Modèles de mélanges appliqués aux données d'images usuelles

Mohamed BEN HAMDOUNE, Lucas ISCOVICI

Paris, France

Université de Paris Descartes

Abstract

Les modèles de mélanges sont régulièrement utilisés pour extraire des caractéristiques à partir de données de parole, mais également dans la détection d'objet à partir d'images. En déduisant des paramètres de la distribution des données, ils permettant de prédire l'emplacement des objets à chaque image d'une séquence vidéo. Le modèles de mélange gaussien est un modèle probabiliste que l'on utilisera, permettant de représenter des sous-populations normalement distribuées au sein d'une population globale. Les modèles de mélange en général n'exigent pas de savoir à quelle sous-population appartient un point de données, ce qui permet au modèle d'apprendre automatiquement les sous-populations. Comme l'affectation des sous-populations n'est pas connue, il s'agit d'une forme d'apprentissage non supervisé.

Keywords: Visualisation, Classification, Modèles de mélange, Autoencodeur, Algorithme espérance-maximisation

Contents

1 Introduction **3**

2 Analyse exploratoire des données **4**
2.1 JAFFE 4
2.2 MNIST5 5
2.3 OPTIDIGITS 5
2.4 USPS 5
2.5 MFEA 6

3 Classification **11**
3.1 Classification Hierarchique Ascendante 11
3.2 Partitionnement 12
3.3 Résultats 13
3.4 Classification avec ou sans analyse en composantes principales 14

4 Classification avec un modèle de mélange gaussian **14**
4.1 L'approche du maximum de vraisemblance : l'algorithme EM . 15
4.2 Évaluation des résultats 15
4.2.1 Données initiales 15
4.2.2 Données ayant subi une ACP 17
4.3 MclustDR 17
4.4 Comparatif avec le vrai nombre de classes avec GMM 18

5 Visualisation avec t-SNE **19**

6 Autoencodeur **21**

7 Modèle de mélange et analyse factorielle simultanés **23**

8 Conclusion **24**

1. Introduction

La classification des images est l'un des défis majeurs du traitement d'images et de la vision par ordinateur. Cependant, l'application de modèles de mélanges appliqué à la segmentation présente quelques difficultés. Pour le modèle statistique de mélange classique, chaque pixel doit être associé à exactement une classe. Cette hypothèse peut ne pas être réaliste. Ainsi, plusieurs méthodes ont été proposées pour contourner ce problème (comme la classification flou). Cette approche donne des résultats satisfaisants dans de nombreux cas, mais dans la plupart des cas l'hypothèse d'un seul gaussien limite généralement la précision du modèle.
Nous utiliserons, l'algorithme EM pour estimer les paramètres des mélanges gaussiens.
Ensuite le modèle de mélange gaussien est un modèle probabiliste flexible et un puissant outil de modélisation. Il peut être utilisé pour fournir un modèle basé sur le regroupement dans le domaine de la reconnaissance de formes. Cependant, l'application présente quelques difficultés puisqu'il est sensible au bruit.

2. Analyse exploratoire des données

Nous commencerons l'analyse exploratoire par des techniques de visualisations et ensuite nous ferons une description des classes et des individus.

2.1. JAFFE

C'est une base de données sur les expressions faciales de femmes japonaises (JAFFE). La base de données contient 213 images de 7 expressions faciales (6 expressions faciales de base + 1 neutre) posées par 10 modèles féminins japonais. Chaque image a été évaluée sur 6 adjectifs d'émotion par 60 sujets japonais. La base de données a été planifiée et assemblée par Michael Lyons, Miyuki Kamachi et Jiro Gyoba.
Les photos ont été prises au département de psychologie de l'université de Kyushu.

La répartition des individus ont fonction des classes est disponible sur le notebook (la version HTML).

2.2. MNIST5

Le jeu de données MNIST a été construit à partir de deux jeux de données de l'Institut national américain des normes et de la technologie (NIST). L'ensemble de formation comprend des chiffres manuscrits de 250 personnes différentes, 50% d'élèves du secondaire et 50% d'employés du Census Bureau. Notez que le jeu de test contient des chiffres manuscrits de personnes différentes suivant le même partage.

2.3. OPTIDIGITS

Ils ont utilisé des prétraitement mis à disposition par le NIST pour extraire des bitmaps normalisées de chiffres manuscrits à partir d'un formulaire préimprimé.
Les bitmaps 32x32 sont divisés en blocs de 4x4 distincts et le nombre de pixels inactifs est compté dans chaque bloc, cela a pour conséquence de généré une matrice d'entrée de 8x8 où chaque élément est un entier compris entre 0 et 16.

2.4. USPS

Un jeu de données sur des chiffres manuscrits cette fois avec 9298 lignes et 256 colonnes (soit des images de 16x16).

2.5. MFEA

Le jeu de données est composé d'écritures manuscrites extraits d'une collection de cartes utilitaires néerlandaises. Un total de 2000 observations sont disponibles sous la forme de 240 colonnes.

L'ACP 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 est le moteur central de toutes les autres méthodes d'analyse factorielles.
Dans notre cas, nous projeterons les points du jeux de données sur les deux premiers axes factoriels créant ainsi un nouvel espace, et grâce à ces nouveaux axes et en fonction de l'inertie expliquée, nous pourrons en avoir une idée de la structure des données.

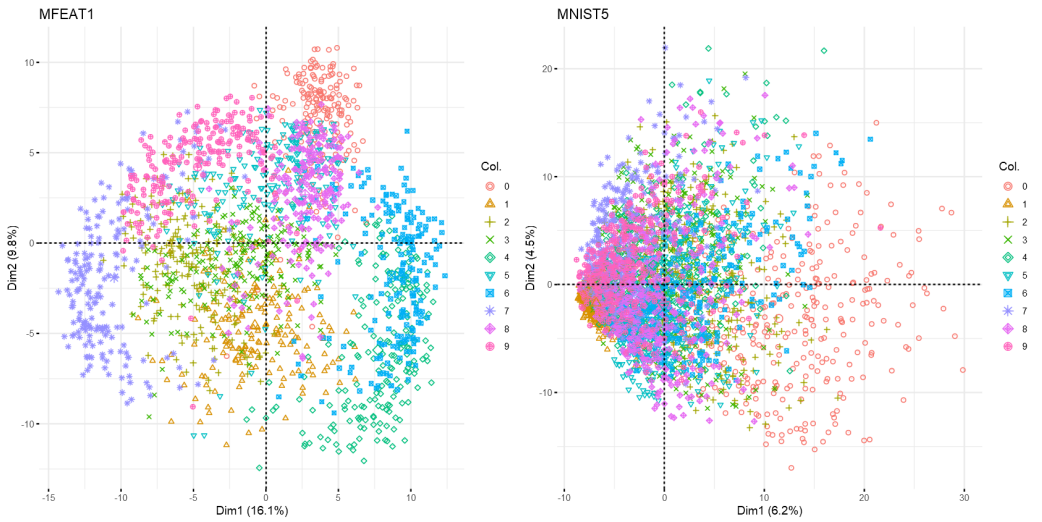


Figure 1: Projection des individus pour MFEAT et MNIST

l'ACP a du mal a séparer MNIST puisque chaque classe est mélangée avec une grande densité, le problème est trop complexe pour qu'il puisse être linéairement résolu. Mais en ce qui concerne MFEAT, les chiffres possédant une grande différence géométrique (6 et 7 par exemple) sont bien distingué et globalement il y a une certaine forme qui est dégager pour chaque classe.

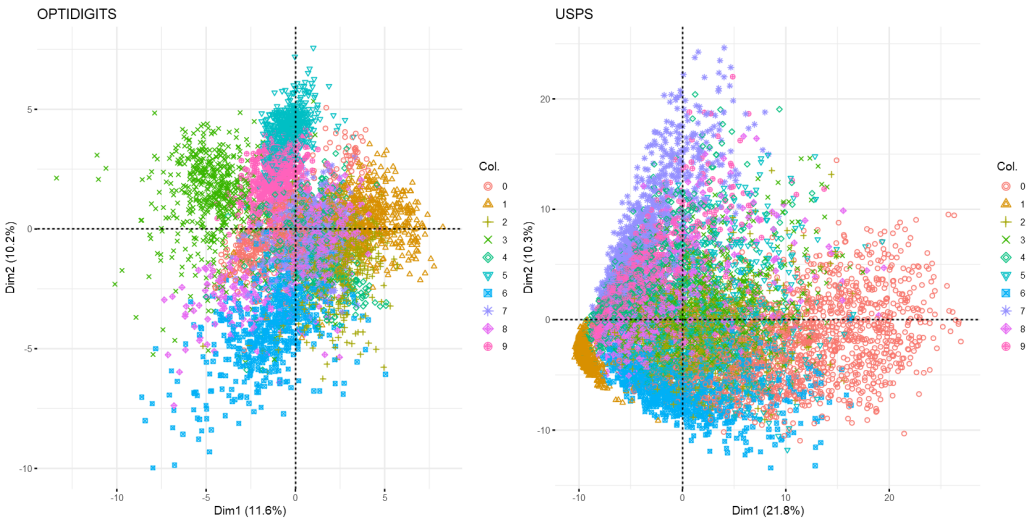


Figure 2: Projection des individus pour OPTDIGITS et USPS

Nous avons le même problème que MNIST concernant la projection factorielle sur les deux premiers axes pour USPS, on peut en conclure qu'il n'y a pas de grande différence globalement sur la topologie. Ensuite pour le jeu de données OPTIDIGITS, on remarque 4 classes se distinguent (1,3,5,6), cela s'explique aux prétraitement que le jeu de données a subi.

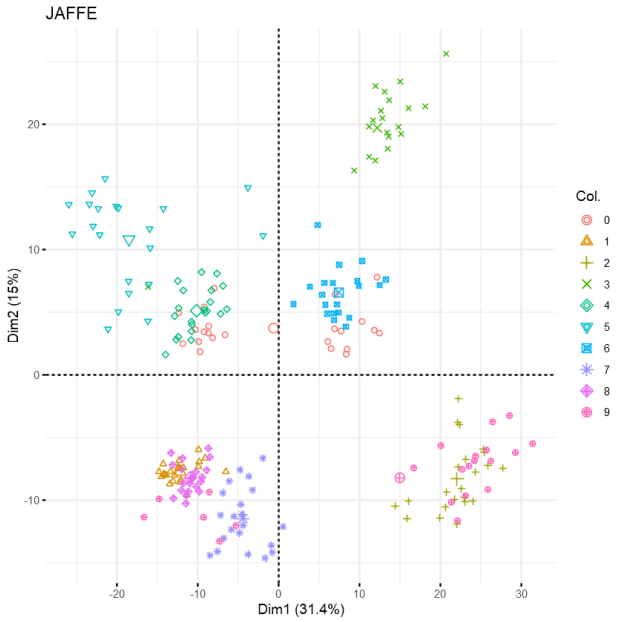
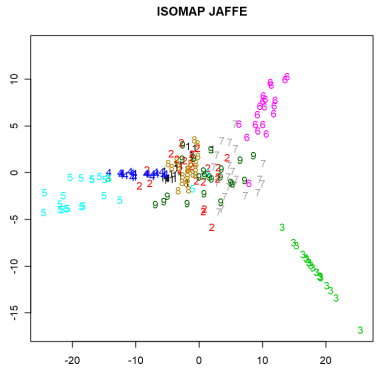
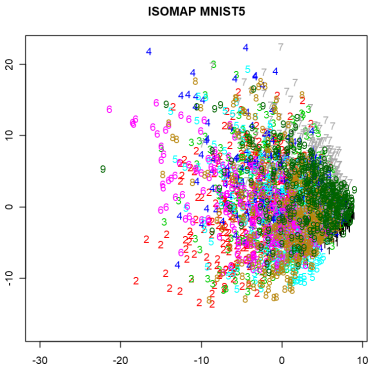


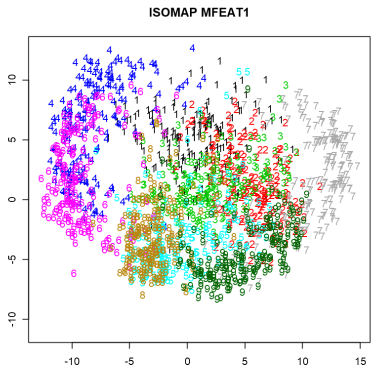
Figure 3: Projection des individus pour JAFFE

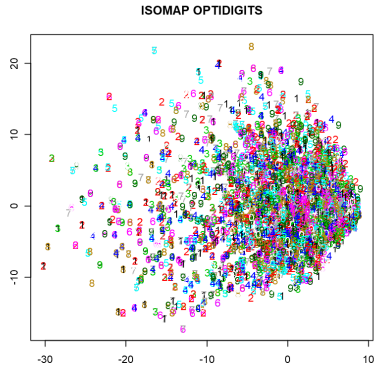
Dans la cas des expressions faciales, l'acp arrive à distinguer les différentes classes sans trop de problème aux vues du nombre d'observations relativement faible (213 images).

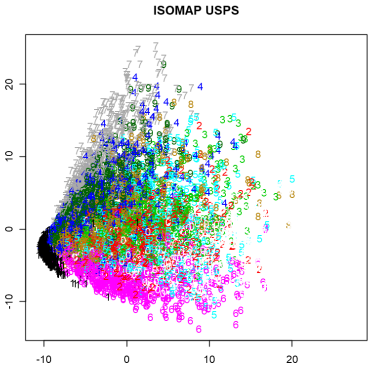
Ensuite l'Isomap sera initialisé par une ACP à 30 dimensions (plus de 90% d'inertie pour chaque jeux de données) et 10 voisins afin d'avoir une autre technique de visualisation avec une approche original. L'Isomap peut être considéré comme une extension de MDS ou de K-PCA. Cet algorithme cherche un espace de dimension réduite qui conserve les distances "géodésiques" entre tous les points.
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. Au final on représentera les données sur 2D.


ISOMAP JAFFE


ISOMAP MNIST5


ISOMAP MFEAT1


ISOMAP OPTIDIGITS


ISOMAP USPS

Nous avons voulu tester une autre méthode conservant la topologie globale des données. Nous constatons que le jeu de données *JAFFE* arrive a être bien discriminer une fois de plus avec en deuxième position *MFEAT1* où on distingue les différents chiffres. D'un autre côté on a toujours le même problème de discrimination pour *MNIST* et *USPS*, les classes ont une grandes densité dans une zone du plan. *OPTIDIGITS* possède une singularité, chaque lettre sont éparpillés sans frontière réel entre les classes. La méthode n'est pas approprié, nous devons nous concentrer sur des méthodes **non-linéaire** et aussi conservant une **topologie locale**.

La dernière technique de visualisation que nous utiliserons est la t-SNE puisqu'elle 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.

3. Classification

3.1. Classification Hierarchique Ascendante

La classification hiérarchique ascendante est un algorithme qui un ensemble de clusters dans lequel chaque cluster est distinct, et les objets de chaque cluster sont globalement similaires.
La classification hiérarchique peut être réalisée avec une matrice de distance ou des données brutes. Lorsque des données brutes sont fournies, le logiciel (dans notre cas le langage R) calcule automatiquement une matrice de distance en arrière-plan. La classification hiérarchique commence par traiter chaque observation comme une grappe distincte. Ensuite, il exécute à plusieurs reprises les deux étapes suivantes:

1. Identifier les deux clusters les plus proches l'un de l'autre.
2. Fusion des deux clusters les plus similaires.

Cela continue jusqu'à ce que tous les clusters soient fusionnés.
Comme pour les métriques de distance, le choix des critères de couplage doit être effectué sur la base de considérations théoriques du domaine d'application. La méthode de Ward consiste à regrouper les classes de façon que l'augmentation de l'inertie intraclasse soit maximale.

Une question théorique clé concerne les causes de la variation.
En l'absence de justifications théoriques claires pour le choix des critères de couplage, la méthode de Ward est la méthode par défaut raisonnable. Cette méthode calcule les observations à regrouper en réduisant la somme des distances au carré de chaque observation par rapport à la moyenne des observations d'un groupe. Nous utiliserons aussi le critère du minimum, maximum et moyenne.

3.2. Partitionnement

K-means est une méthode de partitionnement de données qui, à partir d'un ensemble de points, va pouvoir déterminer pour un nombre de classes fixé une répartition des points qui minimise un critère appelé *inertie* ou variance *intra-classe*.
Plus formellement étant donnée k, nous allons donc chercher à répartir les points $x_1, x_2, .., x_n$ en k groupes $C_1, C_2, \ldots, C_K$ de telle sorte que :

$$\sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{x \in C_k} ||x - (\frac{1}{|C_k|} \sum_{x \in C_k} x)||^2$$

soit le plus minimale.

3.3. Résultats

Nous créons une fonction permettant de calculer les clusters suivant les 5 méthodes comme ceci :

```
five_cluster_method = function(data_images,min_nc=6,max_nc=15,acp=FALSE){
  res = list()
  indices_vector = c("silhouette")
  if(acp == TRUE){
    pca = PCA(data_images,ncp = 2)
    data_images = pca$ind$coord
  }
  res$kmeans = NbClust(data=data_images, min.nc = min_nc, max.nc = max_nc, method = "kmeans", index = indices_vector)
  res$cah_complete = NbClust(data=data_images, min.nc = min_nc, max.nc = max_nc, method = "complete", index = indices_vector)
  res$cah_single = NbClust(data=data_images, min.nc = min_nc, max.nc = max_nc, method = "single", index = indices_vector)
  res$cah_ward = NbClust(data=data_images, min.nc = min_nc, max.nc = max_nc, method = "ward.D2", index = indices_vector)
  res$cah_average = NbClust(data=data_images, min.nc = min_nc, max.nc = max_nc, method = "average", index = indices_vector)
  return(res)
}

```

On récupère les résultat pour chaque méthode et on obtiens pour les jeux de données sans ACP: *OPTDGITS* possède son meilleur résultat avec 9 clusters en utilisant une CAH Ward. Ensuite *JAFFE* est a 12 clusters pour chaque méthode, nous soulignons que la meilleur silhouette est pour K-means. Pour K-means, la méthode fonctionne plutôt bien pour *MFEAT1* puisque nous avons le meilleur score pour un nombre de classe à 9. En revanche, pour le jeu de données MNIST, nous avons un nombre commun qui est 8 (CAH Average).
Finalement, le jeu de données *USPS* est à 10 nombre de classes avec une CAH Complète.

On applique une ACP en récupérant les deux premières composantes principales puis on applique les 5 méthodes : La CAH avec le critère de Ward, nous donne un résultat de 10 classes pour *OPTDIGITS* et MFEAT1. K-means retourne 9 classes pour *JAFFE* alors que *USPS*, la méthode retourne le bon nombre de classes. CAH avec le critère du minimum retourne le meilleur

résultat pour *MNIST* (9 classes à une silhouette à 0.4541).

3.4. Classification avec ou sans analyse en composantes principales

Dans cette section, nous comparons les méthodes avec le vrai nombre de classes suivant ou non une initialisation en ACP.

Table 1: Résultats CAH (Ward, Single, Average, Complete), K-means Pour JAFFE

	accuracy_normal	accuracy_pca	kappa_normal	kappa_pca
kmeans	63.84977	81.69014	59.85460	79.66114
cah_complete	46.00939	49.29577	40.05286	43.72110
cah_single	44.60094	35.21127	38.36987	27.55281
cah_ward	45.53991	71.83099	39.46788	68.69872
cah_average	38.02817	51.17371	31.18101	45.71520

On a de meilleur résultat avec une initialisation suivant une ACP globalement pour les 5 jeux de données, l'accuracy a un gain de 20-25% pour chaque dataset. Il y a quelques cas d'exception comme par exemple *JAFFE*, les classes étant élongées et faiblement séparer, il se produit un effet de chaine du à la méthode.
Les CAH avec le critère du lien minimum et moyenne se voient en baisse de performance à la suite d'une ACP sauf *OPTIDIGTS* dans laquelle la méthode du lien moyen va avoir de meilleur résultat. Graphiquement on peut constater sur les deux plan factoriel, des classes volumiques pondérés.
Les autres résultats sont disponibles en annexe.

4. Classification avec un modèle de mélange gaussian

Le modèle de mélange fini de lois de probabilité consiste à supposer que les données proviennent d'une source contenant plusieurs sous-populations homogènes appelées composants. De façon générale, les mélanges paramétriques se caractérisent par l'existence d'hypothèses sur la distribution de probabilité induisant une classification. La distribution de probabilité appartient à une famille paramétrique c'est à dire l'espace des paramètres est de dimension finie.

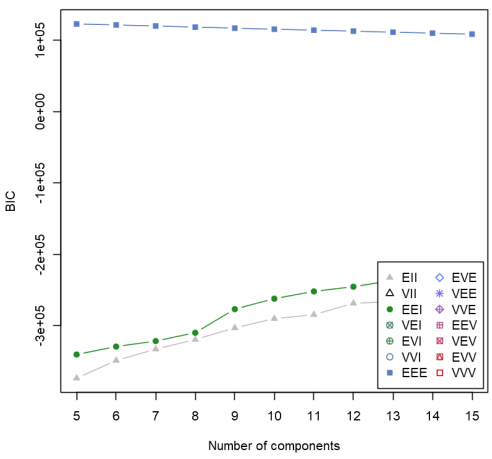
4.1. L'approche du maximum de vraisemblance : l'algorithme EM

L'approche ML consiste à maximiser la vraisemblance ou la log vraisemblance afin d'estimer les paramètres inconnus. Toutefois, ce problème de maximisation ne peut être résolu analytiquement en raison de données cachées. Il faut donc trouver les solutions à l'aide d'algorithmes itératifs. Parmi ces algorithmes, figure l'algorithme EM.

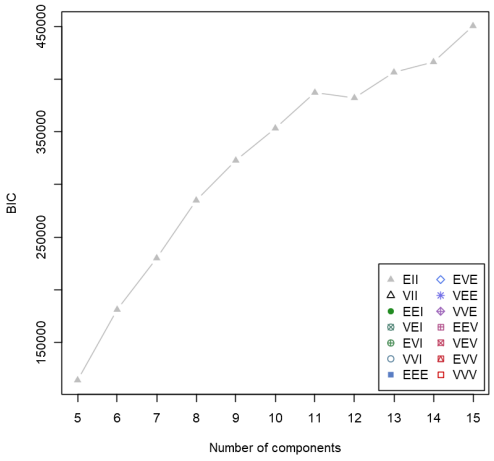
4.2. Évaluation des résultats

4.2.1. Données initiales

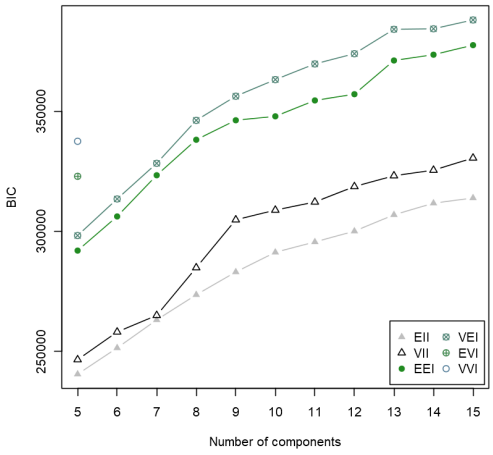
Tout d'abord, nous avons utilisé les packages mclust et Rmixmod et nous avons rencontrer des problèmes de mémoire avec Rmixmod, alors les résultats un nombre de classe allant de 5 à 15 sont seulement disponible sous mclust. On constate que sur le jeu de données initiales, le modèles de m"langes a beacoup de mal à reconnaitre le nombre de classe dù au nombre de variables très élévées.



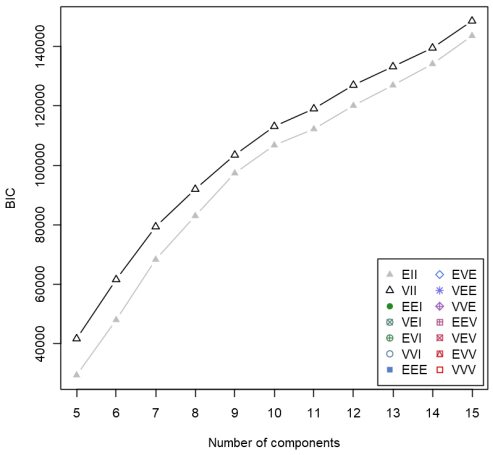
(f) Graphique du BIC pour MFEAT



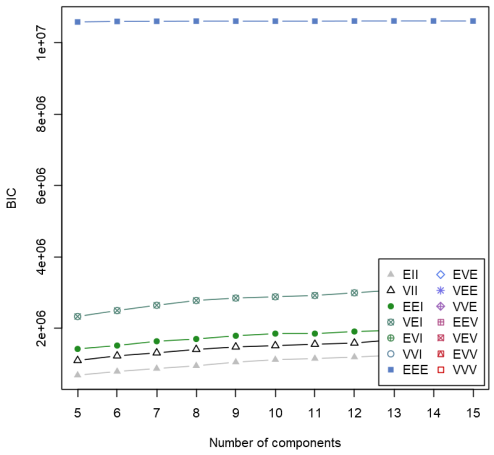
(g) Graphique du BIC pour MNIST



(h) Graphique du BIC pour JAFFE



(i) Graphique du BIC pour OPTDIGITS



(j) Graphique du BIC pour USPS

On voit que pour tous les datasets ont ne peux pas trancher avec le critère BIC, nous verrons par la suite qu'avec une ACP, les résultats seront interpretables.

Nom du jeux de données	NbCluster avec Rmixmod	NbCluster avec mclust
JAFFE	7	7
MFEAT	13	8
MNIST	8	8
OPTDIGITS	10	9
USPS	18	9

4.2.2. Données ayant subi une ACP

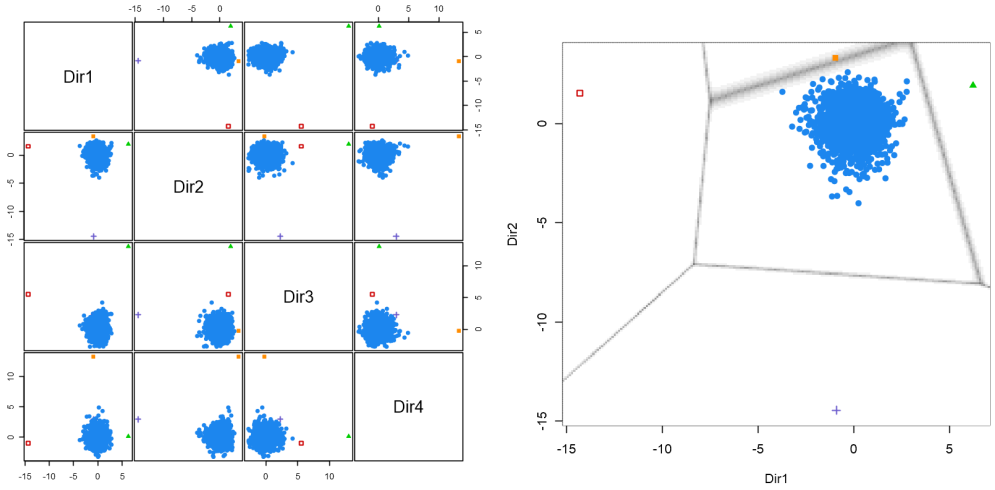
Nous lançons mclust et Rmixmod pour chaque jeu de données en initialisant avec une ACP. Les paramètres pour mclust sont laisser par défaut mais en revanche nous avons modifié la stratégie d'exécution pour Rmixmod. Nous avons utilisé une initialisation *SEMMax* afin de régler les problèmes du choix du nombre de la classe ainsi que les problèmes d'initialisation du *smallEM*. Nous avons aussi augmenter le nombre d'essais et le nombre de convergence que ça soit en initialisation ou aussi durant l'exécution de l'algortihme afin de comparer avec mclust.

Le modèle associer à *JAFFE* est VEE c'est à dire que seul le volume est variable, comme on a pu le voir si l'ACP. *OPTDIGITS* et *USPS* sont VVV, le volume ainsi que la forme et orientation sont variables. Les directions des classes pour *MFEAT* peuvent en effet être vu comme identique. De plus, pour *MNIST*, on remarque le modèle possède une forme égales entre classes. Puisque les calculs allaient plus vite avec le jeu de données réduit, nous avons agrandi l'intervalle de classe. *OPTDIGITS* arrive à avoir ses nombres de classes à 10 avec RmixMod mais la méthode possède des difficultés pour *MFEAT* et *USPS*. mclust est plus fiable que Rmixmod, l'initialisation avec une CAH donne des résultats fiables suivant les 5 jeux de données.
Les graphiques du BIC sont disponibles en annexes et aussi sur le notebook ainsi que sa version HTML.

4.3. MclustDR

C'est une méthode de réduction de dimensions permettant de visualiser la structure des classes suite à la classification obtenue à partir d'un modèle de mélange gaussien.
La méthode vise à réduire la dimension en identifiant un ensemble de combinaisons linéaires, classées par ordre d'importance, quantifiées par les valeurs propres associées, des caractéristiques d'origine qui capturent la majeure partie de notre jeu de données.

Nous esseyons cette fonction avec les objets mclust calculé sur les données d'origines. Les résultats ne sont pas très concluant puisque pour le jeu de données *USPS*, nous avons un nombre de classe à 13. Le jeu de données *OPTDIGITS* est à 15 ainsi que *MNIST* et *JAFFE*.
Finalement le jeu de données *MFEAT*, nous retourne un nombre de classe à 5 avec 1992 observation dans la première classe et 2 dans les 4 autres classes.



(a) Diagramme pairs MFEAT

(b) Frontière de décision MFEAT

4.4. Comparatif avec le vrai nombre de classes avec GMM

Nous utiliserons la NMI, l'ARI et aussi la sensibilité (le taux de mal classé).
L'information mutuelle (MI) de deux variables aléatoires est une mesure de la dépendance mutuelle entre les deux variables. Plus précisément, il quantifie la "quantité d'informations" (en bits) obtenues à propos d'une variable aléatoire en observant l'autre variable aléatoire.

La mesure de l'indice Rand en statistiques est une mesure de la similarité entre deux clusters de données. l'index de Rand est lié à la précision, mais est applicable même lorsque les labels ne sont pas utilisées puisque nous mesurons la consistance (le taux d'accord) entre deux clusters.

Table 2: Mixmod

	ARI	NMI	Sensitivity	Specificity
JAFFE	0.5829272	0.7453579	59.33117	95.40517
MFEAT	0.2646793	0.4664391	39.30000	93.25556
OPTDIGITS	0.2755413	0.4343411	38.53692	93.15606
MNIST	0.1068012	0.2093746	23.86142	91.56160
USPS	0.1905894	0.3718075	28.71948	92.39658

Table 3: Mclust

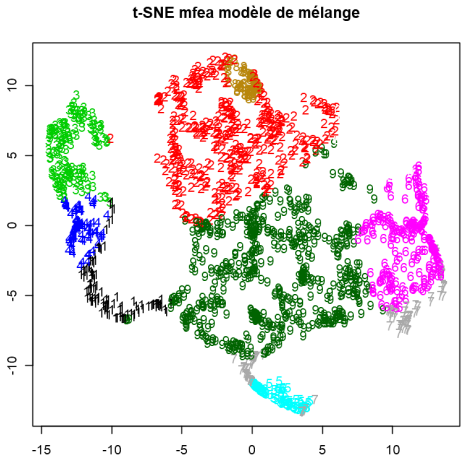
	ARI	NMI	Sensitivity	Specificity
JAFFE	0.5471135	0.7298906	52.57143	94.61926
MFEAT	0.2868834	0.4573999	46.45000	94.05000
OPTDIGITS	0.3060481	0.4303758	50.70569	94.51981
MNIST	0.1144522	0.2049724	21.90802	91.39795
USPS	0.2616559	0.3903143	38.71661	93.46606

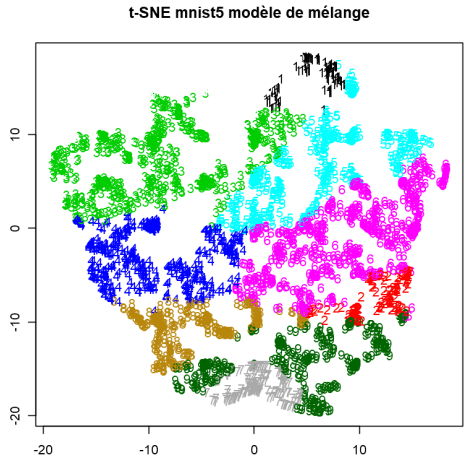
Nous avons de meilleur résultat globalement avec le package Rmixmod, cela s'explique par le nombre d'itération accordé à l'algorithme ainsi que le nombre d'essei à l'initialisation avec "SEMmax" réglant les problèmes du nombre de classes.

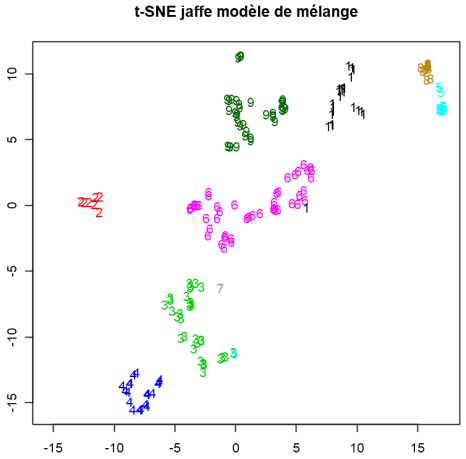
5. Visualisation avec t-SNE

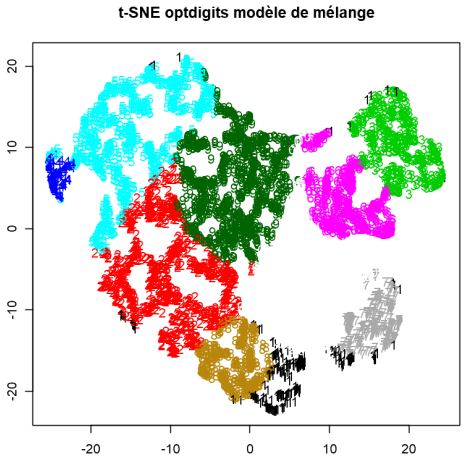
t-SNE tente de trouver une configuration optimale selon un critère de théorie de l'information pour respecter les proximités entre points : deux points qui sont proches (resp. éloignés) dans l'espace d'origine devront être proches (resp. éloignés) dans l'espace de faible dimension.
Une distribution de 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.
La divergence de Kullback-Leibler n'est pas une métrique propre, car elle n'est pas symétrique et ne satisfait pas non plus l'inégalité triangulaire. Donc, la divergence KL est préférable de ne pas être interprétée comme une

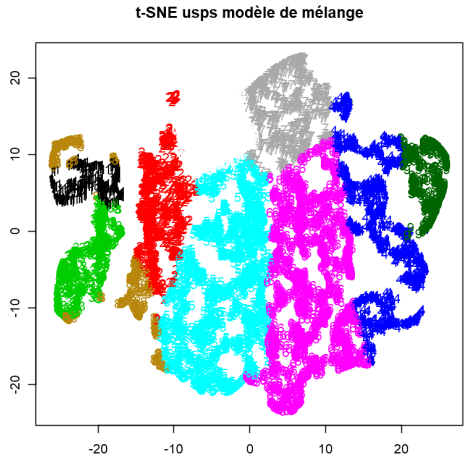
"mesure de distance" entre les distributions, mais plutôt comme une mesure de l'augmentation d'entropie due à l'utilisation d'une approximation de la distribution vraie plutôt que de la distribution vraie elle-même.







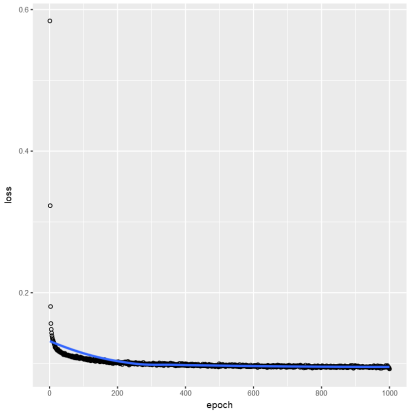




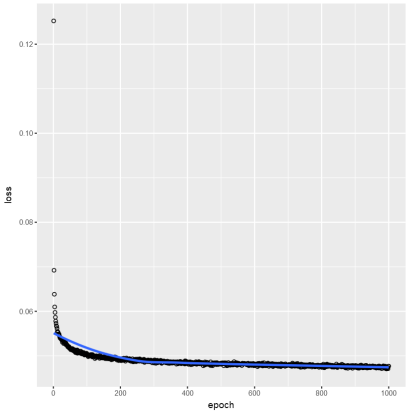
On voit que globalement pour tout les jeu de données testés, la t-SNE donne de bon résultats. En effet, les classes sont vraiment bien séparées et homogènes. Nous avons utilisées les modèles provenant mixmod car elles ont de meilleurs distribution parmi les partitions comme vu précédemment avec l'ARI et la NMI.

6. Autoencodeur

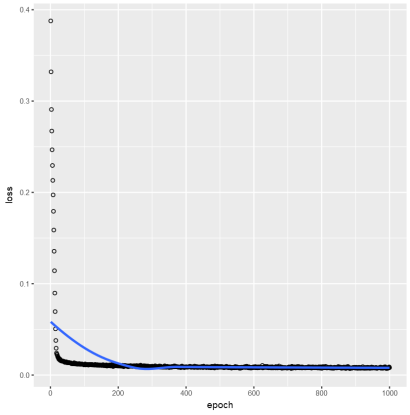
Les auto-encodeurs sont appris automatiquement à partir d'exemples de données. Cela signifie qu'il est facile de former des instances spécialisées de l'algorithme qui fonctionneront bien avec un type d'entrée spécifique.
Un auto-encodeur est un réseau de neurones comportant trois couches: une couche d'entrée, une couche cachée (codage) et une couche de décodage. Le réseau est formé pour reconstruire ses entrées, ce qui oblige la couche cachée à essayer d'obtenir de bonnes représentations des entrées. De nos jours, le débruitage des données et la réduction de la dimensionnalité pour la visualisation des données sont considérés comme deux principales applications pratiques intéressantes des autoencodeurs.
La fonction coût utilisé pour comparer les images est MSE (l'erreur moyen carré) afin de comparer les images originaux et ceux en sortie du décodeur.



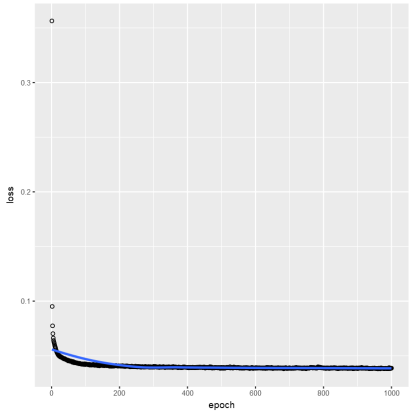
(h) Loss MFEAT



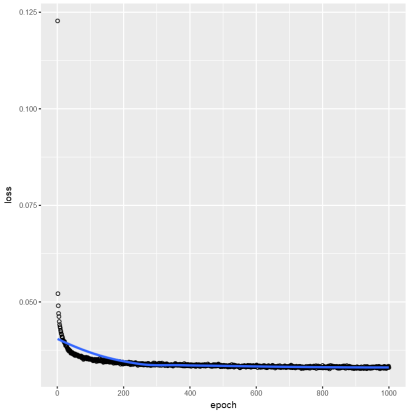
(i) Loss MNIST



(j) Loss JAFFE



(k) Loss OPTIDIGITS



(l) Loss USPS

Si le seul but des auto-encodeurs était de copier l'entrée dans la sortie, ils seraient inutiles. En effet, nous espérons qu'en entraînant l'auto-codeur à copier l'entrée dans la sortie, la représentation latente h aura des propriétés utiles.
Avec des contraintes de dimensionnalité et de parcimonie appropriées, les auto-encodeurs peuvent apprendre des projections de données plus intéressantes que l'ACP ou d'autres techniques de base.
Nous effectuons de nouveau le modèle de mélange sur les 5 jeux de données encodées. (les résultats sont disponibles en annexe, ou sur le notebook/HTML).
Nous avons un gain significatif sur la précision ainsi que la NMI et l'ARI.
Nos données étant encodées, nous avons une meilleure qualité de partition.

7. Modèle de mélange et analyse factorielle simultanés

L'analyse des facteurs de mélange est examinée afin d'estimer avec souplesse les facteurs latents continus et non distribués normalement. Une étude de simulation compare le facteur de mélange et maximise la vraisemblance. le modèle d'analyse factorielle du mélange produit des estimateurs presque non biaisés et offre une probabilité de couverture très bonne.
Nous utilisons la fonction *mfa* provenant du package *EMMIXmfa* :

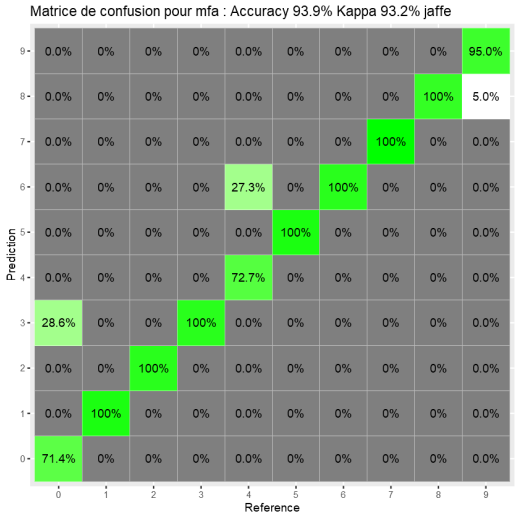


Figure 4: Matrice de confusion pour le jeu de donnés JAFFE

Nous avons de très bon résultat puis que la NMI est de 0.93 et l'ARI est de 0.87 donc la qualité de nos partitions est très bonne.
En revanche cette méthode est très couteuse en temps de calcul. Le modèle de petites diffultés à dinstiguer des lettres dont la forme géometrique sont très proches (0 et 3) ainsi que les 9 et 8.

8. Conclusion

Dans ce projet, nous avons fait beaucoup de comparaisons entre les 5 différents jeux de données. Nous avons vu plusieurs types de méthodes de classification (classification ascendante hiérarchique, partitionnement avec K-means) mais avons pu explorer les modèles de mélanges de plusieurs manières. En effet, nous avons pu remarquer que les méthodes d'analyse factorielle peuvent aider à l'apprentissage d'un modèle de mélange sans avoir à envier des auto-encodeurs, mais finalement la complexité temporelle de celles-ci laisse à désirer.
De plus les mathématiques derrière les modèles de mélanges sont plus difficilement accessibles et donc moins attrayant au grand public.
Le projet a été très instructif puisque nous avons pu mettre en oeuvre beaucoup de méthode provenant de l'apprentissage non-supervisé.

Table .4: Résultats CAH (ward, single, average, complete), Kmeans Pour JAFFE

	accuracy_normal	accuracy_pca	kappa_normal	kappa_pca
kmeans	63.84977	81.69014	59.85460	79.66114
cah_complete	46.00939	49.29577	40.05286	43.72110
cah_single	44.60094	35.21127	38.36987	27.55281
cah_ward	45.53991	71.83099	39.46788	68.69872
cah_average	38.02817	51.17371	31.18101	45.71520

Table .5: Résultats CAH (ward, single, average, complete), Kmeans Pour MFEAT1

	accuracy_normal	accuracy_pca	kappa_normal	kappa_pca
kmeans	50.45	81.80	44.9444444	79.7777778
cah_complete	32.55	42.50	25.0555556	36.1111111
cah_single	10.30	10.35	0.3333333	0.3888889
cah_ward	40.70	58.50	34.1111111	53.8888889
cah_average	39.65	46.00	32.9444444	40.0000000

Table .6: Résultats CAH (ward, single, average, complete), Kmeans Pour MNIST

	accuracy_normal	accuracy_pca	kappa_normal	kappa_pca
kmeans	27.09585	46.95279	18.8592037	40.953714
cah_complete	22.46066	29.38484	13.4712180	20.971200
cah_single	11.33047	11.41631	0.1018995	0.197428
cah_ward	23.46209	60.68670	14.8148398	56.286786
cah_average	19.62804	21.83119	9.8667261	12.114689

Table .7: Résultats CAH (ward, single, average, complete), Kmeans Pour OPTDIGITS

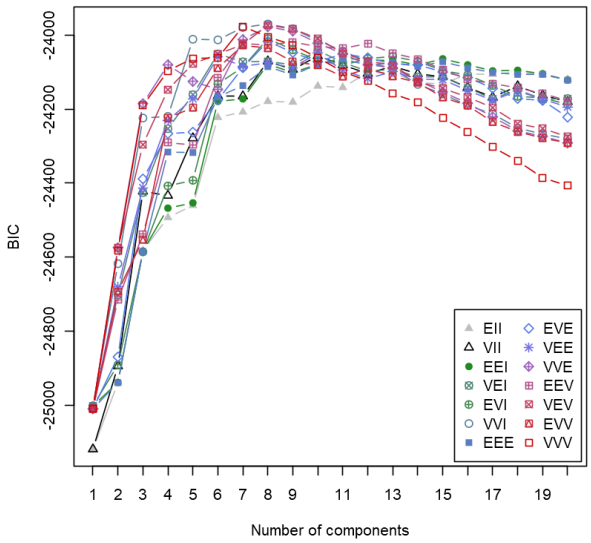
	accuracy_normal	accuracy_pca	kappa_normal	kappa_pca
kmeans	51.29893	66.63701	45.8960069	62.93583631
cah_complete	38.11388	38.79004	31.2877913	32.01561663
cah_single	10.28470	10.26690	0.1193342	0.09942239
cah_ward	39.09253	73.82562	32.3335231	70.91625467
cah_average	33.25623	51.19217	25.8773822	45.77220538

Table .8: Résultats CAH (ward, single, average, complete), Kmeans Pour USPS

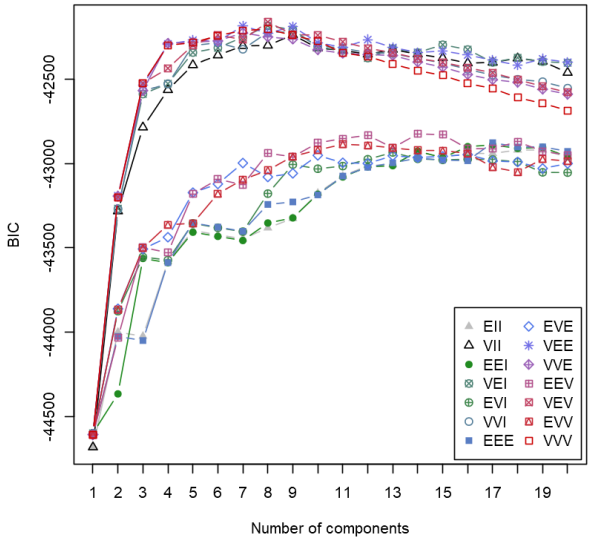
	accuracy_normal	accuracy_pca	kappa_normal	kappa_pca
kmeans	37.18004	67.32631	30.23970249	63.63695112
cah_complete	29.90966	32.80275	20.79732324	25.01416426
cah_single	16.75629	16.72403	0.08855578	0.03447412
cah_ward	38.55668	56.89396	31.73250294	52.15305564
cah_average	31.08195	22.15530	22.78588582	9.63928958

Figure .5: Extrait images USPS

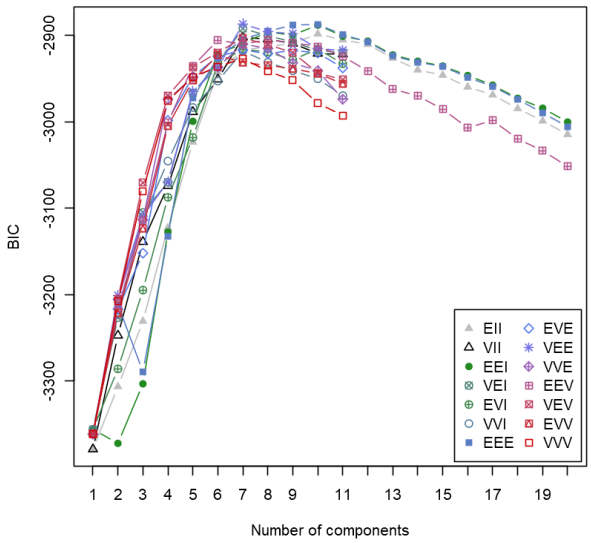
Figure .6: Extrait images OPTDIGITS



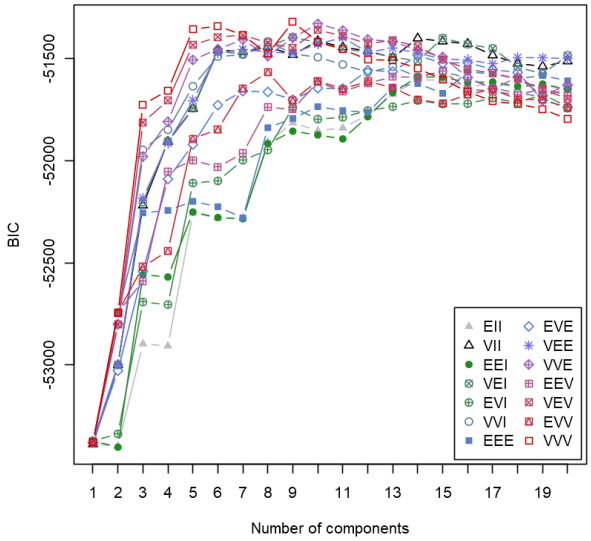
(a) BIC avec une ACP MFEAT



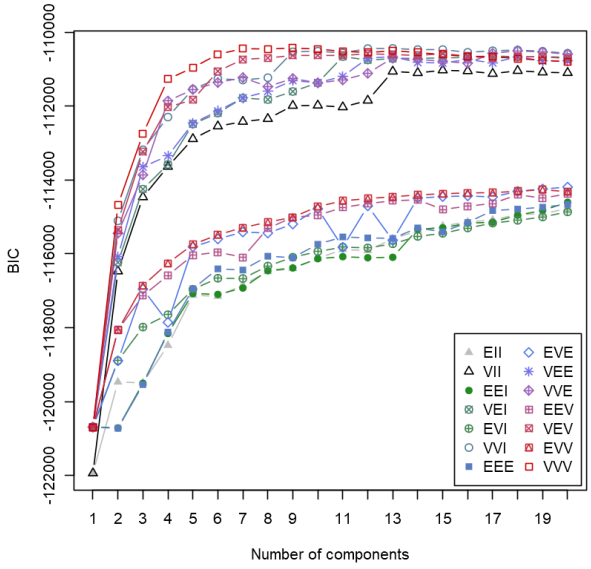
(b) BIC avec une ACP MNIST



(c) BIC avec une ACP JAFFE



(d) BIC avec une ACP OPTDIGITS



(e) BIC avec une ACP USPS

Table .9: Résultats MM (mclust, mixmod), avec autoencoder sur JAFFE

	ari	nmi	sensivity_tpr	specificity_tnr
mclust	62.51010	79.52406	69.17749	96.56947
mixmod	66.27523	81.17323	77.25174	97.43692

Table .10: Résultats MM (mclust, mixmod), avec autoencoder sur USPS

	ari	nmi	sensivity_tpr	specificity_tnr
mclust	33.31823	52.10924	51.82345	94.74357
mixmod	22.47571	42.95288	37.43026	93.16504

Table .11: Résultats MM (mclust, mixmod), avec autoencoder sur MFEAT

	ari	nmi	sensivity_tpr	specificity_tnr
mclust	44.02329	61.35408	51.80	94.64444
mixmod	34.26052	54.11048	37.75	93.08333

Table .12: Résultats MM (mclust, mixmod), avec autoencoder sur MNIST

	ari	nmi	sensivity_tpr	specificity_tnr
mclust	19.43876	36.79277	31.94877	92.47442
mixmod	14.46603	31.84605	24.04864	91.54760

Table .13: Résultats MM (mclust, mixmod), avec autoencoder sur OPTDIGITS

	ari	nmi	sensivity_tpr	specificity_tnr
mclust	53.32243	65.66657	63.54676	95.95094
mixmod	45.47684	59.23033	48.04800	94.19552