les sous-espaces du visage présentent plus de non-linéarité à cause de la variation des expressions et des détails du visage, (2) la taille de l'ensemble de données est plus petite (400 images).

3. **UMIST** (Graham et Allinson, 1998) se compose de 564 images de visages avec 20 sujets distincts. Les visages couvrent une gamme mixte de poses, de races, de sexe et d'apparences telles que différentes expressions, illuminations, avec ou sans lunettes, avec ou sans barbe et différentes coiffures (voir figure 4.1c pour des exemples d'images).

4. **Coil100** (Nene *et al.*, 1996) contient 7 200 échantillons d'images, répartis sur 100 objets tels que le canard et des modèles de voitures (voir figure 4.1d pour des exemples d'images). Chaque objet a été placé sur un fond noir, et 72 images ont été prises avec des intervalles d'angle de vue de 5 degrés. Les objets dans COIL100 sont davantage variés, et les images d'un même objet diffèrent les unes des autres en raison du changement d'angle de vue. Cela pose un défi aux techniques du subspace clustering.

5. **Fashion-MNIST** (Xiao *et al.*, 2017) se compose d'un ensemble d'images de 10 produits de mode. Des photos sont prises par des photographes professionnels montrant différents aspects du produit (des photos à partir de différents angles, mettre en avant les petits détails, avec un modèle portant le produit), voir figure 4.1e pour des exemples d'images. Pour notre expérimentation, nous utilisons trois sous-ensembles à partir de cet ensemble de données :

   (a) **TestFashion-MNIST** : avec 10 000 images de 10 produits de mode.

   (b) **Fashion-MNIST_20K** : avec 20 000 images de 10 produits de mode.

   (c) **Fashion-MNIST_30K** : avec 30 000 images de 10 produits de mode.

6. **Données synthétiques** : nous suivons la configuration utilisée dans (Chen *et al.*, 2020) pour générer aléatoirement $s = 10$ sous-espaces avec des dimensions aléatoires $d_i = [6, 12]$ dans un espace ambiant $D = 784$. Chaque sous-espace contient n_i points de données aléatoires, n_i varie de 30 à 1000. Ainsi, le nombre total n de points de données varie de 300 à 10 000.

TABLEAU 4.2 – Description des données synthétiques utilisées.

Dimension D	**Dimension** d_i	# **sous-espaces** s	**points** n_i	**Taille**
784	$d_i \in [6, 12]$	10	[30, 1000]	$s \times n_i = [300, 10\,000]$

4.1.3 Critères d'évaluation :

Pour évaluer la performance de notre approche ainsi que les approches sélectionnées pour la comparaison, nous considérons un ensemble de critères d'évaluation. Nous utilisons cinq métriques standard citées et appliquées dans la littérature pour comparer les performances de clustering des différentes méthodes sur les ensembles de données décrits dans la section précédente.

Premièrement, nous évaluons l'exactitude du clustering (Accuracy : ACC) en calculant le taux de clustering correct moyen. Cette métrique mesure le taux de correspondance correcte entre les étiquettes [1] identifiées, par les différentes méthodes de subspace clustering, et les vraies étiquettes équivalentes. L'exactitude admet des valeurs entre 0 et 100. Une valeur proche de 0 désigne une mauvaise correspondance, tandis qu'une valeur proche de 100 désigne une forte correspondance. L'exactitude (ACC) est définie comme suit :

$$ACC\% = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[y_i = map(\hat{y}_i)] \times 100\% \tag{4.1}$$

1. Les données originales sont étiquetées pour faire correspondre chaque image à son cluster équivalent.

où $\hat{y}_i$ correspond à l'étiquette i identifiée par le clustering, et y_i correspond à la vraie étiquette. $\mathbb{I}(y_i, \hat{y}_i)$ est une fonction qui vaut 1 si $y_i = \hat{y}_i$ et 0 sinon, et $map(\hat{y}_i)$ est une fonction de correspondance qui permute les étiquettes produites par le clustering pour les faire correspondre aux vraies étiquettes.

Nous évaluons aussi le taux d'information mutuelle normalisée (Normalized Mutual Information : NMI). Cette métrique permet de mesurer la qualité du clustering et la corrélation entre les étiquettes identifiées par le clustering et les vraies étiquettes. Le taux (en pourcentage) d'information mutuelle normalisée admet des valeurs entre 0% (aucune corrélation) et 100% (corrélation parfaite). La métrique NMI est définie comme suit :

$$NMI(Y, \hat{Y}) = \frac{2I(Y, \hat{Y})}{[H(Y) + H(\hat{Y})]} \times 100\% \tag{4.2}$$

où Y désigne l'ensemble des vraies étiquettes, $\hat{Y}$ désigne l'ensemble des étiquettes identifiées par le clustering, $I(Y, \hat{Y})$ représente l'information mutuelle entre Y et $\hat{Y}$, et H(.) désigne l'entropie.

La troisième métrique consiste à mesurer l'erreur de préservation des sous-espaces (Subspace Preserving Error : SPE). Spécifiquement, SPE mesure pour chaque vecteur C_j dans la matrice d'affinité l'erreur d'appartenance des points de données c_{ij} dans son propre sous-espace j. Une valeur nulle indique que tous les points dans C_j appartiennent au sous-espace j, tandis que l'erreur égale à 1 correspond à ce que tous les points appartiennent à d'autres sous-espaces. La métrique SPE est définie comme suit :

$$SPE = \frac{1}{N} \sum_j \left(1 - \frac{\sum_i |C_{ij}|}{|C_j|}\right) \tag{4.3}$$

Nous évaluons également la connectivité du graphe de similarité (CONN). Cette métrique identifie si les points de données dans chaque cluster forment un composant connecté du graphe. Nous utilisons la deuxième plus petite valeur propre du Laplacien normalisé (Chen *et al.*, 2020). Soit $\lambda_2^{(i)}$ la deuxième plus petite valeur propre du Laplacien normalisé correspondant au i-ème cluster. La connectivité est calculée comme suis :

$$CONN = min_i\{\lambda_2^{(i)}\}_{i=1}^{n} \tag{4.4}$$

La dernière métrique utilisée pour la comparaison consiste à mesurer le temps d'exécution relatif à la construction de la matrice d'affinité.

4.2 Expérimentations

4.2.1 Déroulement des expérimentations :

Nous commençons par des expérimentations sur les modèles de subspace clustering scalable. Pour tous les modèles (**SGL, S^{5}C, LMVSC, SSC-OMP, EnSC, SSSC**), nous avons utilisé les codes sources publiés par les auteurs, et nous avons ajusté les paramètres pour obtenir les meilleurs résultats sur chaque ensemble de données. Dans l'indisponibilité du code source pour le modèle **A-DSSC**, la comparaison est faite avec les résultats disponibles pour Yaleb, ORL, et UMIST publiés dans l'article correspondant.

Pour les modèles de subspace clustering profond (**DSC, DSCNet, S^{2}ConvSen, DLRSC**), nous rapportons les résultats publiés dans les différents articles. Les expérimentations présentées dans ces articles ne sont faites que sur les ensembles de données YaleB, ORL et Coil100. De plus, les critères NMI, SPE et CONN ne

sont pas évalués. Pour cette raison, nous comparons uniquement les résultats de l'exactitude du clustering (ACC), sur les ensembles de données YaleB, ORL et Coil100 publiés dans les articles originaux de chaque approche.

Nous avons implémenté notre méthode en PyTorch avec Tensorflow-1.0. Nous avons utilisé un serveur avec un CPU Intel(R) Xeon(R) E5-2620 V4 @ 2.10GHz. Le serveur dispose d'un système d'exploitation Ubuntu (version : 18.04.5 LTS) sur lequel notre modèle a été implémenté avec le langage PyTorch (version : 3.7.4), ainsi que les librairies PyTorch (version : 1.3.1), et Sklearn (version : 0.23.1).

4.2.2 Paramétrage de l'approche proposée **SCS-P** :

Dans le tableau 4.3 nous présentons l'ensemble des paramètres utilisés pour préentrainer un auto-encodeur sans la couche d'autoreprésentation sur chaque ensemble de données. Le paramètre <Taux>, dans les tableaux ci dessous, correspond au taux d'apprentissage.

TABLEAU 4.3 – Configuration des paramètres de **SCS-P** pour chaque ensemble de données pour le préentrainement.

Données/Paramètres	**Taux**	**Filtre**	**Couches cachées**	**Itérations**	**Optimiseur**
YaleB	10^{-3}	[5, 3, 3]	[10, 20, 30]	1000	Adam
ORL	10^{-3}	[3, 3, 3]	[3, 3, 5]	1000	Adam
Coil100	10^{-4}	[5]	[50]	1000	Adam
UMIST	10^{-3}	[5, 3, 3]	[15, 10, 5]	700	Adam
TestFashion-MNIST	10^{-4}	[3, 3, 3]	[10, 20, 30]	1000	Adam
Fashion-MNIST_20k	10^{-4}	[3, 3, 3]	[10, 20, 30]	1000	Adam
Fashion-MNIST_30k	10^{-4}	[3, 3, 3]	[10, 20, 30]	1000	Adam

Pour l'étape d'entraînement, nous présentons dans le tableau 4.4, les paramètres utilisés pour chaque ensemble de données. Le paramètre <t> (à la sixième colonne du tableau) représente la fréquence de mise à jour, en termes de nombre d'itérations, de la matrice $P \in \mathbb{R}^{n \times m}$. Également, notre approche dépend de deux autres paramètres de la couche d'autoreprésentation présentés dans le tableau

4.5. Notamment, le nombre de points de repère utilisés pour l'identification des sous-espaces, et l'hyperparamètre λ.

TABLEAU 4.4 – Configuration des paramètres de **SCS-P** pour chaque ensemble de données pour l'entraînement.

Données/Paramètres	**Taux**	**Filtre**	**Couches cachées**	**Itération**	***t***	**Optimiseur**
YaleB	10^{-3}	[5, 3, 3]	[10, 20, 30]	8000	100	Adam
ORL	10^{-3}	[3, 3, 3]	[3, 3, 5]	6000	100	Adam
Coil100	10^{-4}	[5]	[50]	1000	100	Adam
UMIST	10^{-3}	[5, 3, 3]	[15, 10, 5]	700	100	Adam
TestFashion-MNIST	10^{-4}	[3, 3, 3]	[10, 20, 30]	800	100	Adam
Fashion-MNIST_20k	10^{-4}	[3, 3, 3]	[10, 20,3 0]	1000	100	Adam
Fashion-MNIST_30k	10^{-4}	[3, 3, 3]	[10, 20, 30]	1000	100	Adam

TABLEAU 4.5 – Configuration des paramètres de la couche autoreprésentation de **SCS-P** pour chaque ensemble de données pour l'entraînement.

Données/Paramètres	**Taille**	**Points de repère**	λ
YaleB	2 432	380	10^{-4}
ORL	400	40	4×10^{-2}
Coil100	7 200	1000	4×10^{-2}
UMIST	480	33	10^{-4}
TestFashion-MNIST	10 000	700	10^{-1}
Fashion-MNIST_20k	20 000	1 200	10^{-1}
Fashion-MNIST_30k	30 000	1 800	10^{-1}

Pour les données synthétiques, nous faisons varier le nombre de points n_i dans chaque sous-espace. Ainsi, pour $n_i \in [30, 1000]$ nous évaluons notre approche sur un ensemble de données synthétiques de taille $n \in [300, 10\,000]$. Les tableaux 4.6 et 4.7 présentent les paramètres utilisés pour les deux étapes, préentrainement et entraînement.

TABLEAU 4.6 – Configuration des paramètres de **SCS-P** pour les données synthetiques.

Étape/Paramètres	**Taux**	**Filtre**	**Couches cachées**	**Itération**	***t***	**Optimiseur**
Preentrainement	10^{-5}	[5]	[10]	300	-	Adam
Entrainement	10^{-5}	[5]	[10]	500	100	Adam

TABLEAU 4.7 – Configuration des parametres de la couche autoreprésentation pour les données synthetiques pour l'entraînement.

n_i	**Taille des données**	**Points de repère**	λ
[30-1000]	$n_i \times 10$	n_i	10^{-5}

4.2.3 Résultats et discussion :

L'objectif de cette section est d'analyser les résultats des expérimentations sur les différents ensembles de données :

1. Analyse et comparaison de la qualité du clustering **SCS-P** par rapport aux méthodes de subspace clustering scalable, et aux méthodes de subspace clustering profond ;
2. Analyse de la scalabilité de **SCS-P** sur les données synthétiques par rapport aux approches de subspace clustering scalable (EnSC, SSC-OMP, LMV, SGL, S^5C) ;
3. Analyse du comportement de convergence de la matrice d'affinité C de **SCS-P**.

Qualité du clustering :

Nous comparons l'ACC, NMI et SPE de **SCS-P** par rapport aux modèles scalables de l'état de l'art. Les tableaux 4.8, 4.9 et 4.10 présentent les performances de chaque modèle, évalués sur les ensembles de données (UMIST et Coil100), (YaleB et ORL), et (TestFashion-MNIST, Fashion-MNIST_20k et Fashion-MNIST_30k) respectivement.

Dans l'ensemble, les résultats révèlent que notre approche, en termes de qualité du clustering, dépasse les résultats des méthodes LMVSC, SGL et S^5C qui se basent sur les techniques d'échantillonnage. Notre approche dépasse les méthodes

TABLEAU 4.8 – Comparaison des performances du clustering de notre approche avec les approches de subspace clustering scalable sur les ensembles de données UMIST et Coil100.

Méthodes	**UMIST**			**Coil100**		
	ACC	**NMI**	**SPE**	**ACC**	**NMI**	**SPE**
SSC-OMP	67.7	77.7	0.30	59.46	82.02	0.28
EnSC	61.03	74.12	0.35	67.34	88.53	0.25
SSSC	61.79	72.95	0.45	66.89	88.11	0.03
A-DSSC	72.5	85.1	**0.03**	**82.4**	**94.6**	**0.03**
LMVSC	60.62	74.80	0.4	49.5	71.78	0.55
SGL	65	78.48	0.16	47.57	72.33	0.66
S5C	69.79	78.62	0.08	55.25	79.70	0.17
SCS-P	**83.33**	**90.15**	0.09	71.10	90.95	0.24

TABLEAU 4.9 – Comparaison des performances du clustering de notre approche avec les approches de subspace clustering scalable sur les ensembles de données Yaleb et ORL.

Méthodes	**Yaleb**			**ORL**		
	ACC	**NMI**	**SPE**	**ACC**	**NMI**	**SPE**
SSC-OMP	75.03	80.27	0.17	70	84.44	0.28
EnSC	88.19	89.02	0.30	79.43	90.28	0.63
SSSC	80.67	84.78	0.19	74.72	85.43	0.52
A-DSSC	91.7	94.7	**0.08**	79.0	**91.0**	**0.15**
LMVSC	21.5	45.74	0.32	50	68.24	0.49
SGL	21.5	47.58	0.78	56.75	70.80	0.58
S5C	60.81	65.55	0.29	72.75	85.21	0.44
SCS-P	**96.46**	**95.49**	0.24	**85.5**	89.60	0.54

LMVSC, SGL en moyenne de 74.96% pour YaleBet, 20.52% pour UMIST, 32.12% pour ORL, 22.56% pour Coil100 et 5.36% pour Fashion-MNIST. En effet, contrairement à notre approche, LMVSC et SGL se basent sur une sélection aléatoire ne permettant pas de bien identifier les sous-espaces. La méthode S^5C utilise une sélection ciblée permettant de mieux identifier les sous-espaces que LMVSC et SGL mais se base sur des conditions théoriques trop restrictives et reste moins performante. Nos résultats dépassent les résultats de la méthode S^5C en moyenne de 35.65% pour YaleBet, 13.54% pour UMIST, 12.75% pour ORL, 15.85% pour

TABLEAU 4.10 – Comparaison des performances du clustering de notre approche avec les approches de subspace clustering scalable sur TestFashion-MNIST, Fashion-MNIST_20k et Fashion-MNIST_30k. (M : Limite de mémoire)

Méthodes	**TestFashion-MNIST**			**Fashion-MNIST_20k**			**Fashion-MNIST_30k**		
	ACC	**NMI**	**SPE**	**ACC**	**NMI**	**SPE**	**ACC**	**NMI**	**SPE**
SSC-OMP	44.47	49.79	0.57	37.94	44.44	0.58	36.24	42.78	0.57
EnSC	60.21	61.93	0.33	61.69	61.61	0.31	61.35	62.27	0.28
SSSC	60.35	**61.98**	**0.30**	60.37	63.08	0.24	60.50	**63.12**	**0.23**
LMVSC	60.64	52.39	0.39	61.58	52.21	0.40	59.18	50.44	M
SGL	60.16	53.97	0.55	55.20	56.24	0.49	54.40	55.95	M
S5C	59.15	61.55	0.35	65	78.48	**0.16**	59.78	61.74	M
SCS-P	**63.64**	60.87	0.32	**66.41**	**68.70**	0.26	**61.86**	61.39	0.61

Coil100 et 2.66% pour Fashion-MNIST. Également, nos résultats dépassent les résultats des méthodes SSC-OMP, EnSC, SSSC et A-DSSC en moyenne de 12.56% pour YaleBet, 17.57% pour UMIST, 9.71% pour ORL, 2.07% pour Coil100 et 10.28% pour Fashion-MNIST. Contrairement aux autres méthodes, notre méthode se base sur un apprentissage profond pour apprendre un sous-ensemble de données permettant de mieux identifier les sous-espaces à moindre coût. De plus, les résultats sur le NMI et SPE montrent que notre approche est aussi capable de préserver les sous-espaces.

TABLEAU 4.11 – Comparaison de l'exactitude du clustering de notre approche avec des approches de subspace clustering profond sur les bases de données ORL, Yaleb et Coil100. (- : donnée non disponible).

Méthodes	**ORL**	**Yaleb**	**Coil100**
	ACC	**ACC**	**ACC**
DSC	-	-	69.63
DSCNet	86	97.33	69.04
DLRSC	83	97.53	71.86
S2ConvSCN	**89.5**	**98.48**	**73.33**
SCS-P	85.5	96.46	71.10

Le tableau 4.11 présente une comparaison de l'exactitude de clustering de **SCS-P** avec les méthodes de subspace clustering profond qui ne sont pas scalable. Ces

derniers, il faut le rappeler, utilisent la similarité entre chaque point et tous les autres points de données lors de la construction de la matrice d'affinité. Les résultats de notre approche sont compétitifs aux résultats obtenus par les approches de subspace clustering profond avec un maximum d'écart de 4% pour ORL, 2.02% pour YaleB et 2.23% pour Coil100.

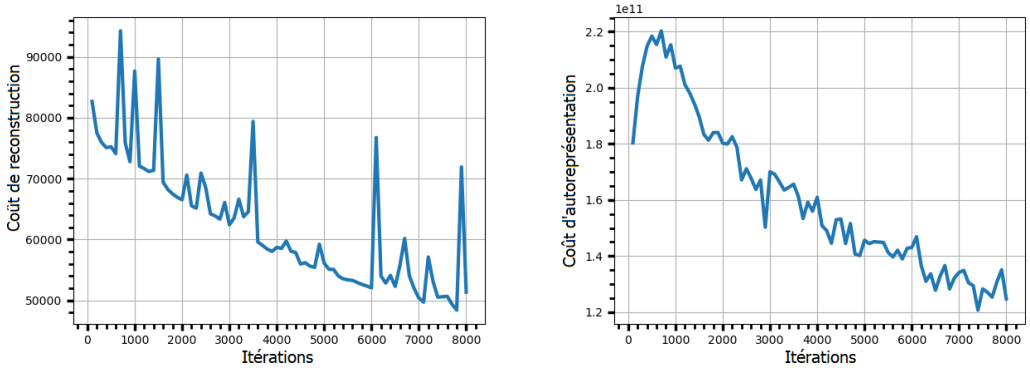


FIGURE 4.2 – Évolution du coût de la reconstruction et de l'autoreprésentation pour l'étape de l'entraînement à travers les itérations pour l'ensemble de données YaleB.

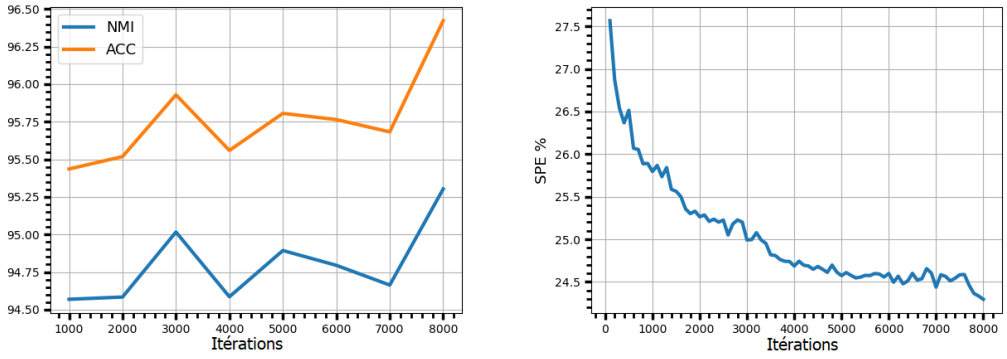


FIGURE 4.3 – Évolution de ACC, NMI et SPE à travers les itérations pour l'ensemble de données YaleB.

Aussi, nous observons les évolutions des différentes fonctions de coût, ACC, NMI et SPE à travers les itérations. Les figures 4.2 et 4.3 présentent les résultats des

expérimentations sur l'ensemble de données YaleB. Nous pouvons voir que le coût de reconstruction ainsi que celui de l'autoreprésentation continuent à diminuer au fil des itérations tandis que la précision du clustering et le NMI sont améliorés et que l'erreur de préservation des sous-espaces diminue.

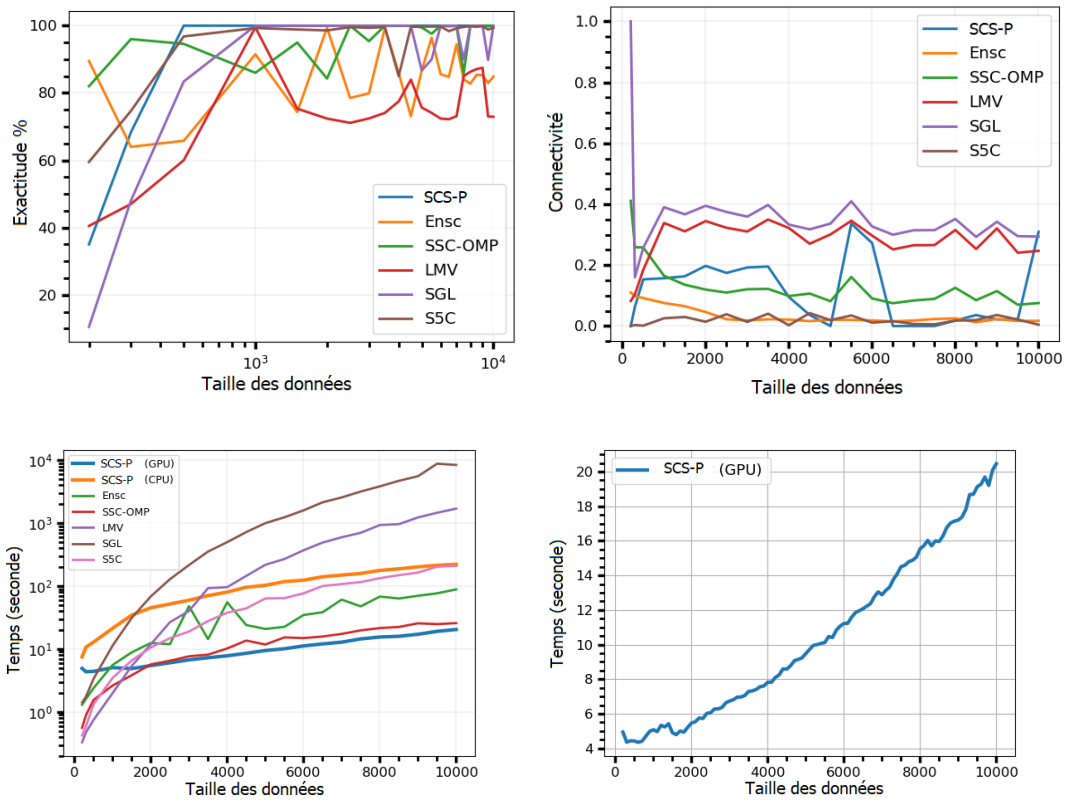


FIGURE 4.4 – Comparaison des performances de EnSC, SSC-OMP, LMV, SGL, S5C et SCS-P sur les données synthétiques.

Analyse de la scalabilité :

Nous comparons dans ce qui suit les performances en termes de scalabilité sur des données synthétiques. En faisant varier la taille des données, la figure 4.4 présente une comparaison des résultats sur l'exactitude, la connectivité et le temps d'exécution pour la construction de la matrice d'affinité. Étant donné que les méthodes

de subspace clustering scalable sélectionnées pour la comparaison s'exécutent sur un CPU, nous présentons deux résultats de temps d'exécution pour **SCS-P** : (1) temps d'exécution avec GPU et (2) temps d'exécution avec CPU.

Nous pouvons voir que **SCS-P** donne la meilleure exactitude de clustering tout en gardant, avec GPU, le plus petit coût de calcul, et un coût de calcul avec CPU comparable voire inférieur à celui des autres méthodes. Également, nous pouvons voir que SGL produit une meilleure connectivité qui est due à l'utilisation d'une contrainte de connectivité pour générer un graphe biparti à k composantes connexes, mais garde un coût de calcul le plus élevé.

Convergence de la matrice d'affinité :

Pour évaluer la convergence, nous montrons le changement relatif de la matrice d'affinité C entre deux itérations successives. La figure 4.5 présente les résultats sur les données synthétiques et les ensembles de données réelles. Nous observons que la matrice d'affinité se stabilise après quelques itérations. Ceci confirme la convergence de notre algorithme.

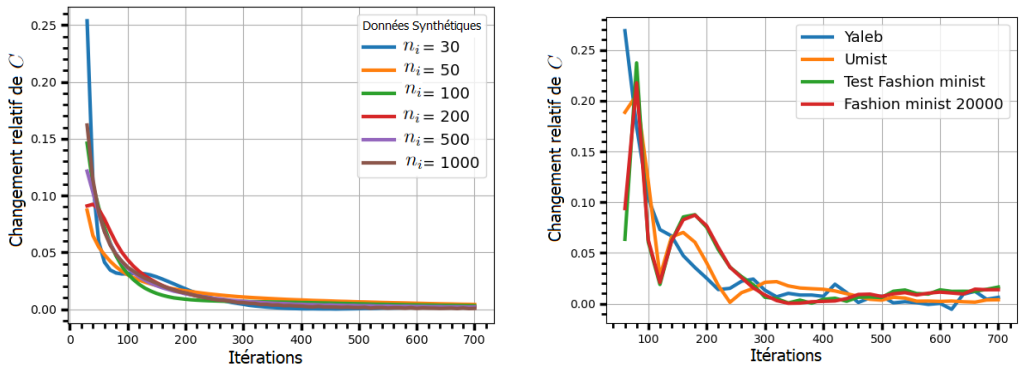
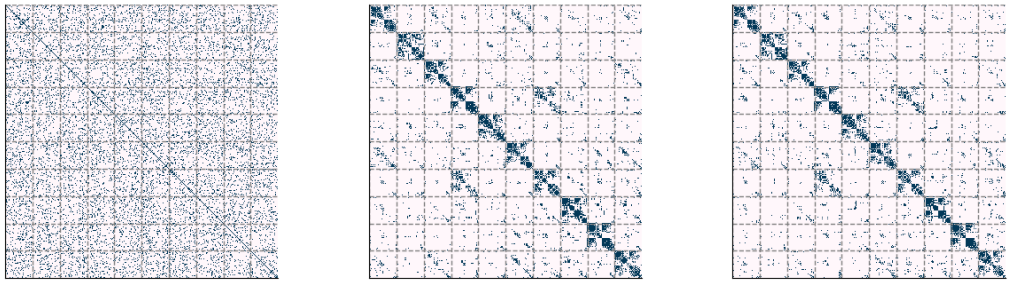


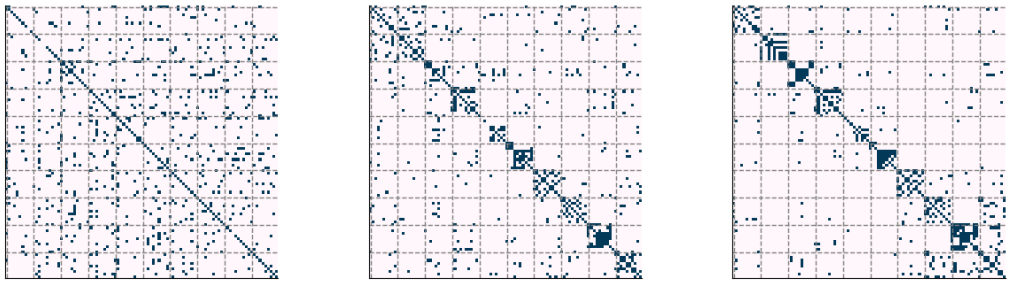
FIGURE 4.5 – Changement relatif de la matrice d'affinité C entre deux itérations successives.

Également, nous visualisons dans la figure 4.6 la matrice d'affinité à travers les itérations pour un sous-ensemble de l'ensemble de données YaleB composé de 10 groupes. Nous pouvons voir qu'au bout de quelques centaines d'itérations les clusters commencent déjà à être bien formés. Cela est dû à notre solution pour le calcul et la mise à jour de la matrice $P \in \mathbb{R}^{n \times m}$.



(a) Itération 0. (b) Itération 100. (c) Itération 500.

FIGURE 4.6 – Visualisation de la matrice d'affinité pour un sous-ensemble de données Yaleb avec 10 groupes.

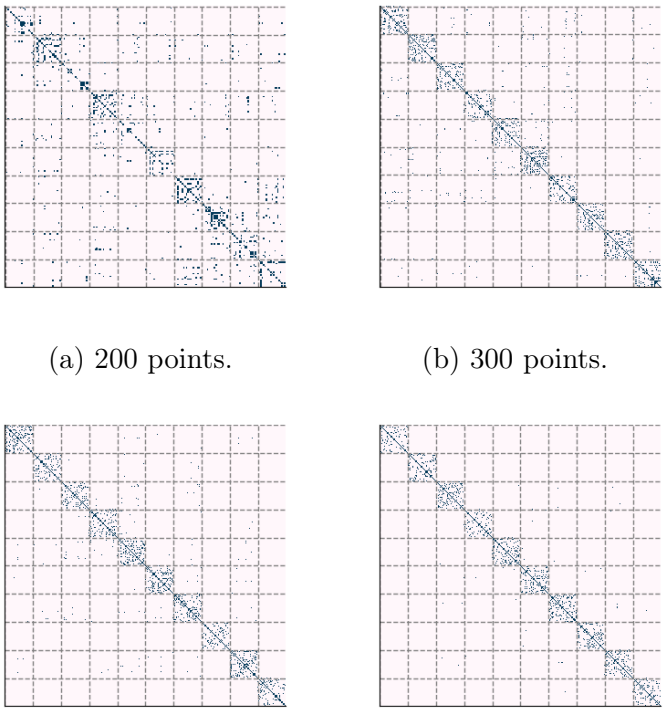


(a) Itération 0. (b) Itération 100. (c) Itération 500.

FIGURE 4.7 – Visualisation de la matrice d'affinité pour un sous-ensemble de données ORL avec 10 groupes.

La figure 4.7 montre la matrice d'affinité à travers les itérations pour un sous-ensemble de l'ensemble de données ORL composé de 10 groupes. Il est à rappeler

que cet ensemble de données est composé de 400 images de visages, avec 40 sujets ayant chacun 10 images, et que le regroupement est plus difficile en raison que les sous-espaces du visage présentent plus de non-linéarité et que la taille de l'ensemble de données est trop petite. Nous pouvons voir qu'au bout de quelques centaines d'itérations les clusters commencent déjà à être bien formés.



(a) 200 points. (b) 300 points.

(c) 400 points. (d) 500 points.

FIGURE 4.8 – Visualisation de la matrice d'affinité avec différentes tailles de l'ensemble de données synthétiques pour 500 itérations.

La figure 4.8 montre la matrice d'affinité avec différentes tailles d'ensembles de données composés de 10 groupes, à partir de l'ensemble de données synthétiques, pour un entraînement de 500 itérations. Nous pouvons voir qu'au fur et à mesure que la taille des données augmente, les clusters sont beaucoup mieux formés. Les expérimentations sur les données synthétiques confirment la capacité de notre

approche à identifier effectivement les sous-espaces.

4.3 Conclusion :

Dans ce chapitre, nous avons présenté les résultats des expérimentations de notre approche **SCS-P** sur des données synthétiques et réelles. Les résultats démontrent que **SCS-P** fournit une amélioration significative de la précision du clustering par rapport aux approches scalables existantes. En effet, notre méthode se base sur l'apprentissage profond, ce qui permet de prendre en considération des données complexes qui ne résident pas initialement dans des sous-espaces linéaires. Également, notre méthode permet d'apprendre une plus petite matrice permettant de réduire la complexité du subspace clustering tout en améliorant l'exactitude du clustering.

CHAPITRE V

CONCLUSION

Dans le cadre de ce mémoire, nous avons exposé la problématique de la scalabilité du subspace clustering. Les deux étapes du subspace clustering, à savoir, la construction de la matrice d'affinité et le clustering spectral, souffrent d'une complexité temporelle et spatiale élevée, ce qui rend difficile le regroupement de grands ensembles de données. Nous avons présenté un aperçu de quelques travaux existants traitant cette problématique. Toutefois, nous avons remarqué qu'il n'y avait pas de travaux traitant la scalabilité du subspace clustering dans le contexte de l'apprentissage profond.

Dans le cadre de notre travail, nous avons proposé une approche capable d'effectuer le subspace clustering profond en un temps linéaire par rapport à la taille des données. Le modèle présenté est la première proposition à cette problématique en utilisant la propriété d'autoreprésentation du subspace clustering dans le contexte d'apprentissage profond. Notre approche utilise l'auto-encodeur permettant de réduire la dimensionnalité des données et de prendre en considération des données qui résident dans des sous-espaces non linéaires. Dans une première étape, la couche d'autoreprésentation placée entre l'encodeur et le décodeur permet d'apprendre une matrice plus petite pour la construction de la matrice d'affinité. Dans la deuxième étape, cette petite matrice est utilisée pour le regroupement des don-

nées. L'utilisation d'une matrice plus petite a permis de réduire la complexité relative à la construction de la matrice d'affinité et à celle du clustering spectral de $O(n^3)$ vers une complexité linéaire $O(n)$.

Les résultats des expérimentations de notre solution montrent une amélioration significative de la précision du clustering par rapport aux solutions scalables existantes, et confirment la capacité de notre approche à identifier effectivement les sous-espaces.

Bien que nous ayons pu répondre aux objectifs de ce mémoire, quelques pistes d'amélioration demeurent envisageables. Par exemple, améliorer et optimiser les résultats du clustering en utilisant les techniques d'amélioration comme l'augmentation de données, ou encore en combinant les avantages de notre méthode avec celles des méthodes proposées du subspace clustering profond. Une autre piste serait de pouvoir identifier les dimensions des sous-espaces vectoriels. En effet, notre approche identifie les sous-espaces pour le regroupement des données, mais les dimensions des différents sous-espaces restent à identifier automatiquement. Ceci représente un défi que nous comptons relever dans la continuité de nos travaux sur le subspace clustering profond.

RÉFÉRENCES

Absil, P.-A., Mahony, R. et Sepulchre, R. (2004). Riemannian geometry of grassmann manifolds with a view on algorithmic computation. *Acta Applicandae Mathematica*, *80*(2), 199–220.

Adler, A., Elad, M. et Hel-Or, Y. (2012). Probabilistic subspace clustering via sparse representations. *IEEE Signal Processing Letters*, *20*(1), 63–66.

Agarwal, P. K. et Mustafa, N. H. (2004). K-means projective clustering. Dans *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 155–165.

Basri, R. et Jacobs, D. W. (2003). Lambertian reflectance and linear subspaces. *IEEE transactions on pattern analysis and machine intelligence*, *25*(2), 218–233.

Blondel, M., Seguy, V. et Rolet, A. (2018). Smooth and sparse optimal transport. Dans *International conference on artificial intelligence and statistics*, 880–889.

Chen, G. et Lerman, G. (2009). Spectral curvature clustering (scc). *International Journal of Computer Vision*, *81*(3), 317–330.

Chen, X. et Cai, D. (2011). Large scale spectral clustering with landmark-based representation. Dans *Twenty-fifth Association for the Advancement of Artificial Intelligence*.

Chen, Y., Li, C.-G. et You, C. (2020). Stochastic sparse subspace clustering. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4155–4164.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. et Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, *12*(ARTICLE), 2493–2537.

Costeira, J. P. et Kanade, T. (1998). A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, *29*(3), 159–179.

Dahl, G. E., Yu, D., Deng, L. et Acero, A. (2011). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, *20*(1), 30–42.

De la Torre, F. et Kanade, T. (2006). Discriminative cluster analysis. Dans *Proceedings of the 23rd international conference on Machine learning*, 241–248.

Dyer, E. L., Sankaranarayanan, A. C. et Baraniuk, R. G. (2013). Greedy feature selection for subspace clustering. *The Journal of Machine Learning Research*, *14*(1), 2487–2517.

Elhamifar, E. et Vidal, R. (2010). Clustering disjoint subspaces via sparse representation. Dans *IEEE International Conference on Acoustics, Speech and Signal Processing*, 1926–1929.

Elhamifar, E. et Vidal, R. (2013). Sparse subspace clustering : Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, *35*(11), 2765–2781.

Fang, Y., Wang, R., Dai, B. et Wu, X. (2014). Graph-based learning via auto-grouped sparse regularization and kernelized extension. *IEEE Transactions on Knowledge and Data Engineering*, *27*(1), 142–154.

Favaro, P., Vidal, R. et Ravichandran, A. (2011). A closed form solution to robust subspace estimation and clustering. Dans *Conference on Computer Vision and Pattern Recognition*, 1801–1807.

Gan, G., Ma, C. et Wu, J. (2020). *Data clustering : theory, algorithms, and applications*. Society for Industrial and Applied Mathematics.

Goh, A. et Vidal, R. (2007). Segmenting motions of different types by unsupervised manifold clustering. Dans *Conference on Computer Vision and Pattern Recognition*, 1–6.

Graham, D. B. et Allinson, N. M. (1998). Characterising virtual eigensignatures for general purpose face recognition. In *Face Recognition* 446–456.

Gruber, A. et Weiss, Y. (2004). Multibody factorization with uncertainty and missing data using the em algorithm. Dans *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, I–I.

Haeffele, B. D., You, C. et Vidal, R. (2020). A critique of self-expressive deep subspace clustering. *arXiv preprint arXiv :2010.03697*.

Ho, J., Yang, M.-H., Lim, J., Lee, K.-C. et Kriegman, D. (2003). Clustering appearances of objects under varying illumination conditions. Dans *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, I–I.

Horn, R. A. et Johnson, C. R. (2012). *Matrix analysis*. Cambridge university press.

Huang, P., Huang, Y., Wang, W. et Wang, L. (2014). Deep embedding network for clustering. Dans *22nd International conference on pattern recognition*, 1532–1537.

Ji, P., Salzmann, M. et Li, H. (2014). Efficient dense subspace clustering. Dans *IEEE Winter conference on applications of computer vision*, 461–468.

Ji, P., Zhang, T., Li, H., Salzmann, M. et Reid, I. (2017). Deep subspace clustering networks. *arXiv preprint arXiv :1709.02508*.

Kanatani, K.-i. (2001). Motion segmentation by subspace separation and model selection. Dans *Proceedings Eighth IEEE International Conference on computer Vision.*, volume 2, 586–591.

Kang, Z., Lin, Z., Zhu, X. et Xu, W. (2021a). Structured graph learning for scalable subspace clustering : From single view to multiview. *IEEE Transactions on Cybernetics*.

Kang, Z., Peng, C., Cheng, Q., Liu, X., Peng, X., Xu, Z. et Tian, L. (2021b). Structured graph learning for clustering and semi-supervised classification. *Pattern Recognition*, *110*, 107627.

Kang, Z., Zhou, W., Zhao, Z., Shao, J., Han, M. et Xu, Z. (2020). Large-scale multi-view subspace clustering in linear time. Dans *Proceedings of the Association for the Advancement of Artificial Intelligence*, volume 34, 4412–4419.

Kheirandishfard, M., Zohrizadeh, F. et Kamangar, F. (2020). Deep low-rank subspace clustering. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 864–865.

Kriegel, H.-P., Kröger, P. et Zimek, A. (2009). Clustering high-dimensional data : A survey on subspace clustering, pattern-based clustering, and correlation clustering. *Acm transactions on knowledge discovery from data (tkdd)*, *3*(1), 1–58.

Krizhevsky, A., Sutskever, I. et Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

Lee, K.-C., Ho, J. et Kriegman, D. J. (2005). Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on pattern analysis and machine intelligence*, *27*(5), 684–698.

Li, J. et Zhao, H. (2017). Large-scale subspace clustering by fast regression coding. Dans *International Joint Conference on Artificial Intelligence*.

Lim, D., Vidal, R. et Haeffele, B. D. (2020). Doubly stochastic subspace clustering. *arXiv preprint arXiv :2011.14859*.

Lin, Z., Liu, R. et Su, Z. (2011). Linearized alternating direction method with adaptive penalty for low-rank representation. *Advances in neural information processing systems*, *24*.

Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y. et Ma, Y. (2012). Robust recovery of subspace structures by low-rank representation. *IEEE transactions on pattern analysis and machine intelligence*, *35*(1), 171–184.

Liu, G., Lin, Z., Yu, Y. *et al.* (2010). Robust subspace segmentation by low-rank representation. Dans *International Conference on Machine Learning*, volume 1, p. 8.

Lorenz, D. A., Manns, P. et Meyer, C. (2021). Quadratically regularized optimal transport. *Applied Mathematics and Optimization*, *83*(3), 1919–1949.

Lu, C.-Y., Min, H., Zhao, Z.-Q., Zhu, L., Huang, D.-S. et Yan, S. (2012). Robust and efficient subspace segmentation via least squares regression. Dans *European conference on computer vision*, 347–360.

Matsushima, S. et Brbic, M. (2019). Selective sampling-based scalable sparse subspace clustering. *Advances in Neural Information Processing Systems*, *32*.

Nasihatkon, B. et Hartley, R. (2011). Graph connectivity in sparse subspace clustering. Dans *Conference on Computer Vision and Pattern Recognition*, 2137–2144.

Nene, S. A., Nayar, S. K., Murase, H. *et al.* (1996). Columbia object image library (coil-100).

Nie, F., Wang, X., Jordan, M. et Huang, H. (2016). The constrained laplacian rank algorithm for graph-based clustering. Dans *Proceedings of the Association for the Advancement of Artificial Intelligence*, volume 30.

Panagakis, Y. et Kotropoulos, C. (2014). Elastic net subspace clustering applied to pop/rock music structure analysis. *Pattern Recognition Letters*, *38*, 46–53.

Parsons, L., Haque, E. et Liu, H. (2004). Subspace clustering for high dimensional data : a review. *Acm sigkdd explorations newsletter*, *6*(1), 90–105.

Pati, Y. C., Rezaiifar, R. et Krishnaprasad, P. S. (1993). Orthogonal matching pursuit : Recursive function approximation with applications to wavelet decomposition. Dans *Proceedings of 27th Asilomar conference on signals, systems and computers*, 40–44.

Peng, X., Feng, J., Zhou, J. T., Lei, Y. et Yan, S. (2020). Deep subspace clustering. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(12), 5509–5521.

Rokach, L. et Maimon, O. (2005). Clustering methods. In *Data mining and knowledge discovery handbook* 321–352.

Rontsis, N. et Goulart, P. (2020). Optimal approximation of doubly stochastic matrices. Dans *International Conference on Artificial Intelligence and Statistics*, 3589–3598.

Samaria, F. S. et Harter, A. C. (1994). Parameterisation of a stochastic model for human face identification. Dans *Proceedings of IEEE workshop on applications of computer vision*, 138–142.

Soltanolkotabi, M. et Candes, E. J. (2012). A geometric analysis of subspace clustering with outliers. *The Annals of Statistics*, *40*(4), 2195–2238.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. et Salakhutdinov, R. (2014). Dropout : a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Sugaya, Y. et Kanatani, K. (2004). Geometric structure of degeneracy for multi-body motion segmentation. Dans *International Workshop on Statistical Methods in Video Processing*, 13–25.

Tipping, M. E. et Bishop, C. M. (1999). Mixtures of probabilistic principal component analyzers. *Neural computation*, *11*(2), 443–482.

Tseng, P. (2000). Nearest q-flat to m points. *Journal of Optimization Theory and Applications*, *105*(1), 249–252.

Van Der Maaten, L., Postma, E., Van den Herik, J. *et al.* (2009). Dimensionality reduction : a comparative. *Journal of Machine Learning Research*, *10*(66-71), 13.

Vidal, E. E. R. (2009). Sparse subspace clustering. Dans *IEEE Conference on Computer Vision and Pattern Recognition*, volume 6, 2790–2797.

Vidal, R. (2011). Subspace clustering. *IEEE Signal Processing Magazine*, *28*(2), 52–68.

Vidal, R. et Favaro, P. (2014). Low rank subspace clustering (lrsc). *Pattern Recognition Letters*, *43*, 47–61.

Vidal, R., Ma, Y. et Sastry, S. (2005). Generalized principal component analysis (gpca). *IEEE transactions on pattern analysis and machine intelligence*, *27*(12), 1945–1959.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, *17*(4), 395–416.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y. et Fergus, R. (2013). Regularization of neural networks using dropconnect. Dans *International conference on machine learning*, 1058–1066.

Wang, S., Tu, B., Xu, C. et Zhang, Z. (2014). Exact subspace clustering in linear time. Dans *Proceedings of the Association for the Advancement of Artificial Intelligence*, volume 28.

Wang, X., Nie, F. et Huang, H. (2016). Structured doubly stochastic matrix for graph based clustering : Structured doubly stochastic matrix. Dans *Proceedings of the 22nd ACM SIGKDD International conference on Knowledge discovery and data mining*, 1245–1254.

Wang, Y.-X., Xu, H. et Leng, C. (2013). Provable subspace clustering : When lrr meets ssc. *Advances in Neural Information Processing Systems*, *26*.

Xiao, H., Rasul, K. et Vollgraf, R. (2017). Fashion-mnist : a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv :1708.07747*.

Xie, J., Girshick, R. et Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. Dans *International conference on machine learning*, 478–487.

Yan, J. et Pollefeys, M. (2006). A general framework for motion segmentation : Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. Dans *European conference on computer vision*, 94–106.

Ye, J., Zhao, Z. et Wu, M. (2007). Discriminative k-means for clustering. *Advances in neural information processing systems*, *20*.

You, C., Li, C., Robinson, D. P. et Vidal, R. (2018). Scalable exemplar-based subspace clustering on class-imbalanced data. Dans *Proceedings of the European Conference on Computer Vision*, 67–83.

You, C., Li, C.-G., Robinson, D. P. et Vidal, R. (2016a). Oracle based active set algorithm for scalable elastic net subspace clustering. Dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3928–3937.

You, C., Robinson, D. et Vidal, R. (2016b). Scalable sparse subspace clustering by orthogonal matching pursuit. Dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3918–3927.

You, C. et Vidal, R. (2015). Geometric conditions for subspace-sparse recovery. Dans *International Conference on Machine Learning*, 1585–1593.

Zelnik-Manor, L. et Irani, M. (2003). Degeneracies, dependencies and their implications in multi-body and multi-sequence factorizations. Dans *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, II–287.

Zhang, J., Li, C.-G., You, C., Qi, X., Zhang, H., Guo, J. et Lin, Z. (2019a). Self-supervised convolutional subspace clustering network. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5473–5482.

Zhang, T., Ji, P., Harandi, M., Hartley, R. et Reid, I. (2018). Scalable deep k-subspace clustering. Dans *Asian Conference on Computer Vision*, 466–481.

Zhang, T., Ji, P., Harandi, M., Huang, W. et Li, H. (2019b). Neural collaborative subspace clustering. Dans *International Conference on Machine Learning*, 7384–7393.

Zhang, T., Szlam, A., Wang, Y. et Lerman, G. (2012). Hybrid linear modeling via local best-fit flats. *International journal of computer vision*, *100*(3), 217–240.

Zou, H. et Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the royal statistical society : series B (statistical methodology)*, *67*(2), 301–320.