Projet Model Based Learning

Mouhamadou Mansour Lo , Mame Libasse Mboup

***Abstract*— Ce document est le compte rendu du projet de Manifold Learning qui s'inscrit dans le cadre du Master 2 Data-mining de l'Université de Lyon 2. Il est une étude comparative de la performance de plusieurs méthodes de réduction de dimension appliquées sur des jeux de données artificiels comme reel.**

***Keywords*— Manifold Learning Dimensionality reduction Convexity**

CONTENTS

I Introduction 1

II Objectifs 1

III Matériels et Méthodes 1

IV Jeu de Données 2
- IV-A Jeu de Données Artificielles 2
- IV-B Jeu de Données réel 4

V Expériences numériques sur les données artificielless 4
- V-A Calibration des Méthodes 4
- V-B Utilisation des Méthodes 4

VI Comparaison des méthodes 5

VII Application sur données réelles 6

References 7

I. INTRODUCTION

De nos jours, le monde qui nous entoure se remplit au fur et à mesure de l'explosion des nouvelles technologies de données brutes de plus en plus volumineux et complexe.

Ces données peuvent provenir de diverses sources comme des capteurs d'IOT à hautes fré- quences , des données publiques disponible sur des sites comme Data-Gouv , des études de la population par l'INSEE . . .

Malheureusement, ces données sont en de très grandes dimensions : le nombre de colonnes ou de variables sont très élevés.

Cette grosse dimension des données rend les plupart les modèles de Machine Learning inutilisable, ce phénomène se nomme plus précisément la malédiction de la dimensionalité

Mouhamadou Mansour Lo, ICOM, Université de Lyon 2, Departement Informatique, E-mail: m.lo@univ-lyon2.fr; Mame Libasse Mboup, ICOM, Université de Lyon 2, E-mail: mame-libasse.mboup@univ-lyon2.fr

Ces données massives nécessitent donc d'être traitées et résumées pour qu'elles puissent être comprises et interprétés par des experts métiers.

Plusieurs types de méthodes de réduction de dimension existent :

- Les méthodes de réduction de dimension linéaires
- Les méthodes de réduction de dimension non linéaires

La méthode de réduction de dimension principalement utilisée aujourd'hui est la **PCA**(Principal Component Analysis) qui est une méthode linéaire.

Cette méthode basée sur des notions de distance euclidienne se retrouve vite inutile quand la dimension des données explose.

C'est pour cela que nous allons dans les parties qui suivent étudier et comparer diverses méthodes de réduction de dimension et voir leurs performances et pertinences par rapport des jeux de données artificiels et un jeu de donnée réel .

II. OBJECTIFS

Comme indiqué précédemment , nous nous devons d'implémenter des méthodes de réduction de dimension plus sophistiquées des lors que la donnée est en très grande dimension ou des que la structure intrinsèque de la donnée est trop complexe.

Nous avons donc choisi pour la suite de tester quelques méthodes de réduction de dimension se basant sur la distance euclidienne : **PCA** et **MDS** et des méthodes non linéaires convexes :

1) Full Spectral
   - Une méthode se basant sur la distance géodésique : **Isomap**
   - Une méthode de type Kernel-based : **Kernel PCA**
2) Sparse Spectral
   - Une méthode se basant sur la reconstruction de poids : **LLE**
   - Une méthode de se basant sur la Neighborhood graph Laplacian : **Laplacian Eigenmaps**

Nous avons essayé de prendre une large palette de méthodes afin de tester un maximum de possibilité et pour se faire une idée la plus globale possible sur les performances générales de ces méthodes.

III. MATÉRIELS ET MÉTHODES

Dans cette partie , nous allons décrire le fonctionnement général des différentes méthodes utilisés au cours de nos expériences .

MDS : Classical Scaling

MDS ou Multidimensional Scaling est une famille de méthode de réduction de dimension lineaire qui est pratiquement identique a PCA , d'ailleur on le nomme aussi **PCoA** (Principal Coordinates Analysis).

Cette méthode prend en entrée une matrice de distance euclidienne avec des paires de points

L'algorithme d'obtention des distances en faible dimension est le suivant :

- Si l'entrée de l'algorithme est un point Y : on calcule la matrice de **produit scalaire** entre les points deux a deux **S**.
- Sinon si l'entrée est une matrice de distance euclidienne , on les transforme en produit scalaire grace la matrice **D** puis **S**.
- On réalise la décomposition spectrale par **Eigen**.
- Puis enfin , La représentation est obtenue en réalisant le produit de la matrice identité par des vecteurs obtenus après décomposition spectrale.

Isomap

Isomap est une méthode réduction de dimension basée sur les distances **géodésiques** et qui a pour principale objectif le maintient de la distance géodésique entre les points d'un jeu de donnée.

Une distance géodesique est la distance entre deux points mesuré au niveau de la **variété** topologique .

Cette distance s'obtient en construisant le **Graphe de voisinage** dans lequel chaque point est connecté a ses **K** plus proches voisins.

Le chemin le plus court est ainsi une **estimation** de cette distance géodésique entre ces deux points.

Une **matrice** de distance géodésique est alors construite avec des paires de points , puis la représentation en faible dimension des points s'obtient en appliquant l'algorithme de **Classical Scaling** sur cette matrice obtenue.

Kernel PCA

Kernel PCA est une méthode réduction de dimension qui est une reformulation de la PCA classique dans un espace en grande dimension en utilisant des fonctions noyeaux.

Kernel PCA calcule les **valeurs propres** par décomposition spectrale de la matrice de noyeaux, plutot que ceux de la matrice de covariance.

Ce qui donne a Kernel PCA de construire des **mapping** non lineaire.

LLE

LLE ou Local Linear Embedding est une méthode de réduction de dimension de type Sparse Spectral similaire a Isomap.

Sa similitude avec Isomap vient du faite qu'il construit une représentation en **Graphe** des points du jeu de données mais lui par contre ne cherche qu'a seulement maintenir les **propriétés locales** des points

Les propriétés locales des points de la **variété** topologique sont construites en écrivant les points en grande dimensions comme étant une combinaison linéaire de leurs voisins les plus proches.

En faible dimension , LLE tente de maintenir la reconstruction des poids de la combinaison lineaire du mieux possible.

Laplacian Eigenmaps

Laplacian Eigenmaps est une méthode de réduction de dimension de type Sparse Spectral similaire a LLE.

Cette méthode cherche une représentation des données en faible dimension , en préservant les propriétés locales de la variété .

Les propriétés locales se basent sur les distances deux a deux des **points voisins**.

Elle cherche une représentation des données en faible dimension des points tel que la distance entre un point et ses K plus proches voisins soit minimisée. Cette minimisation se définit comme un "eigenproblem" en utilisant la **théorie spectrale** des graphes

IV. Jeu de Données

Dans cette partie, Nous allons parler des données utilisées pour mener a bien nos expériences .

A. Jeu de Données Artificielles

Nous avons choisis d'utiliser 4 **datasets Artificielles** donc obtenus par simulation .

Nous retrouvons entre autres :

- Le dataset **SwissRoll**
- Le dataset **Broken-SwissRoll**
- Le dataset **Helix**
- Le dataset **Twinpeaks**

Tout naturellement , nous avons choisi **Swissroll** qui est un dataset déjà vu en cours et dans les séances de TPs pour

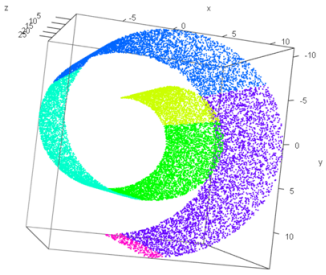


Fig. 1. Visualisation 3D du Dataset Swissroll

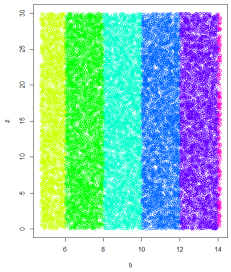


Fig. 2. Visualisation de Swissroll dans sa véritable dimension

servir de base de comparaison connues.

Ce dataset est en 3 Dimension(X,Y,Z) mais sa dimension intrinsèque est plus petite que 3.

Le second dataset est **brokenswissroll** qui est lui aussi en 3 Dimension(X,Y,Z) mais sa dimension intrinsèque est plus petite que 3.

Le troisieme dataset est **helix** qui est lui en 2 Dimension(X,Y) mais sa dimension intrinsèque est plus petite que 2.

Le quatrième et dernier dataset artificiel est **Twinpeaks** qui est lui en 3 Dimension(X,Y,Z) mais sa dimension intrinsèque est plus petite que 3.

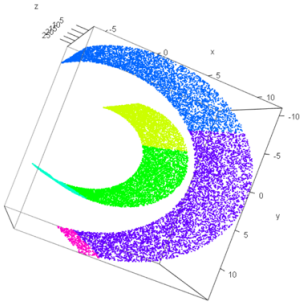


Fig. 3. Visualisation 3D du Dataset Brokenswissroll

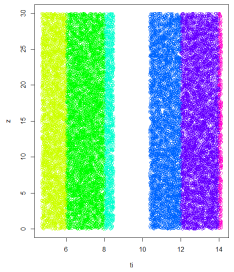


Fig. 4. Visualisation de Brokenswissroll dans sa véritable dimension

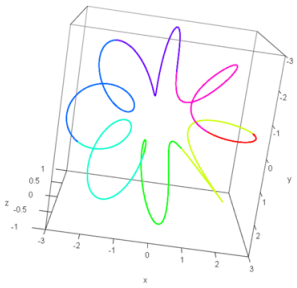


Fig. 5. Visualisation 3D du Dataset Helix

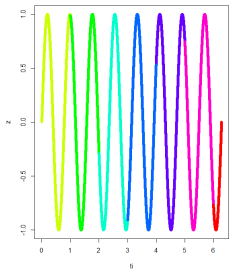


Fig. 6. Visualisation de Helix dans sa véritable dimension

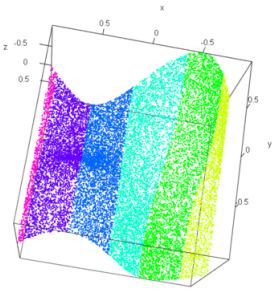


Fig. 7. Visualisation 3D du Dataset Helix

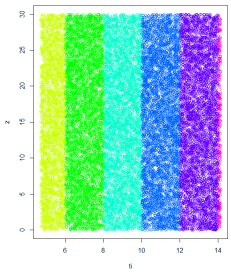


Fig. 8. Visualisation de Helix dans sa véritable dimension

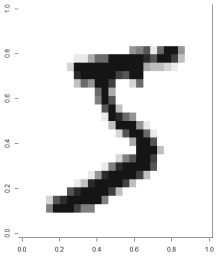


Fig. 9. Exemple de visualisation d'un chiffre du dataset MNIST

B. Jeu de Données réel

Pour le jeu de données réel , nous avons décidé de choisir le dataset **MNIST** qui est un jeu de données de chiffre écrits a la main utilisé principalement pour la détection de chiffre (1 a 9) en apprentissage supervisé.

Ce jeu de données contient **60000** observations et **785** variables (**28*28 pixels** et 1 variable cible [1-9]) pour le fichier de Train et **10000** observations et 785 variables pour le fichier Test. Il est disponible sur le site de Yann Lecun

Le code de génération des données artificielles est disponibles dans le fichier **Simulation_ML.R**

V. Expériences numériques sur les données artificiellesss

A. Calibration des Méthodes

Nous avons utilisés 3 méthodes de calibration de la dimension intreseque :

- Calibration par **CorrDim**
- Calibration par **ACP Normée**
- Calibration par **Kernel PCA**
- Calibration par **MLE** (Maximum Likelihood Estimation)

Dataset	*Swissroll*	*BrokenSwissroll*	*Helix*	*Twinpeaks*
Dimension Estimée	1.97	1.98	1	1.97

TABLE I

Résultat de la recherche de Dimension intrinsèque par MLE

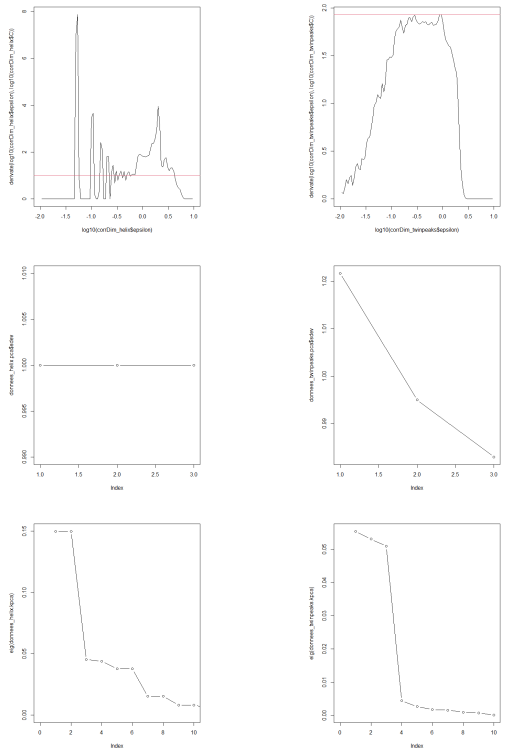


Fig. 10. Estimation de la dimention par ACP Normé , Kernel ACP , CorrDim pour a gauche Helix , a Droite Twinpeaks

Les résultats de ces différentes techniques nous laisse penser que la dimension intrinsèque de :

- **Swissroll** est 2
- **BrokenSwissroll** est 2
- **Helix** est 1
- **Twinpeaks** est 2

Les codes de calibration sont disponibles dans le fichier **Reduc_Dim_Calibration.R**

B. Utilisation des Méthodes

Dans cette partie nous allons calibrer et tester les différentes méthodes présentées plus haut

LLE

Il faut d'abord calibrer LLE en trouvant le K optimal du modèle . C'est le seul modèle ou la calibration est nécessaire.

Dataset	*Swissroll*	*BrokenSwissroll*	*Helix*	*Twinpeaks*
K Estimé	16	19	19	14

TABLE II

K optimal trouvé pour LLE

LLE

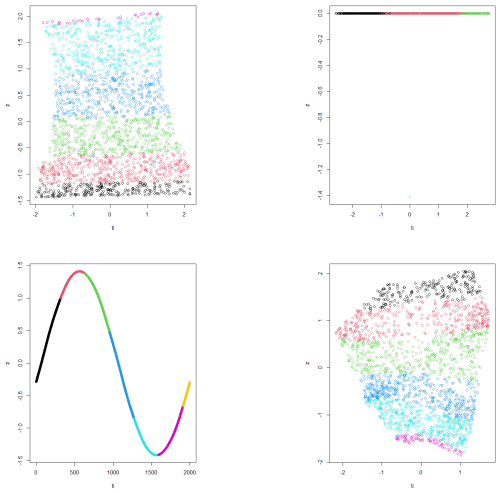


Fig. 11. Résultats obtenus avec LLE , En haut a Droite brokenswissroll, En Haut a Gauche Swissroll , en Bas a Droite Twinpeaks , En bas a Gauche Helix

Kernel PCA

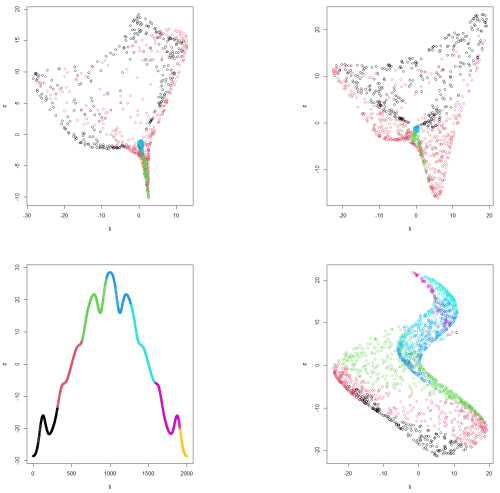


Fig. 13. Résultats obtenus avec KPCA , En haut a Droite brokenswissroll, En Haut a Gauche Swissroll , en Bas a Droite Twinpeaks , En bas a Gauche Helix

MDS

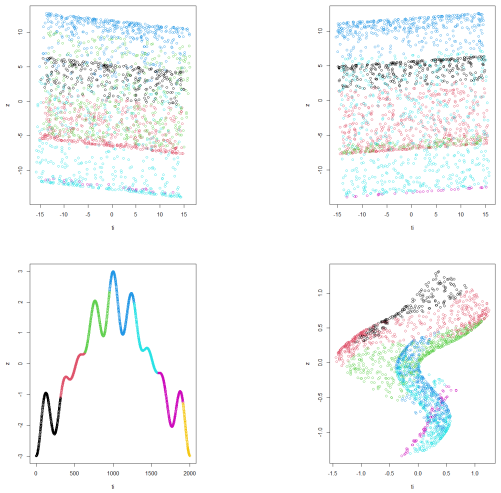


Fig. 12. Résultats obtenus avec MDS , En haut a Droite brokenswissroll, En Haut a Gauche Swissroll , en Bas a Droite Twinpeaks , En bas a Gauche Helix

Isomap

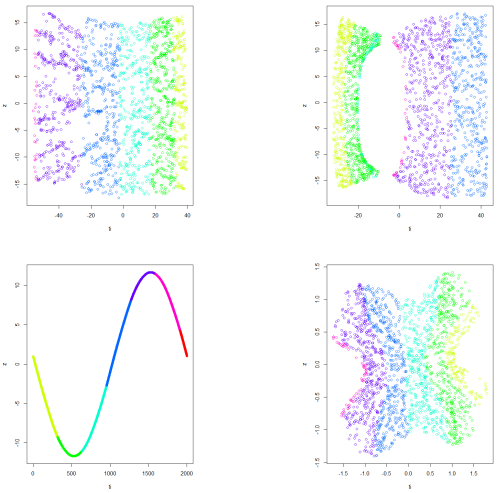


Fig. 14. Résultats obtenus avec Isomap , En haut a Droite brokenswissroll, En Haut a Gauche Swissroll , en Bas a Droite Twinpeaks , En bas a Gauche Helix

Juste en comparant les Outputs obtenus qualitativement: **Isomap** semble donner les meilleurs résultats de reconstruction , suivi de **LLE**

Les codes des Techniques sont disponibles dans le fichier **Reduc_Dim_Methodes.R**

VI. Comparaison des méthodes

Comparons maintenant les précédentes techniques avec comme méthode de comparaison : Le **1-NN** (1-Nearest Neighborhood ou 1-Plus-Proche-Voisin) plus précisément l'erreur de généralisation de ce modèle avec comme données d'apprentissage des points pris dans les dimensions reconstruites.

Les Résultats sont présentés dans les tableaux ci-après :

Les résultats de **Isomap** > résultats de **LLE** > résultats de **LAPLACIAN EIGENMAPS** > résultats de **Kernel PCA** > **résultats de MDS**

Les codes de comparaison sont disponible dans le fichier **Comparaison_ML.R**

LAPLACIAN EIGENMAPS

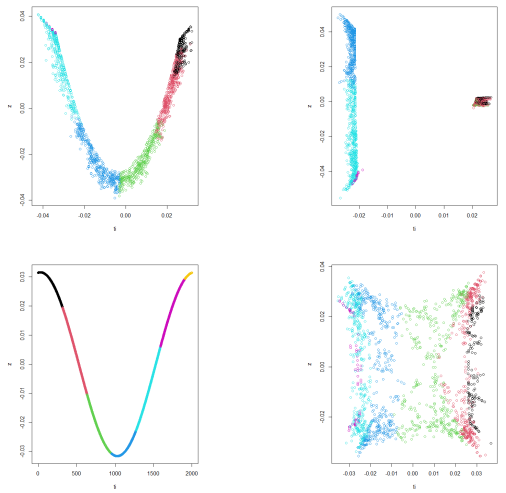


Fig. 15. Résultats obtenus avec LAPLACIAN EIGENMAPS , En haut a Droite brokenswissroll, En Haut a Gauche Swissroll , en Bas a Droite Twinpeaks , En bas a Gauche Helix

Dataset	*Swissroll*	*BrokenSwissroll*	*Helix*	*Twinpeaks*
Accuracy	0.952	0.938	0.196	0.968

TABLE III

ACCURACY DU 1-NN POUR LLE

Dataset	*Swissroll*	*BrokenSwissroll*	*Helix*	*Twinpeaks*
Accuracy	0.576	0.704	0.424	0.744

TABLE IV

ACCURACY DU 1-NN POUR MDS

Dataset	*Swissroll*	*BrokenSwissroll*	*Helix*	*Twinpeaks*
Accuracy	0.76	0.796	0.266	0.802

TABLE V

ACCURACY DU 1-NN POUR KERNEL PCA

Dataset	*Swissroll*	*BrokenSwissroll*	*Helix*	*Twinpeaks*
Accuracy	0.982	0.928	0.232	0.964

TABLE VI

ACCURACY DU 1-NN POUR ISOMAP

Dataset	*Swissroll*	*BrokenSwissroll*	*Helix*	*Twinpeaks*
Accuracy	0.932	0.854	0.16	0.892

TABLE VII

ACCURACY DU 1-NN POUR LAPLACIAN EINGENMAPS

VII. APPLICATION SUR DONNÉES RÉELLES

Appliquons maintenant nos méthodes de reduction de dimension sur MNIST.

Pour le dataset **MNIST** , seul PCA et Isomap ont été testé et comparés.

Nous avons d'abord choisi un échantillon de **2000** individus repartit en **1300** pour le train et **700** pour le test

Puis nous avons utilisés **MLE** pour determiner la dimension intrinsèque du dataset , nous avons obtenu comme résultat 13.

Pour **PCA** nous avons troncaturé le **13** premières dimensions renvoyées par la PCA et pour **Isomap** nous avions déjà les 13 bonnes dimensions.

Nous avons obtenu un résultat de **88,1**% avec le classifier 1-NN sur Isomap et **83,4**% sur PCA.

La raison pricipale qui nous a motivé a prendre MNIST est que ce dataset était exigé pour tester notre modèle développé pour le projet de **Model Based Learning** .

Nous avions donc pu tester notre algorithme sur le jeu de donnée avec des dimensions obtenus par réduction de dimension et les résultats sont les suivant :

Pour le projet de Model based Learning , pour réduire le nombre de variables du train , Nous avons choisi que les variables avec une **variance** > 12350 donc , il ne restait que **45 variables** / 785

Nous avions train sur 10000 individus et test sur 10000 individus les resultats de notre modèle était de :

```
    0    1    2    3    4    5    6    7    8    9
0 897    0   20   42    0   62   25    9    3    9
1   0 1019    3    0    1    0    2    9    2    4
2  38   36  834   33   14    7   33   34   39   26
3  10    2   30  775    0  129   20    1   95   24
4   3   30   48   18  927   19   16  150    9  265
5  17    2   18   57    0  601   28    1   25    6
6   5    5   16   13    1   12  807    2   13    3
7   3    0    6    0    1   10    1  754    4   13
8   7   38   45   63   20   48   26   30  773   10
9   0    3   12    9   18    4    0   38   11  649

Overall Statistics

               Accuracy : 0.8036
                 95% CI : (0.7957, 0.8113)
    No Information Rate : 0.1135
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7817
```

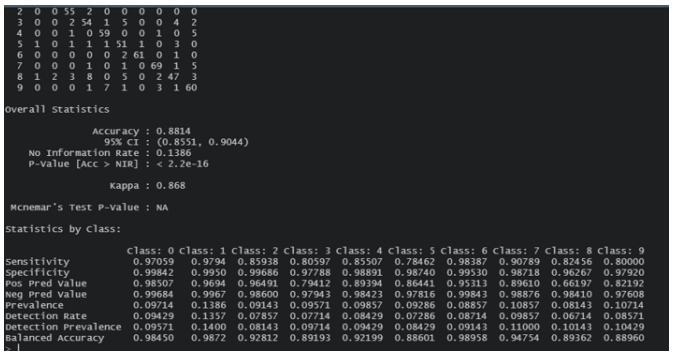
Fig. 16. Resultat de notre algo EM avec une sélection de variable par variance

Maintenant regardons les resultats obtenus avec isomap et 13 dimensions.

L'accuracy de notre modèle est passée de **80,16**% avec la selection de variable par variance a **88,14**% avec l'utilisation de Isomap avec 13 dimensions.

Cela montre la pertinence et l'utilité des méthodes de réduction de dimensions car nous sommes passé de **785 dimensions** a **13 dimensions** sans trop perdre en **accuracy**.

Les codes de Test sur MNIST sont disponible dans le fichier **Test_dataset_reel.R**



```
  2  0  0 55  2  0  0  0  0  0
  3  0  0  2 54  1  5  0  0  4  2
  4  0  0  1  0 59  0  0  1  0  5
  5  1  0  1  1  1 51  1  0  3  0
  6  0  0  0  0  0  2 61  0  1  0
  7  0  0  0  1  0  1  0 69  1  5
  8  1  2  3  8  0  5  0  2 47  3
  9  0  0  0  1  7  1  0  3  1 60

Overall Statistics

               Accuracy : 0.8814
                 95% CI : (0.8551, 0.9044)
    No Information Rate : 0.1386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.868

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8 Class: 9
Sensitivity           0.97059  0.9794  0.85938  0.80597  0.85507  0.78462  0.98387  0.90789  0.82456  0.80000
Specificity           0.99842  0.9950  0.99686  0.97788  0.98891  0.98740  0.99530  0.98718  0.96267  0.97920
Pos Pred Value        0.98507  0.9694  0.96491  0.79412  0.89394  0.86441  0.95313  0.89610  0.66197  0.82192
Neg Pred Value        0.99684  0.9967  0.98600  0.97943  0.98423  0.97816  0.99843  0.98876  0.98410  0.97608
Prevalence            0.09714  0.1386  0.09143  0.09571  0.09857  0.09286  0.08857  0.10857  0.08143  0.10714
Detection Rate        0.09429  0.1357  0.07857  0.07714  0.08429  0.07286  0.08714  0.09857  0.06714  0.08571
Detection Prevalence  0.09571  0.1400  0.08143  0.09714  0.09429  0.08429  0.09143  0.11000  0.10143  0.10429
Balanced Accuracy     0.98450  0.9872  0.92812  0.89193  0.92199  0.88601  0.98958  0.94754  0.89362  0.88960
>
```

Fig. 17. Resultat de notre algo EM avec isomap

REFERENCES

[1] Laurens van der Maaten et al., *Dimensionality Reduction: A Comparative Review*. TiCC TR 2009–005, 2009.