Rapport de Projet : Modèles de mélanges

SOREL Tony 21301072

I. Objectif

Dans le cadre de ce projet il nous a été amené à mettre en pratique les différentes méthodes de clustering sur 5 jeux de données de grandes tailles et dimension très utilisé en Deep Learning. Chaque variable correspond à un pixel. Ce grand nombre de dimension rends ces jeux de données difficile a visualiser.
L'objectif de ce projet est découvrir des méthodes pour partitionner correctement ce genre de jeu de données.

II. Ensembles de données

Jeu de données	Échantillon	Variables	Nombre de classes
JAFFE	213	676	10
MNIST5	3495	784	10
MFEA	2000	240	10
USPS	9298	265	10
OPTIDIGITS	5620	64	10

Table 1: Caractéristiques des différents jeux de données

1. Jaffe

Le jeu de données Jaffe ou Japanese Female Facial Expression est un jeu d'images contenant 213 photo de 7 expression faciales de 10 femmes japonaises prise par le departement psychologie de L'université de Kyushu. Chaque image est composé de 676 pixels et chaque image est lebelisé en fonction de la personne photographiée (les labels vont de 1 à 10)

2. MNIST5,MFEA,USPS,OPTIDIGITS

Ce sont des jeux de données d'images représentant des chiffres écrit à la mains où chaque label indique à quel chiffre l'image représente.

3. Visualisation des jeux de données via l'ACP

Nous avons appliqué une ACP afin de visualiser chaque dataset en deux dimension: Les figures ci-dessous comportent les résultats obtenus pour chaque data set.

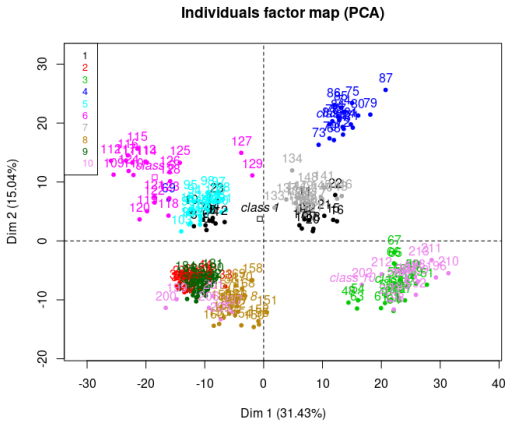

Figure 1: ACP sur JAFFE

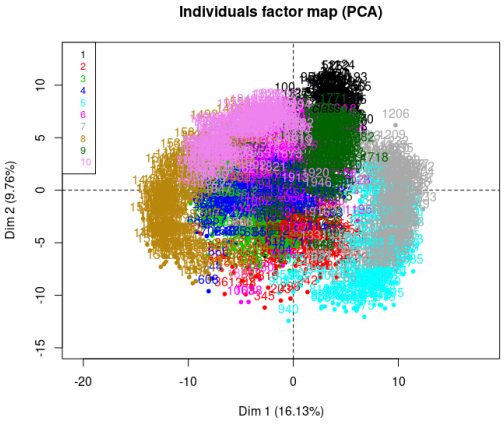

Figure 2: ACP sur mfeat

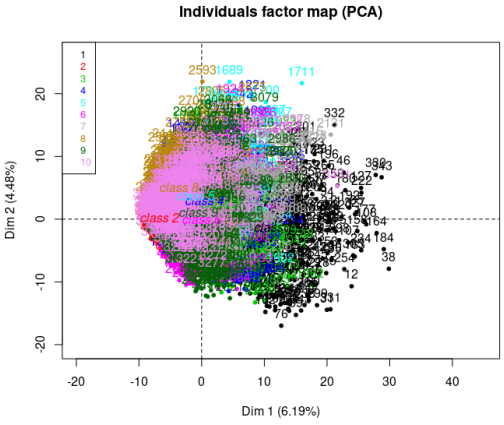

Figure 3: ACP sur mnist

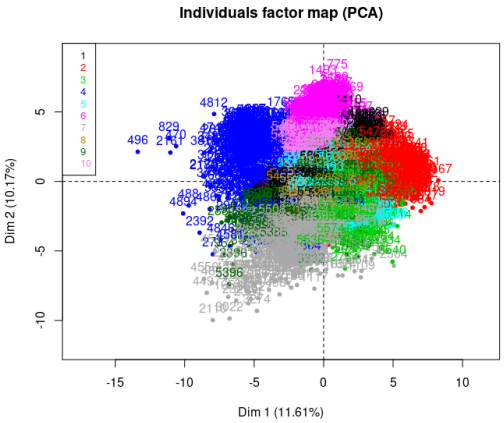

Figure 4: ACP sur Optdigits

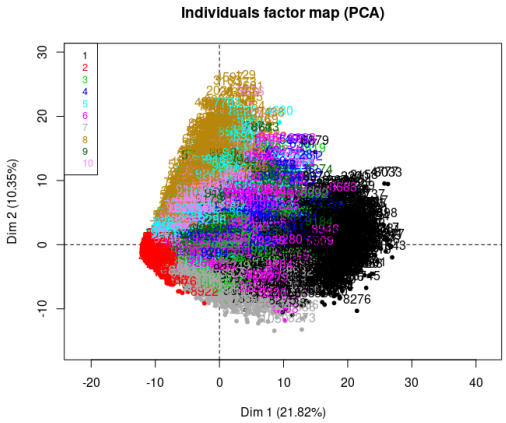

Figure 5: ACP sur USPS

D'après ces graphes nous remarquons que se Jaffe est le seul jeu de donnée auquel l'ACP à réussi a crée une représentation où les classes sont bien distinguées mais pour les 4 autres il est quasiment impossible de distinguer leur classes et c'est probablement dû au fait que l'ACP n'est pas adapté au jeux à très grande dimension où chaque individus qui ont énormément de variables nulles comme les images.

III. Évaluation des différentes méthodes de partitionnement

Nous avons ensuite appliquer différentes méthodes de partitionnement sur ces 5 jeux de données et utiliser la mesure "NMI" pour toutes les évaluer.

1. Méthodes classiques

1.1 NbClust

Dans un premier temps nous avons du utiliser les différentes présentes dans NbClust, cette méthode permet aussi de déterminer le nombre de clusters mais les résultats sont plutôt décevants, dans le jeu Jaffe le nombre de cluster fluctue énormément en fonction de la méthode utilisé (entre 13 et 43) et pour les 4 autres ils ont tendance a déterminer qu'il y a 2 ou 3 cluster.

	NMI				
	Kmeans	***ward***	***average***	***single***	***complete***
JAFFE	0.884	0.883	0.608	0.39	0.66
MNIST5	0.486	0.594	0.129	0.0025	0.289
MFEA	0.755	0.796	0.5846	0.005	0.469
USPS	0.613	0.772	0.08	0.0009	0.3532
OPTDIGITS	0.694	0.833	0.695	0.001	0.5483

Table 2: Précision (NMI) pour chaque méthode de partitionnement

Dans ce tableau nous pouvons apercevoir que :

- Les différentes méthodes obtiennent leur meilleur résultat sur JAFFE
- La CAH single n'est pas adapté au jeu de données de type images
- La CAH ward les Kmeans sont les meilleurs technique de NbClust pour ce genre de données

1.2 HCPC

Nous avons ensuite utilisé la librairie HCPC pour lancer des CAH depuis le résultat d'une analyse factorielle comme l'ACP. Dans les 5 cas les méthodes de HCPC avaient tendance à détecter 3 ou 4 cluster ce qui est loin des 10 classes annoncées.

	Précision			
	ward	***complete***	***single***	***average***
JAFFE	0.744	0.734	0.663	0.669
MNIST5	0.2	0.194	0.196	0.203
MFEA	0.480	0.487	0.48	0.49
USPS	0.373	0.376	0.374	0.373
OPTDIGITS	0.459	0.459	0.459	0.459

Table 3: Précision pour chaque méthode de clustering

Comme HCPC se base sur le résultat d'une ACP ça signifie que ces performances sont très lié à celle de l'ACP. En effet ont remarque dans le tableau ci dessus que seul HCPC a uniquement obtenu des résultat convainquant avec JAFFE qui est le seul jeu où l'ACP est performant.

2. Modèles de mélanges

Maintenant nous allons voir comment les jeux de données réagissent au approches basé sur les modèles de mélanges. Vu que les variables de ces dataset sont continues nous nous dirigerons vers les modèles Gaussien de type "EM" via les deux librairie Rmixmod et Mclust. Après plusieurs essais nous avons remarqué que la complexité en temps et espace de Rmixmod et dans une moins mesure de Mclust étaient très liées au nombre de variables et qu'il n'était pas possible de lancer directement Rmixmod à partir d'un jeu non réduit. Dans ce cas nous avons du avoir recours a plusieurs méthode de réduction qui nous on amené à différents résultat.

2.1 À partir d'une ACP

Dans un premier temps nous avons lancé une ACP sur chaque jeu et utiliser les 10 premières composantes pour lancer les deux méthodes, ce qui nous a permit de trouver une estimations des modèles optimaux pour chaque jeu de données:

	JAFFE	MNIST5	MFEA	USPS	OPTDGITS
Modèle	EVE	VEV	VEV	VVI	EEV

Table 4: Modèles optimaux d'après Rmixmod et Mclust

	Précision	
	Rmixmod	***Mclust***
JAFFE	0.90	0.95
MNIST5	0.6	0.34
MFEA	0.742	0.760
USPS	0.566	0.428
OPTDIGITS	0.825	0.88

Table 5: NMI des deux méthode de clustering basé sur le modèle Gaussien

Comme nous pouvons le voir ci-dessus ces résultat sont certes lié au performances de l'ACP mais elle permet tout de même d'obtenir de meilleur résultat que HCPC dans le cas où l'ACP donne une mauvaise représentation du dataset.

2.2 À partir d'un auto-encodeur

Ensuite nous avons décider d'utiliser une approche Deep Learning pour réduire les dimensions en utilisant des auto-encodeurs. Étant donné que l'ACP peut être considéré est un cas particulier d'un auto-encodeur nous allons voir si il est possible de crée un auto-encodeur plus performant que l'ACP.

Pour cela nous avons passer sous python avec la librairie Keras pour crée des auto-encodeur permettant de réduire à 2 le nombre de dimension de chaque jeu:

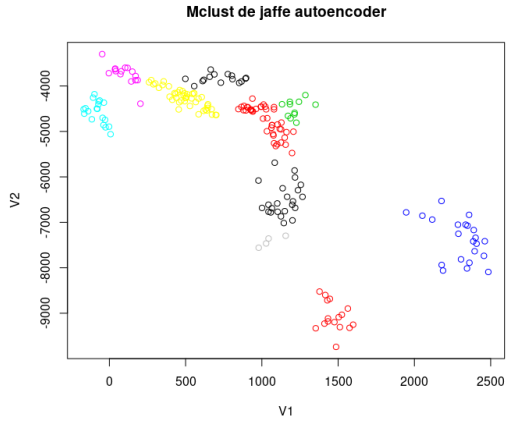

Figure 6: Résultat de l'auto-encodeur sur JAFFE

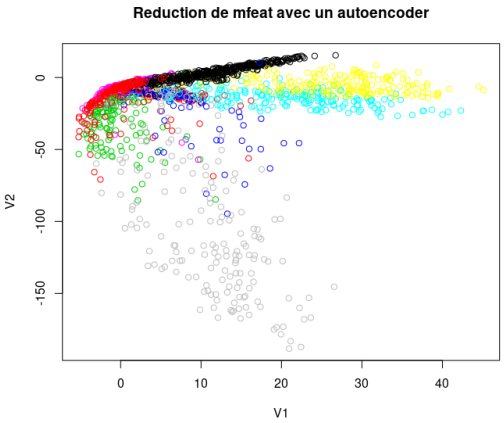

Figure 7: Résultat de l'auto-encodeur sur mfeat

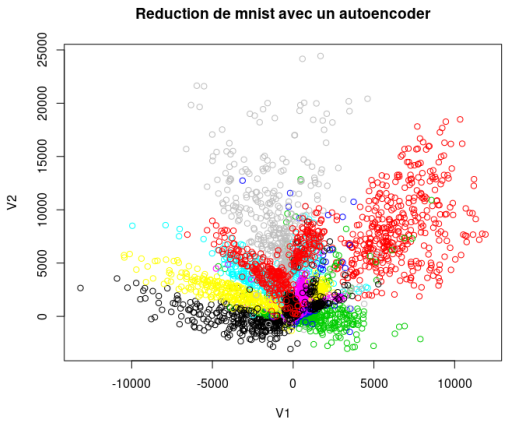

Figure 8: Résultat de l'auto-encodeur sur mnist

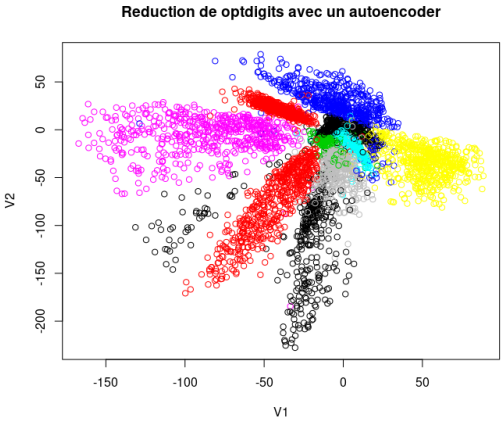

Figure 9: Résultat de l'auto-encodeur sur Optdigits

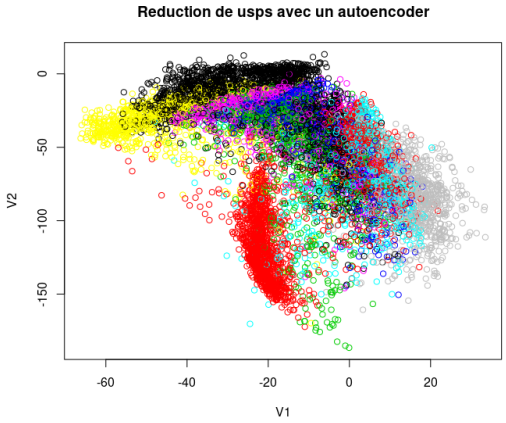

Figure 10: Résultat de l'auto-encodeur sur USPS

Comme nous pouvons le voir ce n'est pas parfait mais les représentation des 4 dernier jeux sont plus claire qu'avec une ACP et on peut mieux distinguer les classes.
Nous avons ensuite utiliser ces résultat pour lancer Mclust , sachant que contrairement au précédent test avec l'ACP nous somme ici sur 2 dimension et donc nous pouvons espérer des résultat aussi bon qu'avec 10 dimension:

	JAFFE	MNIST5	MFEA	USPS	OPTDIGITS
NMI	0.807	0.45	0.48	0.44	0.677

Table 6: NMI de Mclust réalisé a partir d'un auto-encodeur

Mais ce tableau montre que pour USPS et MNIST5 on arrive a obtenir des résultat supérieur au précédents test avec Mclust.

2.3 À partir d'une t-SNE

Cette fois ci nous nous somme intéressés à la t-SNE qui est une technique non-linéaire de réduction de dimension basé sur l'interprétation probabilistes des proximité entre individus. Pour utiliser cette méthodes nous avons installer la librairie "Rtsne" qui est un wrapper R vers une implémentassions C++ de l'algorithme qui bien plus rapide que celle de la librairie "tsne". Comme avec les auto-encodeur nous avons décider de ne garder que 2 dimension pour mesurer les performance de l'algorithme:

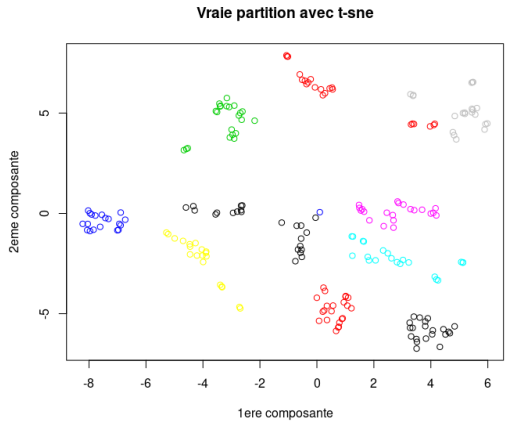

Figure 11: Résultat de la t-sne sur JAFFE

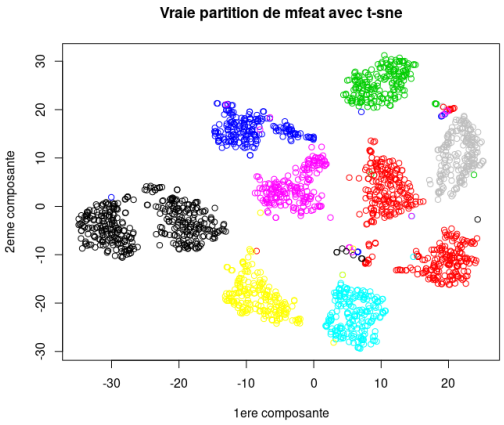

Figure 12: Résultat de la t-sne sur mfeat

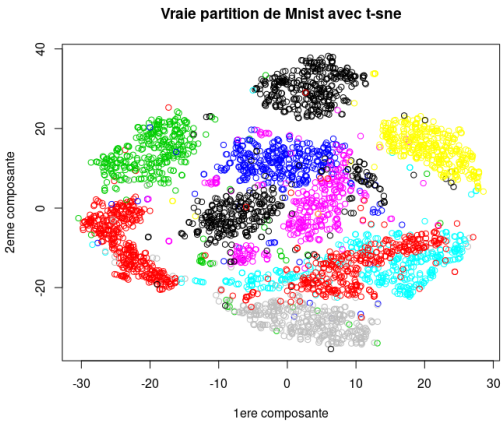

Figure 13: Résultat de la t-sne sur mnist

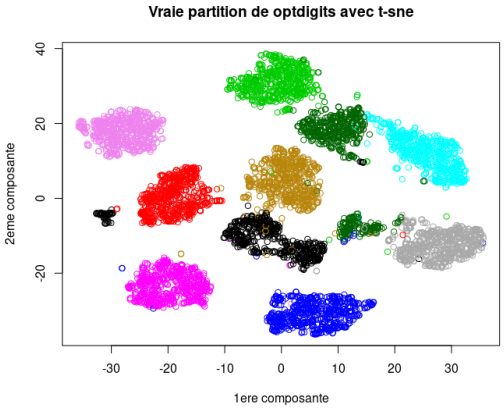

Figure 14: Résultat de la t-sne sur Optdigits

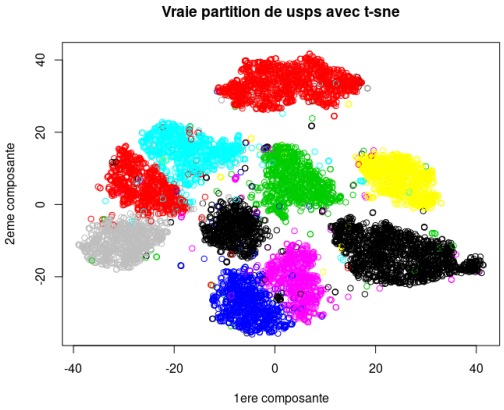

Figure 15: Résultat de la t-sne sur USPS

Cette fois ci les classes des 4 derniers jeux de données sont bel et bien distinguable contrairement à l'ACP/Auto-encodeur ce qui montre que la t-SNE est une technique très adapté aux données d'images:

	JAFFE	MNIST5	MFEA	USPS	OPTDIGITS
NMI	0.9	0.72	0.925	0.81	0.92

Table 7

Pour quasiment tout les jeux de données on obtient de meilleur résultat que dans les deux précédents exemple ce qui confirme que la t-SNE conserve mieux en deux dimension les caractéristique de ce genre de dataset que l'ACP en 10.

IV. Conclusion

Lors ce projet, nous avons du étudier 5 jeux de données à grande dimension et utiliser plusieurs méthodes pour en venir à la conclusion que la meilleur méthode pour partitionner la individus d'appliquer un Mclust après une t-SNE. Les modelés de mélanges peuvent être de bonnes approches pour partitionner des jeux de données mais on peut se retrouver dans des situation comme celle ci où une réduction de dimension doit nécessairement être effectué avant de les utilisés ce qui les rend dépendant des performances de la technique utilisée.