Université Paris Descartes

MLSD 2 2018-2019

Apprentissage profond pour la réduction de dimension

Professeur :
Mohamed Nadif
LIPADE

Auteurs :
Joseph Gesnouin
Yannis Tannier

Contents

List of Figures

List of Tables

1 Introduction

Apprendre des jeux de données mutidimentionels n'est pas une tâche aisée. Tandis que les jeux de données de faibles dimension sont très facilement manipulables car il est aisé de représenter la structure inhérente de leurs données, il est à l'inverse plus compliqué de réaliser des traitements pour des jeux de données de grande dimension. La réduction de dimension permet d'apprendre une nouvelle représentation des données à partir des données originales. Malheureusement, les méthodes classiques de réduction de dimension ne parviennent pas à apprendre avec précision une nouvelle représentation de données lorsque les données sont volumineuses. Ces méthodes souffrent de la malédiction de la dimensionnalié: des données fortement corrélées, données clairsemées, bruyantes et hétérogènes.

L'objectif de ce projet est de proposer une méthode combinant la réduction de dimension et l'apprentissage profond pour apprendre une représentation plus précise des données. Pour les données volumineuses, ces méthodes qui optimisent différents critères sont souvent utilisées séparément: un autoencodeur suivi d'une réduction de dimension. Cette approche présente un inconvénient majeur, à savoir le manque de connexion entre les deux tâches et affecte donc grandement la qualité de la représentation finale. De plus, la robustesse de ces approches est entravée par plusieurs problèmes de réglage de l'autoencodeur parmi lesquels figurent l'initialisation des poids, la largeur et le nombre de couches ou le nombre d'épochs. Atténuer l'impact d'un tel réglage d'hyperparamètres sur la performance des méthodes serait possible en combinant ces deux approches au lieu de les réaliser séparement.

Dans un premier temps, nous nous interesserons aux méthodes classiques de réduction de dimension en réalisant un état de l'art de la réduction de dimension, puis nous proposerons une approche simultanée combinant l'apprentissage profond via une architecture autoencodeur et les techniques de reduction de dimension telles que LLE.

2 Etat de l'art de la reduction de dimension

La famille des méthodes de réduction de dimension peut se subdiviser en deux familles de méthodes:

- Celles dont l'approche est linéaire: les axes utilisés sont des combinaisons linéaires des variables initiales. Ces algorithmes essaient de choisir la projection linéaire "optimale/intéressante" de la data sur un espace vectoriel reduit. Elles peuvent s'avérer puissantes, mais ne prennent pas en compte l'importance de certaines structures non-linéaires dans la data.
- Les approches non-linéaires: La plupart de ces méthodes s'inspirent des méthodes linéaires telle que la PCA. Ces approches non-linéaires s'avèrent plus sensibles à certaines structures de données (32). Contrairement aux méthodes linéaires facilement généralisables à divers problèmes de réduction de dimension, il peut être compliqué de réussir à généraliser ce type d'algorithme non linéaire afin qu'il soit performant dans tous les types de structure non linéaire qu'un data scientist est amené à rencontrer.

Ces deux types de méthodes se basent toutes deux sur l'idée que la dimension de beaucoup de jeux de données est souvent artificiellement élevée et qu'il est possible de réduire celle-ci sans relever une perte notable d'information.

2.1 L'ACP une approche linéaire

L'ACP (1) consiste à transformer des variables corrélées en axes décorrélés les un des autres. Ces axes sont composés uniquement de combinaisons linéaires des variables précédentes. Cette méthode est une des première méthode d'analyse factorielle et moteur central de toute autre méthode d'analyse factorielle. La question de l'ACP se ramène à un problème de diagonalisation de la matrice de variance covariance. L'ACP est majoritairement utilisée pour débruiter des données en considérant que les vecteurs propres que l'on décide d'oublier sont des axes bruités, mais sert également à décorreler les axes les uns aux autres. (2)(3)(4)

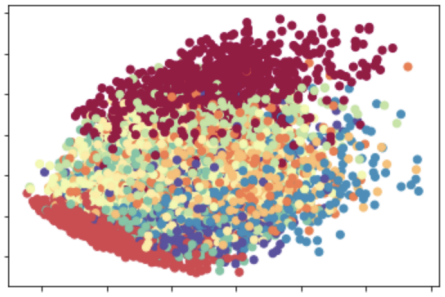


Figure 1: PCA sur le jeu de données MNIST

La fonction objectif J de la PCA et sa prédiction pouvant s'écrire

$$J = \sum_{n=1}^{N} |x(n) – \hat{x}(n)|^2 \quad \hat{x}(n) = Q^{-1}Qx(n)$$

Avec Q pouvant correspondre à la matrice de transformation complète, ce qui donnerait exactement l'entrée, ou une matrice de rang k, conservant alors les vecteurs propres les plus pertinents et donc une approximation de l'entrée.

La fonction objectif de la PCA peut donc s'écrire sous la forme:

$$J = \sum_{n=1}^{N} |x(n) – Q^{-1}Qx(n)|^2$$

2.2 Approches non-linéaires classiques

2.2.1 Kernel-ACP

Kernel-PCA: extension directe de l'ACP, K-ACP utilise la fonction kernel afin de capturer au mieux la non-linéarité:la transformation apprise par l'ACP est linéaire. Une version non-linéaire peut s'obtenir en appliquant l'ACP sur les données transformées par une transformation non linéaire. Ainsi, pour réussir à capturer un peu mieux la non-linéarité, K-ACP utilise une méthode d'augmentation de la dimension. Suite à la création d'une variable supplémentaire, K-ACP diminue la dimension tout en utilisant l'information générée par cette nouvelle variable: Cette méthode peut parraitre contradictoire avec la finalité de la reduction de la dimentionalité, mais fonctionne très bien en pratique: On passe alors d'un jeu de données aux classes

inséparables linéairement en dimension n à un jeu de données aux classes séparables linéairement en dimension n+1. (5)(6)

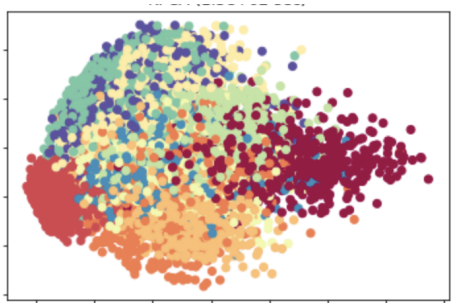


Figure 2: K-PCA sur le jeu de données MNIST

2.2.2 Multi-Dimensional Scaling

Multi-Dimentional Scaling(7) cherche une représentation des données dans un espace vectoriel réduit, pour lequel la projection des individus respecte bien les distances et sont équivalentes à celle de l'espace vectoriel d'origine de dimension bien superieure. En général, c'est une technique utilisée pour analyser la similarité ou la dissimilarité entre les données. Ces dissimilarités sont représentées sous forme de distances dans un espace géométrique. Les résultats produits par MDS en se basant sur la distance euclidienne sont similaires aux résultats de la PCA, mais MDS permet d'utiliser des métriques autres que l'euclidienne. (8)(15)

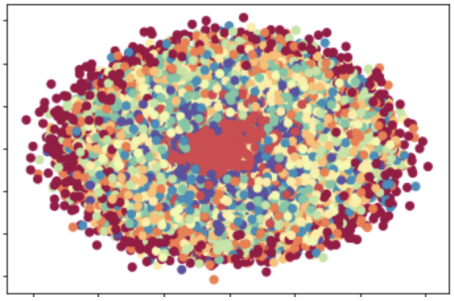


Figure 3: MDS sur le jeu de données MNIST

2.2.3 Isomap

Isomap peut être considéré comme une extension de MDS ou de K-PCA. Cet algorithme cherche un espace de dimension réduite qui conserve les distances géodésiques

entre tous les points au lieu de la distance euclidienne (10). La distance géodésique pouvant être considérée comme un kernel, similaire à l'utilisation de K-PCA.

Cette méthode se base sur les plus proches voisins de chacune des instances afin de conserver cette distance. Premièrement, Isomap génère un graphe de voisinage entre toutes les instances du jeu de données. Une fois ce graphe de voisinage généré, une table de dissimilarité est créée grâce à l'algorithme de distance de Djikstra. De cette manière, Isomap dispose de toute l'information nécéssaire afin de voir quels sont les points les plus proches des autres. Pour finir, on utilise MDS sur cette table de dissimilarité.

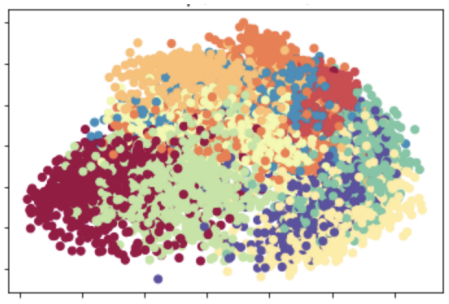


Figure 4: ISOMAP sur le jeu de données MNIST

La connectivité de chaque point de données dans le graphique de voisinage est définie grâce à ses k plus proches voisins selon la distance euclidienne dans l'espace à haute dimension. Si k est trop grand par rapport à la structure du maninfold ou si le bruit dans les données déplace légèrement les points, une seule erreur peut modifier de nombreuses entrées dans la matrice de distance géodésique, ce qui peut conduire à un encastrement très différent dans l'espace de dimension réduit. Inversement, si k est trop petit, le graphique de voisinage peut devenir trop clairsemé pour se rapprocher exactement des trajectoires géodésiques. (42)(43)

2.2.4 Locally Linear Embedding

Locally Linear Embedding(11) peut se définir comme une multitude de méthodes factorielles linéaires comparées et concaténées une-à-une dans des zones locales. On pourrait le comparer à une succession d'ACP lancées dans des zones locales du jeu de données puis comparées globalement afin de garder la meilleur qualité des données non-linéaires. De la même manière qu'ISOMAP, celui-ci se base sur la construction d'un graphe de similarité des points du jeu de données. LLE se base sur une conjecture géometrique: on suppose que tous les points du jeux de données

ainsi que ces voisins sont séparables ou disposés d'une manière linéaire dans une zone locale. Ainsi, de la même manière, chaque point et ses voisins dans l'espace de données réduit seront également répartis d'une manière linéaire: Chaque point est reconstruit en fonction de ses voisins. L'erreur de reconstruction étant mesurée par la fonction de coût:

$$\epsilon(w) = \sum_{i}^{N} |x_i – \Sigma_j w_{ij} x_j|^2$$

Les poids w_{ij} représentent la contribution du point j pour la reconstruction du point i.

Après avoir construit une représentation des données en fonction des voisins basée sur cette conjecture, l'étape finale de l'algorithme consiste à associer à chaque observation x_i un vecteur Y_i de dimension réduite, choisi en minimisant la fonction suivante:

$$\Theta(Y) = \sum_{i}^{N} |Y_i – \Sigma_j w_{ij} Y_j|^2$$

Une variante Hessienne de LLE existe (13): également connue sous le nom de **Hessian Eigenmapping**, il semblerait que LLE soit sensible au problème de la régularisation. Quand le nombre de voisins est superieur au nombre de dimensions, la matrice gardant en mémoire chacun des champs locaux linéaires n'a pas suffisament de rangs. Pour éviter ce problème de régularisation, on utilise une forme hessienne pour chaque groupe local afin de bien définir la structure linéaire.

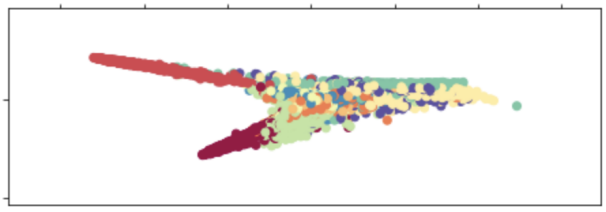


Figure 5: LLE sur le jeu de données MNIST

2.2.5 Local Tangent Space Alignment

LTSA(14) est similaire algorithmiquement parlant à LLE. Cependant, au lieu de se concentrer à préserver les distances locales comme le fait LLE, LTSA essaie de caracteriser chaque groupe local par son espace tangent. Il se base sur l'intuition que

si la réduction est correcte, tous les hyperplans tangents au jeu de données seront alignés.

LTSA recherche simultanément les coordonnées des représentations de données de faible dimension et les correspondances linéaires des points de données de faible dimension avec l'espace tangent local des données de haute dimension.

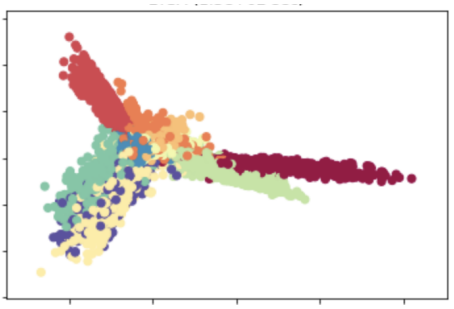


Figure 6: LTSA sur le jeu de données MNIST

2.2.6 Factorisation matricielle non négative

La **NMF**, ou factorisation matricielle non-négative permet d'approximer une matrice de données positives par le produit de deux matrices de dimensions inférieures. Par sa simplicité, cette méthode est devenue populaire et est utilisée à la fois dans un but de réduction de la dimension mais également dans la classification automatique en un nombre de classes k fixé par l'utilisateur. (19)(20)

Soit $\mathbf{X}$ une matrice de données positives (i.e. avec $x_{ij} > 0$, noté $\mathbf{X} > 0$), $\mathbf{X} \in R^{n*p}$, on recherche une matrice positive $\mathbf{W} \in R^{n*k}$ et une matrice positive $H \in R^{k*p}$ telles que :

$$\mathbf{X} \approx \mathbf{W} \times \mathbf{H}$$

L'approche principale de la factorisation matricielle non négative est donc d'estimer $\mathbf{W}$ et $\mathbf{H}$ en tant que minimum local. En pratique, le rang k est choisi de telle sorte que $k \ll min(n, p)$. L'objectif derrière ce choix est de résumer et de séparer l'information contenue dans une matrice de taille conséquente en k facteurs. Ce rang doit être assez petit pour réduire le bruit mais suffisamment élevé pour garder l'information nécessaire à la modélisation des données.

Il existe différents algorithmes de factorisation matricielle non négative: la majorité d'entre eux sont itératifs, il existe également des versions utilisant la descente de gradient. La factorisation matricielle non négative trouve des solutions différentes

en fonctions des conditions d'initialisation de l'algorithme mais aussi de l'algorithme lui même utilisé. (22)

La majorité des algorithmes de NMF résolvent le problème de manière itérative en construisant une suite de matrices (W_k, H_k). A chaque itération, l'algorithme essaie de réduire la fonction objectif jusqu'à obtenir un résultat probant. Ces méthodes se distinguent également de part les techniques d'optimisation utilisées pour calculer la suite des valeurs matricielles.

En ajoutant des contraintes de sparsité sur les matrices **W** et **H**, nous améliorons la qualité du clustering de la NMF (21), ce que nous recherchons dans le cadre de ce projet afin de comparer nos résultats de Deep-DR aux autres méthodes de l'état de l'art. La NMF étant également relativement sensible à l'initialisation du rang (W_0, H_0) de la suite convergente (44), nous avons initialisé W_0 et H_0 grâce à une Nndsvd: Basée sur le procédé de la décomposition en valeurs singulières, elle permet de diagonaliser une matrice normale par une base orthonormée de vecteurs propres. Cette méthode d'ensemencement convient parfaitement pour initialiser une NMF avec des facteurs sparses. (45)

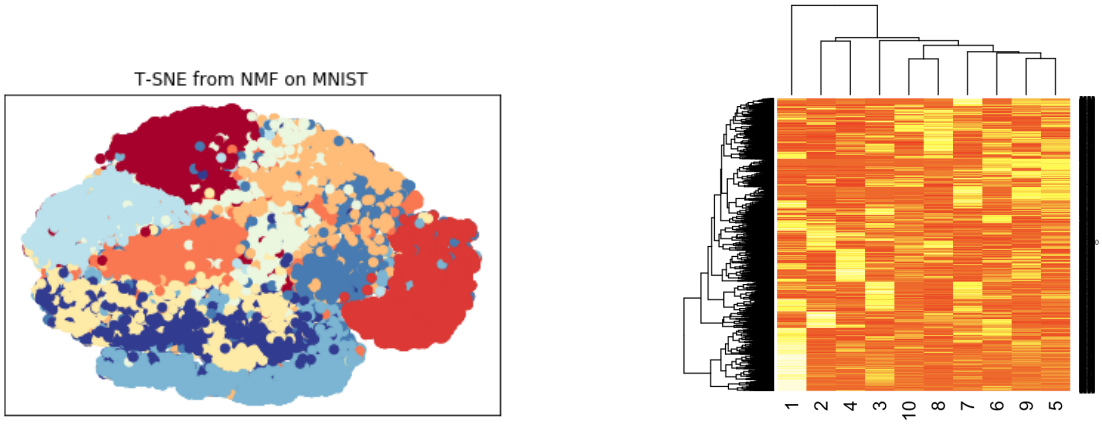


Figure 7: NMF sur le jeu de données MNIST

2.2.7 UMAP

UMAP définit une notion de similarité selon deux exemples autre que la distance euclidienne afin de refléter certaines propriétés locales des deux exemples comme leur densité. (23)

La similarité de UMAP w est bornée dans l'intervale $[0; 1]$ et est définie comme

$$w(\mathbf{x}_i, \mathbf{x}_j) \overset{\text{def}}{=} w_i(\mathbf{x}_i, \mathbf{x}_j) + w_j(\mathbf{x}_j, \mathbf{x}_i) - w_i(\mathbf{x}_i, \mathbf{x}_j) w_j(\mathbf{x}_j, \mathbf{x}_i).$$

avec la fonction $w_i(x_i, x_j)$ définie comme

$$w_i(\mathbf{x}_i, \mathbf{x}_j) \overset{\text{def}}{=} \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j) - \rho_i}{\sigma_i}\right),$$

Avec $d(x_i, x_j)$ la distance euclidienne entre deux points, ρ_i la distance de x_i avec son plus proche voisin et σ_i la distance de x_i avec son k-ième plus proche voisin.

En comparant la similarité w de chacun des exemples dans l'espace d'orgine et w' pour l'espace de dimension réduite, qui seront toutes deux bornées entre 0 et 1, on peut alors les considérer comme des distributions probabilistes et utiliser la cross-entropie:

$$C(w, w') = \sum_{i=1}^{N} \sum_{j=1}^{N} w(\mathbf{x}_i, \mathbf{x}_j) \ln\left(\frac{w(\mathbf{x}_i, \mathbf{x}_j)}{w'(\mathbf{x}'_i, \mathbf{x}'_j)}\right) + (1 - w(\mathbf{x}_i, \mathbf{x}_j)) \ln\left(\frac{1 - w(\mathbf{x}_i, \mathbf{x}_j)}{1 - w'(\mathbf{x}'_i, \mathbf{x}'_j)}\right)$$

avec x' le jeu donnée x dans l'espace de dimension réduit. Lorsque pour tous les couples (i, j), $w(x_i, x_j)$ est similaire à $w'(x'_i, x'_j)$ alors la cross-entropie est minimale: on obtient donc une representation adequate du jeu de données dans un espace réduit: les similarités dans l'espace d'origine et réduit sont équivalentes.

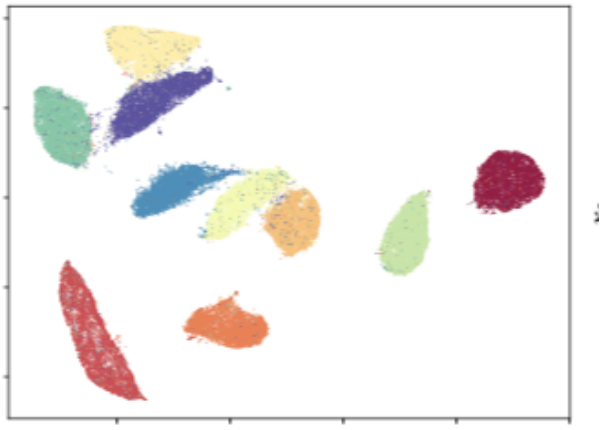


Figure 8: UMAP sur le jeu de données MNIST

2.2.8 t-SNE

t-SNE tente de trouver une configuration optimale selon un critère de théorie de l'information pour respecter les proximités entre points: deux points qui sont proches dans l'espace d'origine devront être proches dans l'espace de faible dimension. Une distribution de probabilité est également définie de la même manière pour l'espace de visualisation. L'algorithme t-SNE consiste à faire concorder les deux densités de probabilité, en minimisant la divergence de Kullback-Leibler entre les deux distributions par rapport à l'emplacement des points sur la carte. (17) (33)

$$KL(\mathbb{P}_r \| \mathbb{P}_g) = \int_{\mathcal{X}} P_r(x) \log \frac{P_r(x)}{P_g(x)} \, \mathrm{d}x$$

t-SNE va se concentrer sur la structure locale des données et va tendre vers une extraction de chacun des groupes locaux. Cette capacité à grouper les échantillons en fonction des structures locales voir topologiques, peut donner de très bon résultats, et ainsi déméler visuellement un dataset comprenant plusieurs structures inhérentes simultanément. Comme c'est le cas pour MNIST.

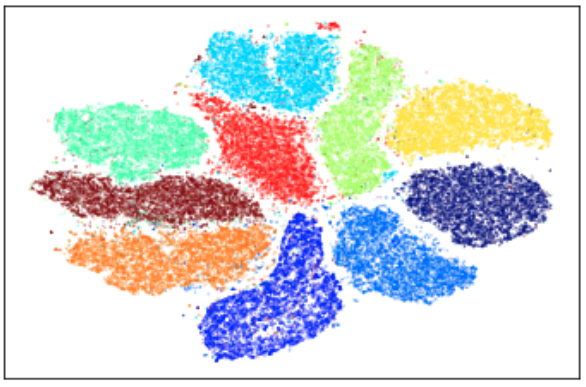


Figure 9: T-SNE sur le jeu de données MNIST

Il faut cependant raison garder et ne pas oublier que t-SNE n'est pas réellement un algorithme de réduction de dimension mais plutôt de visualisation (18): t-SNE n'effectue pas un mapping point par point de l'espace vectoriel original à l'espace de dimension réduite. De ce fait il est inutile d'appliquer des méthodes sur les points générés par t-SNE en esperant garder de l'information de l'espace initial comme pour les autres méthodes. t-SNE essaie simplement de placer les points dans un espace

réduit de telle sorte que les points ayant une dissimilarité élévée soient éloignés. La seule relation que garantit t-SNE contrairement aux autres algorithmes, est qu'un point p sera le même dans les deux espaces vectoriels. De ce fait on aura plusieurs de ces points "ancrés" qui seront directement liés dans les deux espaces. pour illustrer, imaginons un point p équidistant de a et b dans l'espace initial, l'algorithme le placera également à distance égale entre a et b dans l'espace réduit. Le problème est que t-SNE ne garantit pas que ces points voisins soient proches les uns des autres dans l'espace réduit. Il ne peut pas non plus garantir que 3 points alignés dans l'espace initial resteront également alignés dans l'espace réduit. Même en 3d, il existe une infinité de possibilités de plan de points p qui seront équidistants et qui auront a et b comme voisins les plus proches.

2.3 Autoencodeurs

Un **Autoencodeur** prend des données d'entrée avec une très haute dimensionnalité et va fonctionner à travers un réseau neuronal. Celui-ci va essayer de compresser les données pour les obtenir dans une représentation plus petite. (24)

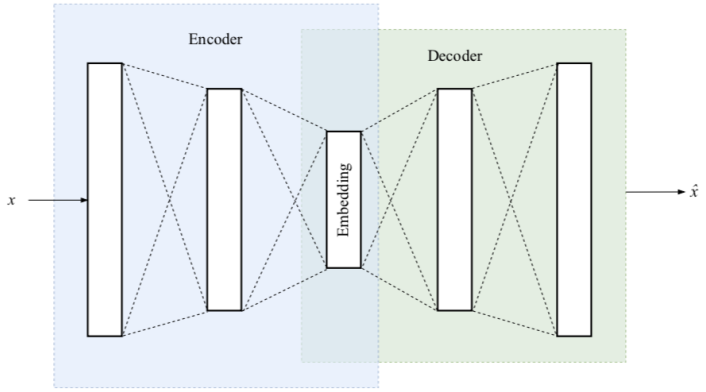


Figure 10: Un autoencodeur

Il se déroule en deux étapes:

- La première étape est ce que l'on peut considérer comme l'étape d'encodage: L'encodeur correspond à un amas de couches, il peut s'agir de couches complètement connectées ou convolutives qui vont prendre l'entrée et la comprimer jusqu'à obtenir une représentation de dimension plus réduite que l'entrée: c'est ce qu'on appelle le "goulot d'étranglement" ou bottleneck.

- À partir du goulot d'étranglement, la seconde partie de l'algorithme correspond à une tentative de reconstruction de la donnée initiale grâce à des couches entièrement connectées ou convolutives.

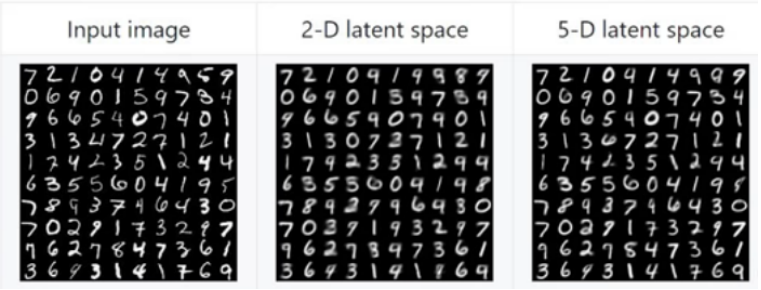


Figure 11: Images reconstruites en variant la taille du latent space d'un AE

Plus le nombre de dimensions dans l'espace latent est élevé, plus les reconstructions seront claires et robustes, ce qui nécéssite néanmoins plus d'informations dans le bottleneck. (25)(26)

La fonction de coût d'un autoencodeur consiste simplement à reconstruire le jeu de données initial et calculer la perte de reconstruction par rapport à l'entrée. Dans sa forme la plus basique, la solution optimale d'un autoencodeur est fortement liée à la PCA (34):

Afin de calculer les valeurs des couches cachées, on multiplie simplement les poids entre l'entrée et la couche et l'entrée, on a donc la formule suivante:

$$z = f(Wx)$$

De même sorte, pour obtenir la sortie, on multiplie les poids entre les couches cachées et la sortie par la valeurs des couches cachées, on obtient donc:

$$y = g(Vz)$$

Les fonction f et g sont libres, tant qu'elles sont non lineaires et qu'on peut les deriver pour la backpropagation.

On obtient donc la fonction suivante:

$$y = g(Vf(Wx)) = VWx$$

Nous permettant d'obtenir la fonction objectif:

$$J = \sum_{n=1}^{N} |x(n) – VWx(n)|^2$$

Ce qui correspond à la fonction objectif de la PCA.

Cependant, l'autoencodeur est bien plus flexible que la PCA: chaque couche dispose d'une fonction d'activition (Relu, tanh, sigmoide,..) permettant d'introduire la notion de non linearité, contraitement à la PCA qui ne peut representer que des transformations lineaires. Avec des contraintes de dimensionnalité et de parcimonie appropriées, les autoencodeurs peuvent donc apprendre des projections de données plus intéressantes que l'ACP ou d'autres techniques de base. De plus, ils peuvent être empilés, ce qui les rend encore plus puissants.

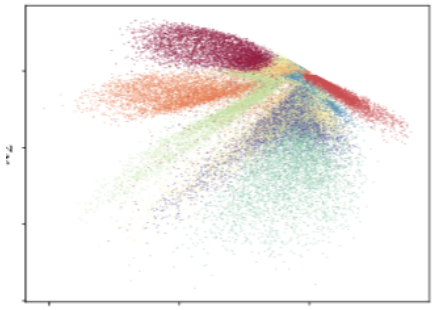


Figure 12: Autoencodeur sur le jeu de données MNIST

2.3.1 Variantes: Denoising

Différentes techniques existent pour empêcher un auto-encodeur d'apprendre la fonction identité afin d'améliorer sa capacité à apprendre des représentations plus riches: En obligeant le latent space à avoir un petit nombre de couches, on force le réseau a apprendre une representation intelligente des données. Cependant, il existe d'autres façons de forcer le réseau à apprendre une bonne representation des données: en ajoutant du bruit aléatoire à nos données d'entrées puis de récuperer l'image originale sans ce bruit. (28)(40)

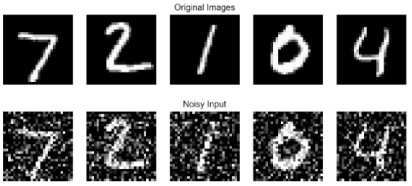


Figure 13: Denoising AE: images MNIST bruitées

La fonction de coût est calculée en comparant la sortie de l'autoencodeur aux données d'entrées non bruitées et non aux données corrompues. De cette facon l'autoencodeur ne peut pas simplement copier l'entrée et sa sortie car l'entrée est bruitée, on lui demande alors de soustraire le bruit et de comprendre les données sous jacentes: ce que l'on qualifie de denoising autoencodeur.

2.3.2 Variantes: VAE

Les autoencodeurs variationnels héritent de l'architecture des autoencodeurs, mais font des hypothèses fortes concernant la distribution des variables latentes. (29)

D'un point de vue probabiliste, un vae modélise des distributions de probabilité pour chaque donnée x et pour les variables latentes z. Le modèle définit alors une probabilité jointe entre les distributions des données et des variables latentes:

$$p(x, z) = p(x|z)p(z)$$

Le but du vae étant d'avoir une bonne estimation des variables latentes grâce aux données en calculant la probabilité à posteriori $p(z|x)$ grâce à la formule de Bayes.

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

L'inférence variationnelle approxime les valeurs à posteriori grâce à une distribution de probabilité: $q_\lambda(z|x)$.

Le paramètre λ répertorie la famille des distributions: si q était Gaussien, λ_{xi} équivaudrait à la moyenne et à l'écart-type de chaque point: $\lambda_{xi} = (\mu_{xi}, \sigma_{x_i}^2)$

Cette probabilité à posteriori est approximée grâce à la minimisation de la divergence de Kullback-Leibler, mesurant l'information perdue en utilisant une distribution pour approximer les données.

Une fois ces distributions trouvées, les poids et les biais du modèle correspondent simplement aux paramètres des distributions.

2.3.3 Variantes: Disentangled VAE

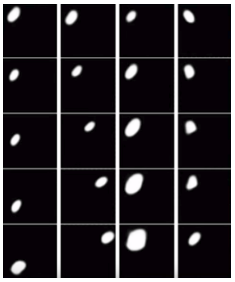
L'idée principale derrière cette variante consiste à garder les differents neurones dans l'espace latent décorrélés les uns des autres. Ces neurones vont tous essayer d'apprendre des informations différentes des variables d'entrée.

La différence avec un VAE dit classique consiste en l'ajout d'un hyper-paramètre dans la fonction de coût pondérant la divergence KL. (30)

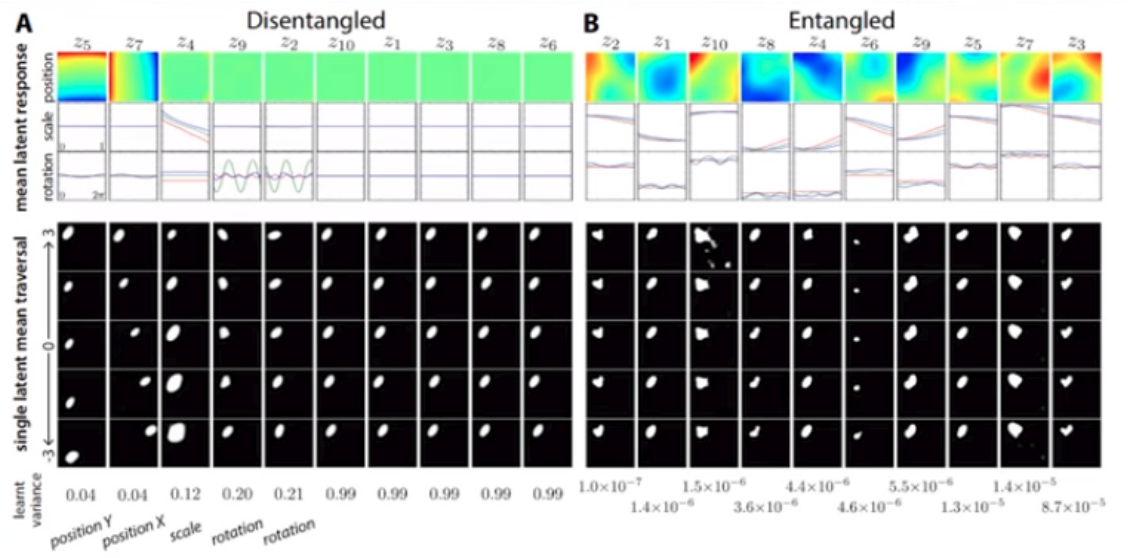
$$\mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z}, \beta) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\big)$$

Cette version utilisera alors les variables latentes que si elle n'en a réellement besoin, c'est à dire que lorsque cela apporte un bénéfice. Lorsqu'une variable supplémentaire n'apporte rien, le réseau ne l'utilisera tout simplement pas.

Si l'on illustre ce propos à l'aide d'un jeu de données simple géneré à partir de 4 latent facteurs : (position x, position y, taille de l'objet et la rotation de l'objet) :



en prenant un échantillon de cette distribution, on dispose de quatres valeurs, ce qui permet alors de génererer des images de cet exact latent space. L'idée est d'entrainer un Disentangled VAE en souhaitant que l'autoencodeur soit capable de reconstruire ces images en comprenant exactement le mapping de ces quatres latent variables pour encodeur l'information.



En utilisant un VAE classique,à droite, on remarque qu'il utilise plusieurs latent space pour encodeur les données d'entrée et ne trouve donc pas réelement ces quatres variables de base utilisées pour génerer les images.

À l'inverse, en utilisant un disantangled VAE, à gauche, on arrive à capturer quelque chose de bien plus proche de la réalité. Cela force l'autoencodeur à représenter l'information dans peu de latent variables. On remarque alors que le premier latent variable représente la position x, la deuxième la position y, le troisième la taille et le quatrième et cinquième représentent la rotation. Toutes les autres representations restant alors vacantes et non utilisées.

3 DeepDR: une approche simultanée

Récemment, un certain nombre de travaux ont étudié les stratégies de clustering qui combinent des algorithmes classiques de clustering et des méthodes de réduction de dimension ou d'apprentissage profond (31)(35)(36)(37). Ces approches suivent souvent une méthode séquentielle, où une représentation dans un espace de dimension réduit est apprise avant d'obtenir des groupes à l'aide d'une technique de clustering.

Pour remédier à l'inconvénient des méthodes traditionnelles de réduction de dimension, nous proposons une nouvelle fonction objectif régularisée. La méthode proposée s'inspirant des travaux de (31), peut être vue comme une procédure recherchant simultanément une nouvelle représentation des données contenant le maximum

d'informations (en utilisant un DAE), et un graphe de similarité caractérisant au mieux la proximité entre le points (en utilisant LLE). cette méthode consiste dans l'optimisation du problème suivant:

$$\min_{\theta_1,\theta_2,\mathbf{S}} \|\mathbf{X} - g_{\theta_2}(f_{\theta 1}(\mathbf{X}))\|^2 + \lambda \|f_{\theta 1}(\mathbf{X}) - \mathbf{S} f_{\theta 1}(\mathbf{X})\|^2$$

θ_1, θ_2 sont respectivement les paramètres des blocs encodeur et décodeur de l'autoencodeur et S est la matrice des poids caractérisant la proximité entre les poids calculée grâce à LLE.

le premier correspond à la fonction objectif d'un autoencodeur et le second terme correspond à la fonction objectif de la LLE. λ correspondant alors à l'importance que l'on accorde à la proximité entre les poids. Plus λ est élevé, plus on prendra en compte l'importance du deuxième terme dans la fonction de coût.

Cet algorithme s'appuie sur deux étapes de mise à jour selon le schéma décrit dans le pseudo-code ci-dessous:

Algorithm 1 : `Deep DR` algorithm

Input: data matrix $\mathbf{X}$;
repeat
(a) - Update $\mathbf{\Theta_1}$ and $\mathbf{\Theta_2}$ using DAE
(b) - Update $\mathbf{S}$ using LLE
until convergence
Output: $\mathbf{S}$: similarity matrix

4 Résultats experimentaux

L'algorithme Deep DR est évalué sur deux jeux de données: MNIST (38) et FashionMNIST (35):

Le jeu de données MNIST est souvent utilisé par la communauté machine learning pour comparer et valider des nouveaux algorithmes, c'est d'ailleurs souvent le premier jeu de données éssayé: si cela ne marche pas sur MNIST, il y'a peu de chance pour que cela soit un algorithme pertinent.

Cependant MNIST est considéré comme trop facile pour certaines tâches , Fashion-MNIST se positionne alors comme une alternative directe à MNIST pour l'évaluation

comparative des algorithmes de machine learning: ces deux datasets partagent la même taille d'image et la même structure d'entraintement et de test.

- MNIST: le jeu de données contient respectivement 60 000 et 10 000 images de taille 28x28 portant sur des chiffres manuscrits allant de 0 à 9. Les images sont en niveau de gris.
- FashionMNIST: le jeu de données contient respectivement 60 000 et 10 000 images de taille 28x28 contenant dix types d'articles différents de chez Zalando: T-shirt, pull, sac, robe...

Nous projetons dans un espace de deux dimension les résultats de DeepDR grâce à t-SNE et à UMAP, les hyper-paramètres ont étés selectionnés grâce aux recommandations d'utilisation de t-SNE (17) et aux recommandations d'utilisations de UMAP(23):

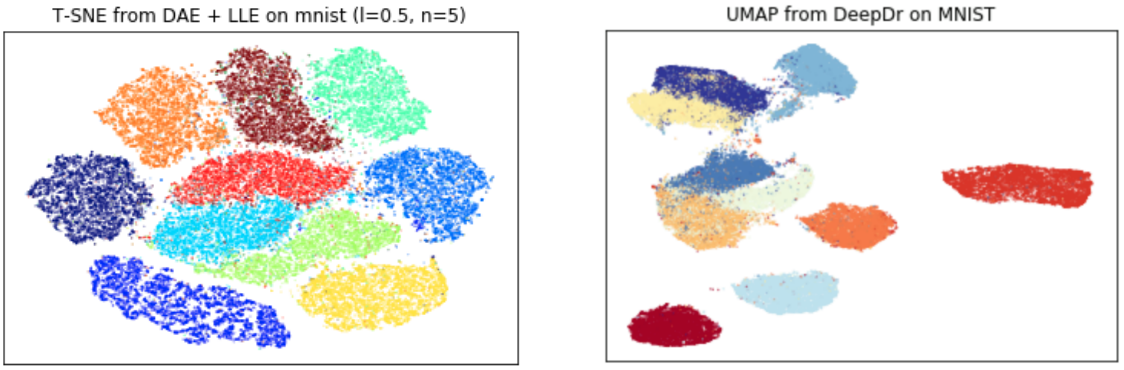


Figure 14: Visualisation T-sne et UMAP: DeepDr sur MNIST

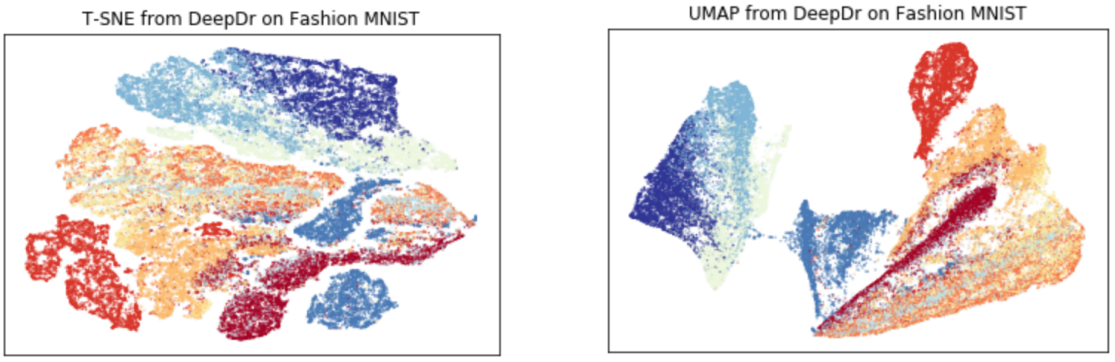


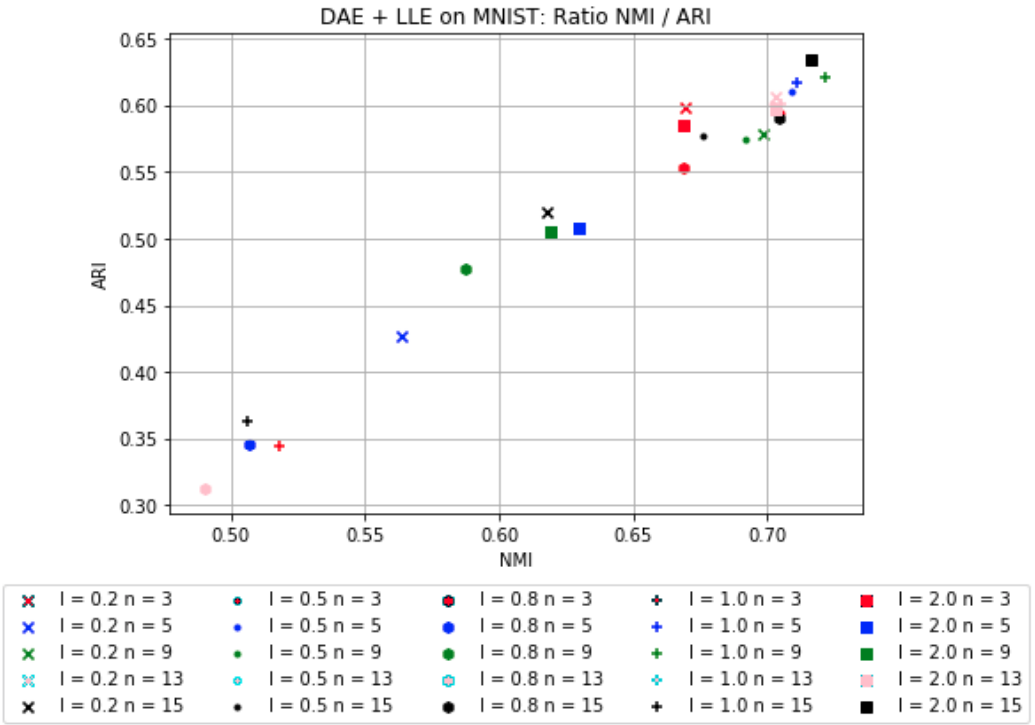
Figure 15: Visualisation T-sne et UMAP: DeepDr sur Fashion-MNIST

La qualité de représentation apprise par l'algorithme est assez bonne: on peut clairement observer certains des clusters séparés, les vraies classes des points s'accordent de manière convaincante avec la représentation générée: cela souligne la capacité de Deep DR à séparer les données selon leur classes.

4.1 Estimation de lambda et du nombre de voisins

Les performances de l'algorithme Deep DR étant directement corrélées à l'influence de λ dans sa fonction de coût et du nombre de voisins sélectionnés pour le calcul de la matrice S provenant de la LLE, nous avons dans un premier temps essayé une multitude de combinaisons possibles pour ces deux valeurs et avons ensuite évalué la qualité des modèles grâce à un clustering via les modèles de mélange gaussiens(39).

Le choix des mélanges gaussiens au détriment d'approches plus simples comme k-means nous offre une flexibilité afin d'approximer au mieux les données en utilisant des modèles appropriés(41).



L'évaluation des résultats d'un clustering n'est pas une tâche facile. Pour mieux évaluer la qualité de notre approche, nous retenons deux mesures largement utilisées

pour évaluer la qualité du clustering, à savoir l'information mutuelle normalisée et l'indice de rand ajusté.

Intuitivement, la NMI quantifie dans quelle mesure la partition estimée est informative sur la véritable partition, tandis que l'ARI mesure le degré d'accord entre une partition estimée et une partition de référence. Un ratio NMI/ARI élevé est préférable et symbolise alors un partitionnement adequat.

Ces résultats soulignent la forte influence des hyper-paramètres de Deep-Dr sur les valeurs NMI et ARI. Paralèllement, la structure de l'auto-encodeur utilisée joue également un rôle pré-pondérant sur ces deux indicateurs, nous nous sommes inspirés des travaux de (31) afin de conserver la structure la plus ad-hoc à notre situation.

4.2 Comparaison aux méthodes présentées

Une fois les hyperparamètres du modèle trouvés, nous avons souhaité comparer les performances de ce modèle aux autres algorithmes dits classiques de réduction de dimension: dans un souci d'équité, pour chaque méthode, le nombre de dimension conservé était de 10.

Les figures ci-dessous présentent les valeurs NMI et ARI pour les jeux de données MNIST et Fashion-MNIST. Les valeurs ARI et NMI sont données pour 10 clusterings réalisés via des mélanges de modèles gaussiens: ces algorithmes n'étant pas déterministes, réaliser plusieurs clusterings sur le même latent space nous permet d'observer la régularité des méthodes et donc leur robustesse et leur fiabilité:

L'algorithme Deep-Dr assure des résultats ARI et NMI élevés avec des écarts-types acceptables pour tous les ensembles de données réelles.

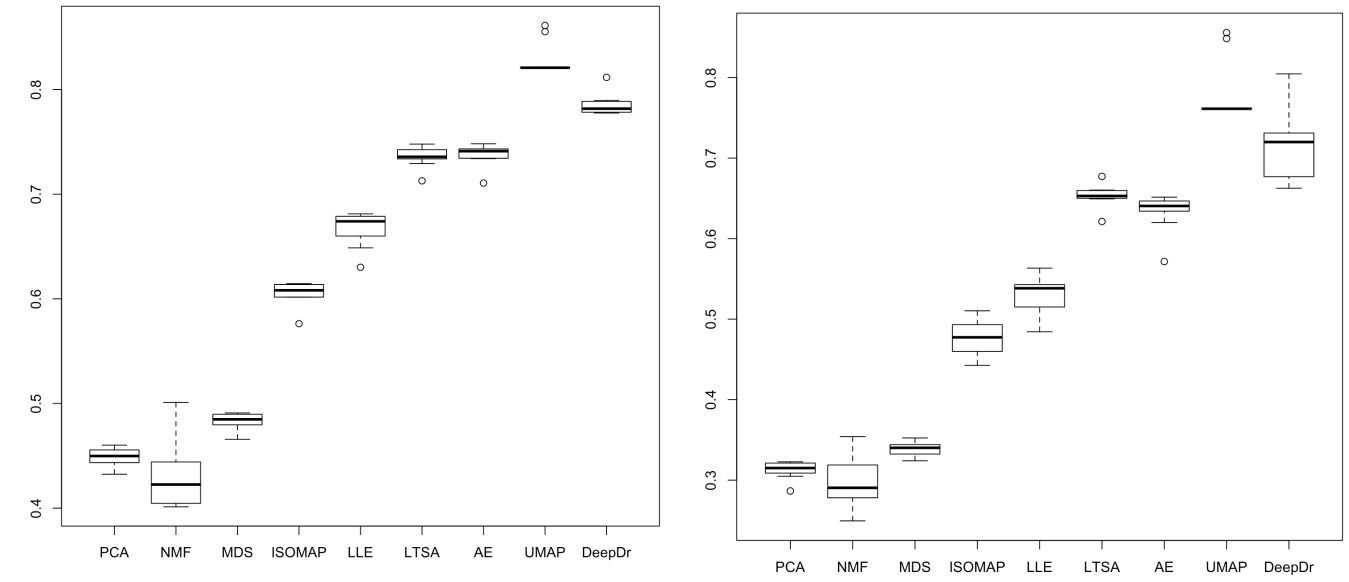


Figure 16: NMI et ARI des méthodes sur le jeu de données MNIST

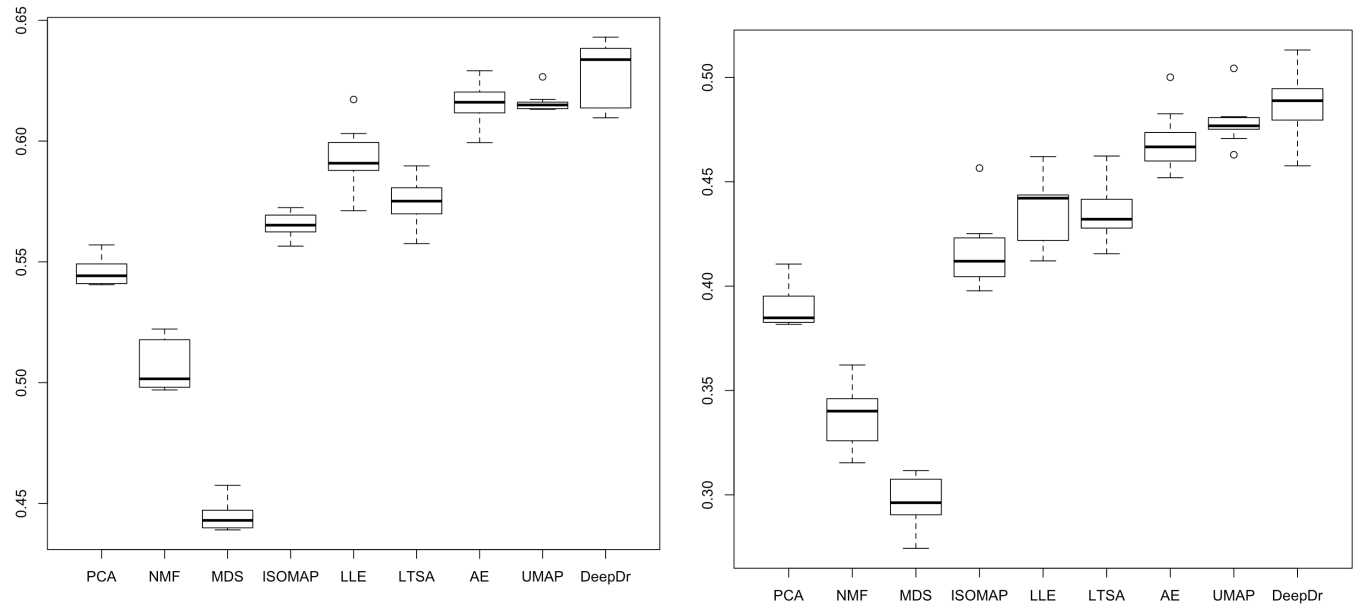


Figure 17: NMI et ARI des méthodes sur le jeu de données Fashion-MNIST

Les bons résultats obtenus par UMAP sur MNIST sont à prendre avec précaution: bon nombre des sujets de préoccupation soulevés par l'utilisation d'un clustering après t-SNE dans le chapitre 2 sont importants pour l'utilisation de UMAP dans le cadre d'un clustering. Le plus notable étant qu'au même titre que t-SNE, UMAP ne préserve pas complètement la densité.

UMAP peut ainsi créer des sous groupes ou des ruptures dans les groupes qui ne sont pas réelles, entrainant alors un clustering plus sensible que ce qui est réellement présent dans les données. Malgrés ces préoccupations, il existe cependant des raisons valables d'utiliser UMAP en tant qu'algorithme de réduction de dimension avant clustering: comme pour toute approche de clustering, on voudra dans l'absolu faire de l'exploration et de l'évaluation des clusters proposés pour essayer de les valider si possible: en connaissant à priori le nombre de classes du jeu de données. Réaliser un clustering correct via UMAP nécéssite un fine-tuning des hyper-paramètres (23) et donc une connaissance partielle du jeu de données initial contrairement à notre approche. Ce fine-tuning permettant de conserver la même structure globale, mais regroupant d'avantage les points de chaque classe les uns aux autres, et par conséquent, faisant apparaitre des espaces plus larges entre les groupes facilitant le clustering.

Le tableau sous-jacent, comporte les p-values des tests de Student de comparaison de moyenne nous permettant de confirmer la robustesse de Deep-DR comparé aux autres méthodes de réduction de dimension.

DeepDr p-values vs	MNIST		FashionMNIST	
	NMI	ARI	NMI	ARI
PCA	3.033e-15	1.643e-10	1.4e-09	4.244e-08
NMF	1.058e-11	9.911e-10	2.036e-10	3.566e-09
MDS	2.683e-15	1.99e-10	4.828e-14	1.755e-10
ISOMAP	4.348e-12	2.415e-09	5.742e-08	3.529e-07
LLE	6.519e-09	6.585e-08	1.994e-05	0.0001222
LTSA	7.78e-08	0.0004858	2.645e-07	3.308e-06
Autoencodeur	4.207e-07	0.0001328	0.02311	0.02443
UMAP	0.8766	0.7849	0.004949	0.03241

Table 1: DeepDR vs méthodes classiques: p-values obtenues grâce à un test de Student

DeepDr semble donc bien être plus robuste qu'une approche séquentielle de clustering après application d'algorithmes dit classiques de réduction de dimension ou d'autoencodeurs. Cela nous permet de confirmer sa compétivité comparé aux méthodes présentées sur des jeux de données classiques de benchmarking.

5 Conclusion

Ce rapport introduit DeepDr, un algorithme combinant la réduction de dimension et l'apprentissage profond pour apprendre une représentation plus précise des données. Après avoir réalisé un bref état de l'art de la réduction de dimension, un benchmarking de DeepDr sur deux jeux de données aura été réalisé afin de le confronter à certaines méthodes de l'état de l'art.

La précision des auto-encodeurs dans la détermination de la structure latente des données a été améliorée grâce à l'ajout d'une pondération dans la fonction de coût pour la tâche de réduction de la dimensionnalité et donc pour le clustering en exploitant le potentiel des modèles de mélanges. Par conséquent, DeepDR ne vise pas seulement à maximiser la variance des données, mais aussi à découvrir la structure potentielle des clusters, ce qui en fait un algorithme à objectif double, robuste et compétitif.

6 Travaux Futurs

Un grand nombre d'hyperparamètres, de tests et d'expériences ont été abandonnés ou laissés à l'avenir par manque de temps. Les travaux futurs portent principalement sur une analyse plus poussée de l'optimisation de l'algorithme DeepDr, nous avons également quelques propositions pour essayer différentes méthodes ou simplement réaliser des tests sur d'autres jeux de données par curiosité.

Certaines de ces idées ont été partiellement essayées lors de notre développement du chapitre 4, l'absence de résultats concluants associée à un trop peu de temps pour tester ces idées ne nous aura pas permis d'en tirer des conclusions satisfaisantes. Néanmoins les idées suivantes pourraient être éssayées ou améliorées:

- Combiner les Denoising auto-encodeurs avec Deep-DR: les résultats obtenus n'étaient statistiquement pas foncièrement meilleurs que les résultats de DeepDR, nous n'avons pas souhaité alourdir le rapport avec ces résultats équivalents: il serait intéressant de pousser l'étude pour améliorer la performance absolue de l'algorithme.
- Il serait également intéressant de combiner les auto-encodeurs variationnels avec Deep-DR: notre implémentation de $\beta - vae$ combinée à DeepDr semble attribuer la même représentation à différents points dans l'espace latent.
- Les deux jeux de données testés étant des images, nous avons envisagé d'utiliser des couches convolutives au lieu de couches denses pour l'autoencodeur, les

résultats n'étaient pour l'instant pas significatifs.

- Deep-DR n'a pas été testé sur des jeux de données déséquilibrés: il serait désirable de tester Deep-DR sur ce type de dataset: USPS par exemple.
- Changer l'architecture du modèle: bien que des travaux aient été réalisés sur l'optimisation des hyperparamètres λ et du nombre de voisins, la structure de l'autoencodeur est néanmoins restée invariante. L'optimisation des résultats de DeepDR pourrait également être améliorée en proposant d'autres type d'architecture.

References

[1] Lindsay I Smith *"A tutorial on Principal Components Analysis"* .

[2] Lei Zhanga, Weisheng Dongab, David Zhanga Guangmin gShib. *"Two-stage image denoising by principal component analysis with local pixel grouping"* .

[3] Y. Murali Mohan Babu, Dr. M.V. Subramanyam, Dr. M.N. Giri Prasad *"PCA based image denoising"* .

[4] Lei Zhang, Weisheng Dong, David Zhang, Guangming Shi. *"Two-stage image denoising by principal component analysis with local pixel grouping"* .

[5] Bernhard Schölkopf, Alexander Smola, Klaus-Robert Müller. *"Kernel principal component analysis"* .

[6] Sebastian Mika, Bernhard Scholkopf, Alex Smola Klaus-Robert Muller, Matthias Scholz, Gunnar Riitsch. *"Kernel pca and De-Noising in Feature Spaces"* .

[7] JB Kruskal. *"multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis."*

[8] Michael C. Hout, Megan H. Papesh, Stephen D. Goldinger *"Multidimensional scaling"* .

[9] Yoshua Bengio, Jean-François Paiement, Pascal Vincent, Olivier Delalleau, Nicolas Le Roux, Marie Ouimet. *"Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering"* .

[10] Tenenbaum, J.B.; De Silva, V.; & Langford, J.C. *"A global geometric framework for nonlinear dimensionality reduction"*.

[11] Roweis, S. & Saul, L. *"Nonlinear dimensionality reduction by locally linear embedding"* .

[12] Zhang, Z. & Wang, J. *"MLLE: Modified Locally Linear Embedding Using Multiple Weights"* .

[13] Donoho, D. & Grimes, C. *"Hessian Eigenmaps: Locally linear embedding techniques for high-dimensional data"*.

[14] Zhang, Z. & Zha, H. *"Principal manifolds and nonlinear dimensionality reduction via tangent space alignment"*.

[15] Kruskal, J. Psychometrika *"Nonmetric multidimensional scaling: a numerical method"* .

[16] Borg, I.; Groenen P. Springer *"Modern Multidimensional Scaling - Theory and Applications"*.

[17] van der Maaten, L.J.P.; Hinton, G *"Visualizing High-Dimensional Data Using t-SNE"*.

[18] van der Maaten, L.J.P. *"t-Distributed Stochastic Neighbor Embedding"* .

[19] Ali Caner Turkmen, Department of Computer Engineering *A Review of Non-negative Matrix Factorization Methods for Clustering(2015).*

[20] Tao Li, Chris Ding, University of Texas at Arlington *Non-negative Matrix Factorizations for Clustering: A Survey.*

[21] P. O. Hoyer *Non-negative matrix factorization with sparseness constraints.(2004).*

[22] Renaud Gaujoux *An introduction to NMF package(2018).*

[23] Leland McInnes, John Healy, James Melville *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction(2018).*

[24] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, *"Learning internal representations by error propagation."* , *Parallel Distributed Processing. Vol 1: Foundations. MIT Press, Cambridge, MA, 1986.*.

[25] Guyon, G. Dror, V. Lemaire, G. Taylor and D. Silver *Autoencoders, Unsupervised Learning, and Deep Architectures.*

[26] G. E. Hinton and R. R. Salakhutdinov. *Reducing the Dimensionality of Data with Neural Networks.*

[27] M Belkin, P Niyogi. *.Laplacian eigenmaps for dimensionality reduction and data representation Neural computation, 2003 MITPress.*

[28] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, Pierre-Antoine Manzagol. *Extracting and Composing Robust Features with Denoising Autoencoders.*

[29] Diederik P Kingma, Max Welling. *Auto-Encoding Variational Bayes.*

[30] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, Alexander Lerchner. *Understanding disentangling in beta-VAE.*

[31] Séverine Affeld, Lazhar Labiod, Mohamed Nadif. *Spectral Clustering via Ensemble Deep Autoencoder Learning (SC-EDAE).*

[32] Laurens van der Maaten, Eric Postma, Jaap van den Herik *Dimensionality Reduction: A Comparative Review.*

[33] Laurens van der Maaten, Geoffrey Hinton *Visualizing Data using t-SNE.*

[34] Elad Plaut *From Principal Subspaces to Principal Components with Linear Autoencoders.*

[35] Han Xiao, Kashif Rasul, Roland Vollgraf *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.*

[36] Xifeng Guo1, Xinwang Liu1, En Zhu1, and Jianping Yin. *Auto-encoder Based Data Clustering.*

[37] Chris Ding, Xiaofeng He. *K-means Clustering via Principal Component Analysis.*

[38] Y. LeCun, L. Bottou, Y. Bengio and P. Haffne. *Gradient-Based Learning Applied to Document Recognition.*

[39] Chris Fraley, Adrian E. Raftery, Luca Scrucca, Thomas Brendan Murphy, Michael Fop *Package 'mclust':Gaussian Mixture Modelling for Model-Based Clustering, Classification, and Density Estimation.*

[40] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent *Generalized Denoising Auto-Encoders as Generative Models.*

[41] Milad Leyli-Abadi, Lazhar Labiod, and Mohamed Nadif. *Denoising Autoencoder as an effective dimensionality reduction and clustering of text data.*

[42] Li Jing and Chao Shao *Selection of the Suitable Parameter Value for ISOMAP.*

[43] Rasa Karbauskaite, Olga Kurasova, Gintautas Dzemyda *Selection of the number of neighbours of each data point for the locally linear embedding algorithm.*

[44] Stefan Wild *Seeding Non-Negative Matrix Factorizations with the Spherical K-Means Clustering.*

[45] C. Boutsidis, E. Gallopoulos *SVD based initialization: A head start for nonnegative matrix factorization.*