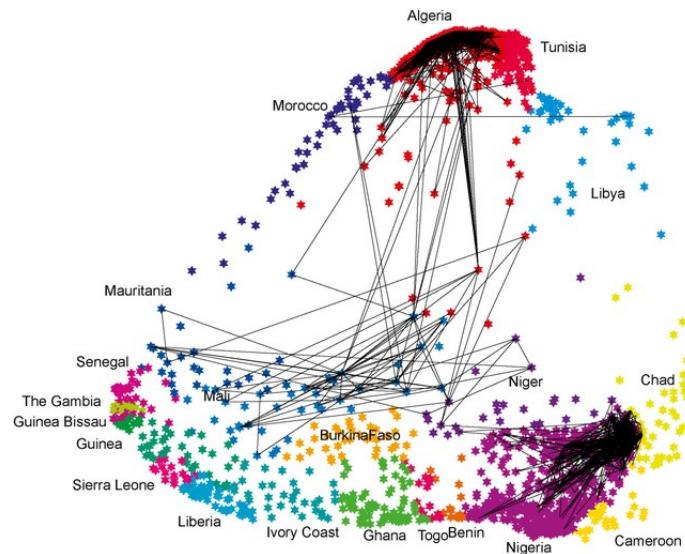

Spectral Embedding

Auteurs:

Adrien MEUNIER,
Julien VANRYSEL,
Selim LAKHDAR,
Yves SAIDO KIBUNDILA

Professeur:

Mohamed MAOUCHÉ



April 19, 2021

Contents

1	Introduction	2
2	Présentation de l'algorithme	2
2.1	Manifold algorithm	2
2.2	Laplacien graph	2
2.3	Eigenvalue	3
2.4	Utilisation avec sklearn	3
3	Fonctionnement interne de l'algorithme	3
3.1	Construction de la matrice d'adjacence	4
3.2	Choix des poids	4
3.3	Dernière étape: Eigenmaps	4
4	Avantages et incovénients	6
4.1	Avantages	6
4.2	Incovénients	7
5	Comparaison avec d'autres méthodes de réduction de dimensions	7
5.1	Datasets Générique	8
5.1.1	Observations: 2 Composantes	11
5.1.2	Observations: 1 Composante	13
5.2	Datasets Réels	14
5.2.1	Observations: 2 Composantes	16
5.2.2	Observations: 1 Composante	18
6	Evaluation	19
6.1	Gen Datasets	19
6.1.1	Sans Reduction	19
6.1.2	Reduction PCA	20
6.1.3	Reduction LDA	20
6.1.4	Reduction t-SNE	20
6.1.5	Reduction Spectral Embedding	21
6.1.6	Reduction Spectral Embedding RBF	21
6.2	Datasets Réels	21
6.2.1	Sans Reduction	21
6.2.2	Reduction PCA	22
6.2.3	Reduction LDA	22
6.2.4	Reduction t-SNE	23
6.2.5	Reduction Spectral Embedding	23
6.2.6	Reduction Spectral Embedding RBF	23
7	Conclusion	24

1 Introduction

2 Présentation de l'algorithme

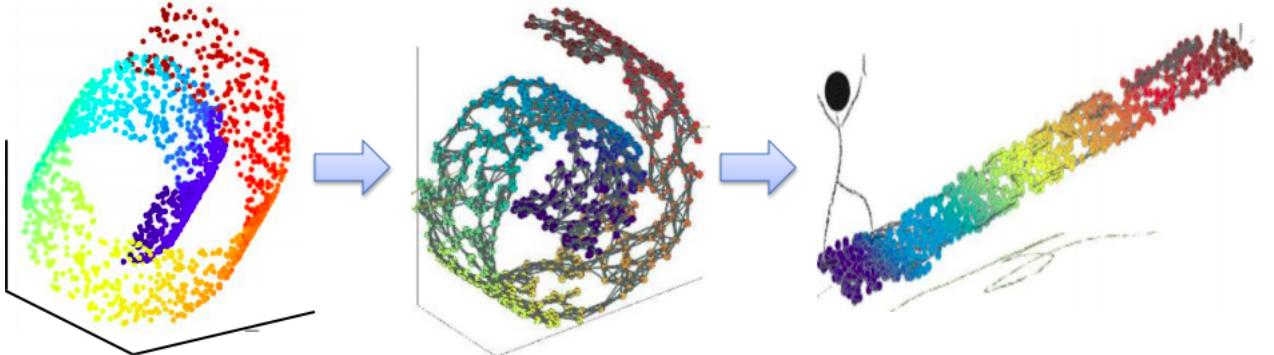


Figure 1: Une image vaut mille mots. résumé de l'algorithme en image

Nous allons présenter l'algorithme qui est le sujet même de ce rapport, voir l'intuition de cet algorithme sans entrer dans les aspects plus mathématique qui régissent cet algorithme. Pour pour voir faciliter la compréhension de cet algorithme nous allons aussi présenter quelques notions complémentaire et afin nous verrons comment utiliser efficacement cet algorithme avec la bibliothèque sklearn (Choisir les bons hyper-paramètres).

2.1 Manifold algorithm

Pour certaines données (non linéaire) les algorithmes tels que le PCA, LDA n'arrivent pas à séparer les données, ils échouent. Il y a une famille d'algorithme qu'on appelle **manifold algorithm** qui arrivent à séparer les données non linéaires. Le spectral embedding fait partie de cette famille, c'est un algorithme de réduction de dimensionnalité non linéaire. Ces algorithmes transforme les données dans un espace de grande dimension qui permet aux méthodes linéaires de séparer ces données dans ce nouvel espace. Ils sont en quelque sorte une extension des méthodes linéaire.

Le spectral embedding utilise un graphe laplacien pour faire sa réduction de dimension en utilisant une décomposition en vecteurs propres sur ces graphes. On peut alors choisir le nombre de vecteurs propres qu'on souhaite garder. Le graphe laplacien est construit à partir d'un graphe de similarité ou d'un graphe de plus proches voisins, ou d'une fonction kernel.

Il faut noter que spectral embedding cherche à ce que les points proches en grande dimension soient aussi en petite dimension. Ce résultat est garanti grâce à la minimisation d'une fonction coût.

2.2 Laplacien graph

Comment est représenté exactement le graphe laplacien ? Comme nous l'avons dit plus tôt, nous construisons tout d'abord le graphe de similarité. Chaque exemple que nous avons dans notre dataset constitue un sommet, et à partir de ces sommets on utilise un algorithme tel que k plus proches voisins (ou un kernel).

Egalement, la matrice laplacienne peut être vue comme une forme quadratique caractérisant la régularité d'un vecteur $x \in R^n$ au regard de la distance définie par le graphe, en effet on a la formule suivante: $x^T L x = \sum_{(u,v) \in E} (x(u) - x(v))^2$.

2.3 Eigenvalue

Une fois que la matrice laplacienne construite nous pouvons chercher les vecteurs propres de cette matrice. Ce sont ces vecteurs qu'on appelle eigenvalue, on peut alors grader le nombre de vecteur qui nous intéressent pour notre réduction de dimension.

2.4 Utilisation avec sklearn

```
class sklearn.manifold.SpectralEmbedding(n_components=2, *, affinity='nearest_neighbors', gamma=None,  
random_state=None, eigen_solver=None, n_neighbors=None, n_jobs=None)
```

La bibliothèque sklearn est une des plus utilisées en machine learning, nous allons voir comment utiliser efficacement spectral embedding. Voyons quelques paramètres importants à régler :

- **n_component** : C'est le nombre de composantes qu'on souhaite pour la réduction de dimension
- **affinity** : Ce paramètre permet de définir de quelle manière le graph laplacien est construit
 - **nearest_neighbors**: construit le graphe laplacien à calculant d'abord le graph de plus proche voisin.
 - **rbf** : construit le graphe laplacien à partir de la fonction kernel RBF
 - **precomputed**: le graphe laplacien n'est pas calculé, X est considéré comme graphe laplacien.
 - **precomputed_near**: le graphe laplacien est calculé directement à partir de X qui est considéré comme graphe de similarité.
 - **callable**: On peut passer une fonction qui sera appliquée à X pour calculer le graphe laplacien.
- **gamma**: C'est le coefficient de la fonction kernel RBF.
- **eigen_solv**: La stratégie à utiliser pour effectuer la décomposition en vecteur propre.
- **n_neighbors**: Nombre de voisins à considérer pour la construction du graph de plus proche voisin.

Dans la section suivante nous allons voir en détails les aspects mathématiques derrière cet algorithme.

3 Fonctionnement interne de l'algorithme

Sklearn implémente des cartes propres laplacianes ce qui permet de récupérer les zones qui sont intéressantes pour appliquer l'algorithme de spectral embedding dessus.

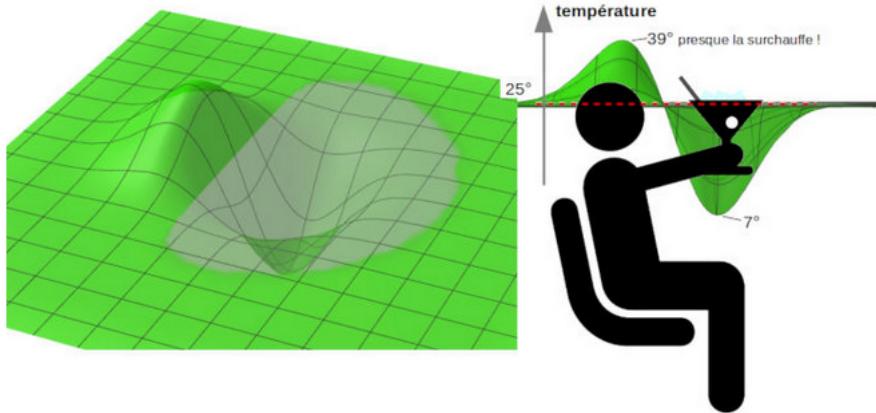


Figure 2: Exemple

Ici on a un exemple de carte propre laplacienne. Imaginez-vous au soleil avec un cocktail (image de droite), on distingue 2 sources de température, une au niveau du cocktail (7°) et une au niveau de la tête (39°). A gauche on a la représentation sous forme de carte propre laplacienne de la situation. On peut voir qu'il y a deux variations dans la carte qui correspondent aux 2 températures. On va pouvoir récupérer ces 2 zones, qui correspondent aux endroits où les données sont intéressantes.

Pour l'algorithme de spectral embedding qui est Laplacian Eigenmaps il comporte 3 étapes :

3.1 Construction de la matrice d'adjacence

On relit les noeuds i et j si leurs valeurs sont proches, 2 variations possibles :

- Les noeuds i et j sont connectés si $|x_i - x_j|^2 < \epsilon$:
 - Avantage : la relation est naturellement symétrique
 - Désavantage : Conduit souvent à des graphiques avec plusieurs composants connectés.
 - Difficulté à choisir ϵ .
- n plus proches voisins :Les noeuds i et j sont reliés par une arête si i est parmi les n voisins les plus proches de j , ou si j est parmi les n voisins les plus proches de i .
 - Avantage : plus facile à choisir. N'a pas tendance à conduire à des graphes déconnectés
 - Désavantage : moins intuitif sur le plan géométrique

3.2 Choix des poids

Ici aussi, nous avons deux variantes de pesée des bords :

- Si les noeuds i et j sont connectés mettre : $W_{ij} = \frac{\exp(-|X_i - X_j|^2)}{t}$, $t \in R$ sinon $W_{ij} = 0$.
- $W_{ij} = 1$ si les sommets i et j sont reliés par une arête et $W_{ij} = 0$ si les sommets i et j ne sont pas reliés par une arête. Cette simplification évite d'avoir à choisir

3.3 Dernière étape: Eigenmaps

Avant d'expliquer l'algorithme, je vous explique comment construire une matrice laplacienne, cela va nous être utile pour l'algorithme.

$$L_{i,j} := \begin{cases} \deg(s_i) & \text{si } i = j \\ -1 & \text{si } i \neq j \text{ et si } i \text{ et } j \text{ sont reliés par une arête} \\ 0 & \text{sinon} \end{cases}$$

Figure 3: Formule pour calculer le graph laplacien

La matrice laplacienne est construite grâce à l'image juste au-dessus. On va d'abord construire la matrice de degrés et d'adjacence du graph. Une fois ces matrices construites on va pouvoir construire la matrice laplacienne. On va avoir une double boucle qui va remplir la matrice au fur et à mesure, cette matrice est de la même taille que la matrice d'adjacence ou de degrés. Lors de ce remplissage si i et j (les valeurs de nos compteurs des 2 boucles) sont égaux alors on va mettre à la position i,j la valeur dans la matrice de degré.

Si i et j sont différents et qu'on a un 1 dans la matrice d'adjacence à cette position (donc que les sommets i et j sont reliés par une arête) on va mettre -1. Sinon on met 0.

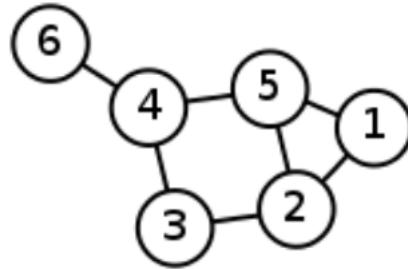


Figure 4: Un graph

Matrice de degrés	Matrice d'adjacence
$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$

Matrice Laplacienne

$$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

Figure 5: Exemple du calcul du graph laplacien

Voici un petit exemple de matrice laplacienne. Au départ nous avons le graph avec sa matrice de degrés et sa matrice d'adjacence et au final on obtient la matrice laplacienne. On peut voir en 0,0 par exemple qu'on a la valeur 2 puisque i et j sont égaux et que dans la matrice de degrés en 0,0 on a la valeur 2.

La troisième et dernière étape concerne l'algorithme de Spectral embedding.

On calcule les valeurs propres et vecteurs propres de la matrice laplacienne :

$$Lf = \lambda Df$$

où D est la matrice de poids diagonale, et ses entrées sont des sommes de colonne (ou de ligne, puisque W est symétrique) de W, $D_{ii} = \sum_j W_{ji}$.

La matrice laplacienne : $L = D - W$. (W est la matrice de poids)

On peut également normaliser notre graph ce qui devient : $D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$

Soit f_0, \dots, f_{n-1} les solutions de l'équation 2.1, ordonnées selon leurs valeurs propres : Nous omettons le vecteur propre f_0 correspondant à la valeur propre 0 et utilisons les vecteurs suivants pour incorporer l'espace euclidien en m dimensions:

$$x_i \rightarrow (f_1(i), \dots, f_m(i))$$

Après avoir enlevé le premier vecteur et valeur propre, on va utiliser la formule juste au-dessus pour effectuer notre réduction de dimensions. Le paramètre m va nous être utilisé maintenant car il va permettre de savoir combien de dimensions on garde. Si par exemple on a choisi m = 5, on va garder les vecteurs et valeurs propres $(f_1, f_2, f_3, f_4, f_5)$ pour les incorporer dans l'espace euclidien. Après cette incorporation nous avons nos données en 5 dimensions.

4 Avantages et incovénients

4.1 Avantages

Un des avantages de Spectral Embedding est une complexité plus faible que les autres méthodes de Manifold pour des jeux de données de grande taille..

La complexité globale de spectral embedding est : $O[D \cdot \log(k) \cdot N \cdot \log(N)] + O[D \cdot N \cdot k^3] + O[d \cdot N^2]$

Mais nous pouvons diviser cette complexité dans les 3 parties de cet algorithme:

1. Construction de graphe de poids : $O[D \cdot \log(k) \cdot N \cdot \log(N)]$
2. Construction du graphe Laplacian : $O[D \cdot N \cdot k^3]$
3. Décomposition en vecteur propre partiel : $O[d \cdot N^2]$

Sachant que :

N : nombre de données d'entraînement

D : dimension d'entrée

k : nombre de voisins les plus proches

d : dimension de sortie

Les différentes méthodes de Manifold possèdent la partie de construction de graphes d'adjacence et de décomposition en vecteur propre. Nous allons alors la partie centrale de ces méthodes. Comme vu ci-dessus, la complexité du graphe de Laplacian est de $O[D \cdot N \cdot k^3]$. C'est-à-dire que l'algorithme de construction de graphe laplacienne nécessite un seul parcours de la matrice des données ($D \cdot N$).

Dans les autres méthodes du manifold, plusieurs parcours des données d'entrée sont nécessaires. Son prédecesseur **Hessian Eigenmapping** possède lui une complexité plus importante de l'ordre $O[D \cdot N \cdot k^3 + N \cdot d^6]$ avec d étant la dimension de sortie. Son successeur **Isomap** possède une complexité de $O[N^2(k + \log(N))]$ qui dans le cas des grands jeux de données est plus grand que pour Spectral Embedding.

Cette différence apporte une meilleure complexité que les autres méthodes de réduction de dimension. Cependant malgré ces meilleures performances au point de vue de la complexité, le temps d'exécution de Spectral Embedding n'est pas équivalent à celui PCA.

4.2 Incovénients

Un des inconvénients de cette méthode est sa sensibilité au bruit. En effet, avec le calcul de la matrice d'adjacence, les bruits sont pris en compte par défaut. Dans le cas de classification où l'on utilise le Spectral Embedding à faible dimension sur des données bruitées non uniformément, nous obtiendrons des résultats non satisfaisants.

Ce résultat est obtenu car cette méthode conserve les distances inter-points.

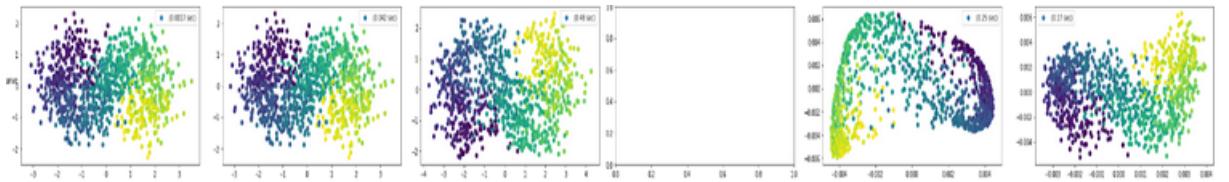


Figure 6: donnée S-curve datasets avec un bruit de 0.5 sur les méthodes PCA, KernelPCA, Hessian Eigenmapping,Isomap et Spectral Embedding avec ces deux matrices d'adjacence différentes

Cette méthode conserve les distances locales mais ne conserve pas la structure initiale de la donnée. Ce choix peut être un inconvénient pour la visualisation de jeu de données

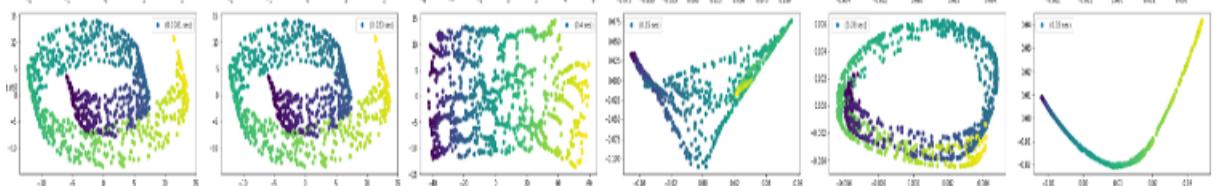


Figure 7: données Swiss roll datasets sur les méthodes PCA, KernelPCA,Hessian Eigenmapping,Isomap et Spectral Embedding avec ces deux matrices d'adjacence différentes

5 Comparaison avec d'autres méthodes de réduction de dimensions

Dans cette partie nous allons comparer **Spectral Embedding** à **LDA**, **PCA** et **t-SNE** qui sont des méthodes de réduction de dimension. On notera ici que LDA est utilisé en tant que méthode de réduction et non de classification.

Avant de nous lancer dans la comparaison des résultats, le tableau suivant résume les aspects les plus importants de ces méthodes.

Résumé des méthodes			
PCA	LDA	t-SNE	Spectral Embedding
Non Supervisé	Supervisé	Non Supervisé	Non Supervisé
Méthode Linéaire	Méthode Linéaire	Méthode Non Linéaire	Méthode Non Linéaire
Préserve les structures globales	Maximise la séparabilité des classes	Préserve les structures locales	Préserve les distances locales

Figure 8: Résumé des Méthodes

Par la suite nous allons tester ces trois méthodes sur des jeux de données générique et réels.

5.1 Datasets Générique

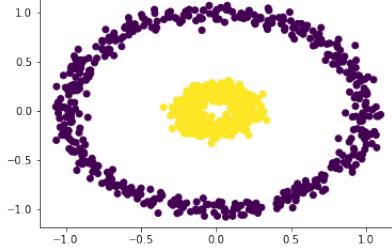
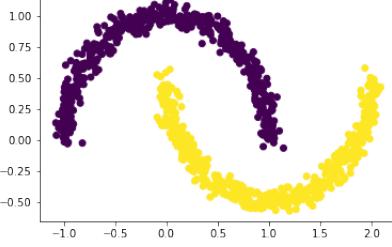
Dataset	Observation	Remarques
Circles		Dataset généré grâce à la fonction <code>make_circles</code> avec les paramètres <code>noise=0.05</code> et <code>factor=.2</code>
Moon		Dataset généré grâce à la fonction <code>make_moons</code> avec le paramètre <code>noise=.05</code>

Table 1: Gen Datasets

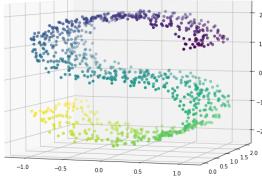
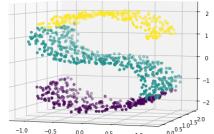
Dataset	Observation	Remarques
S Curve 1		Dataset généré grâce à la fonction make_s_curve avec le paramètre <i>noise=.05</i> . À noter que les labels (y) sont continues et non discrets !
S Curve 2		Dataset généré grâce à la fonction make_s_curve avec le paramètre <i>noise=.05</i> .

Table 2: Gen Datasets

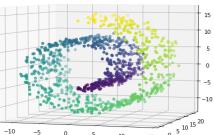
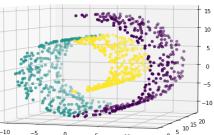
Dataset	Observation	Remarques
Swiss Roll 1		Dataset généré grâce à la fonction make_swiss_roll avec le paramètre <i>noise=.05</i> . À noter que les labels (y) sont continues et non discrets !
Swiss Roll 2		Dataset généré grâce à la fonction make_swiss_roll avec le paramètre <i>noise=.05</i> .

Table 3: Gen Datasets

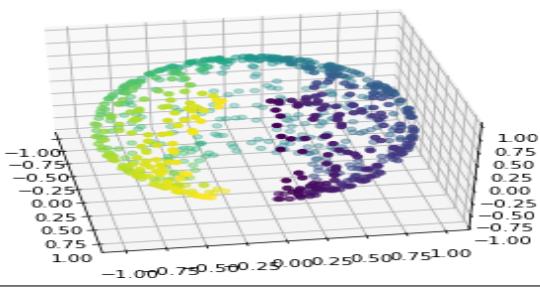
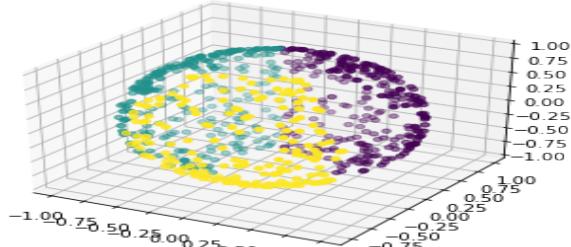
Dataset	Observation	Remarques
Sphere 1		
Sphere 2		À noter que les labels (y) sont continue et non discret !

Table 4: Gen Datasets

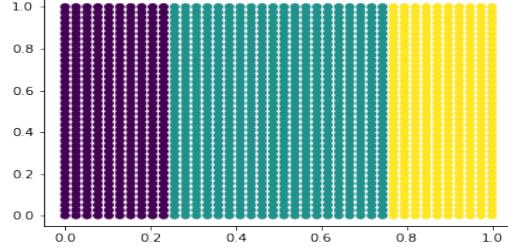
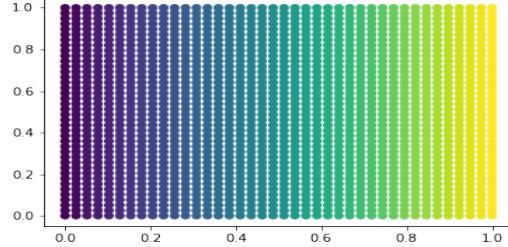
Dataset	Observation	Remarques
Grid 1		Dataset généré grâce à la fonction MeshGrid .
Grid 2		Dataset généré grâce à la fonction MeshGrid . À noter que les labels (y) sont continues et non discrets !

Table 5: Gen Datasets

5.1.1 Observations: 2 Composantes

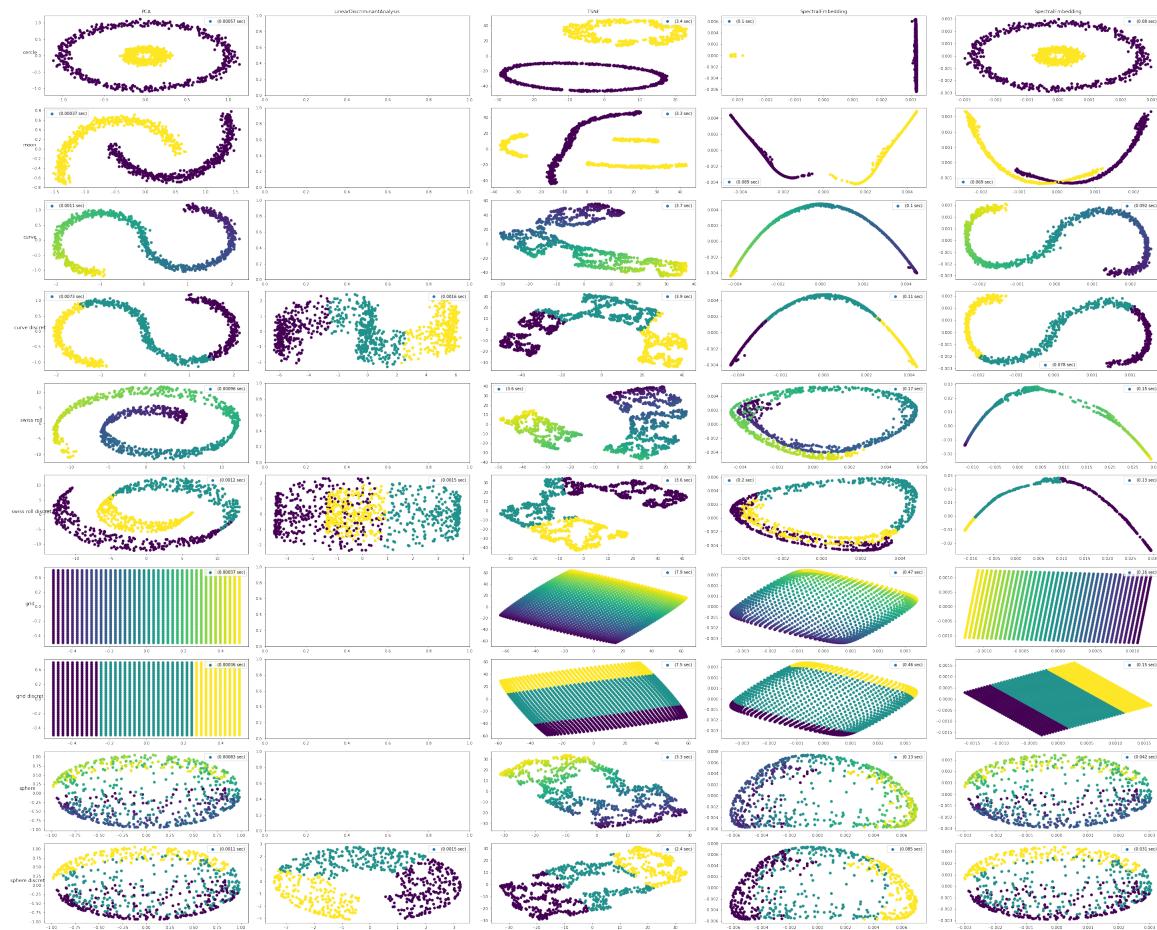


Figure 9: Résultats des différentes méthodes avec 2 compostanes

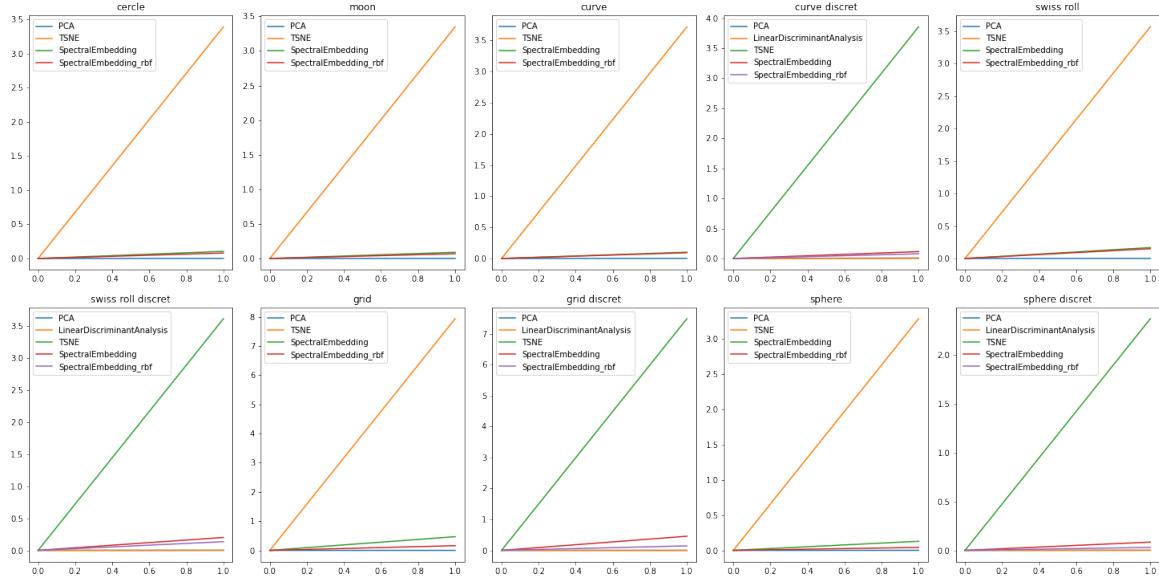


Figure 10: Temps d'exécution

D'après le graphique nous pouvons observer :

- Que dans le cas (nombre de classe > nombre de composants), l'utilisation du LDA est impossible ainsi que le cas des données aux étiquettes continues.
- Dans le cas où les données possèdent la même dimension en entrée quand sortie, PCA conserve la forme de la donnée. Cependant ce n'est pas le cas pour t-SNE, les données transformées sont devenus séparables linéairement.
- Dans le cas des données ayant une dimension de plus, on remarque l'apparition de groupes l'un a conservé plus ou moins la forme de la donnée (PCA) et l'autre non (LDA, TSNE, Spectral Embedding).
- Dans le dataset digit, TSNE est le seul à nous donner des classes bien distantes permettant une meilleure future classification.
- Le t-SNE déstructure la donnée pour mieux la regrouper à la différence de PCA qui conserve aux plus la donnée.

D'après le graphique du temps d'exécution, on peut observer que la méthode t-SNE prend plus de temps que les autres fonctions. Cela est dû au calcul des différents voisins qui prennent du temps.

5.1.2 Observations: 1 Composante

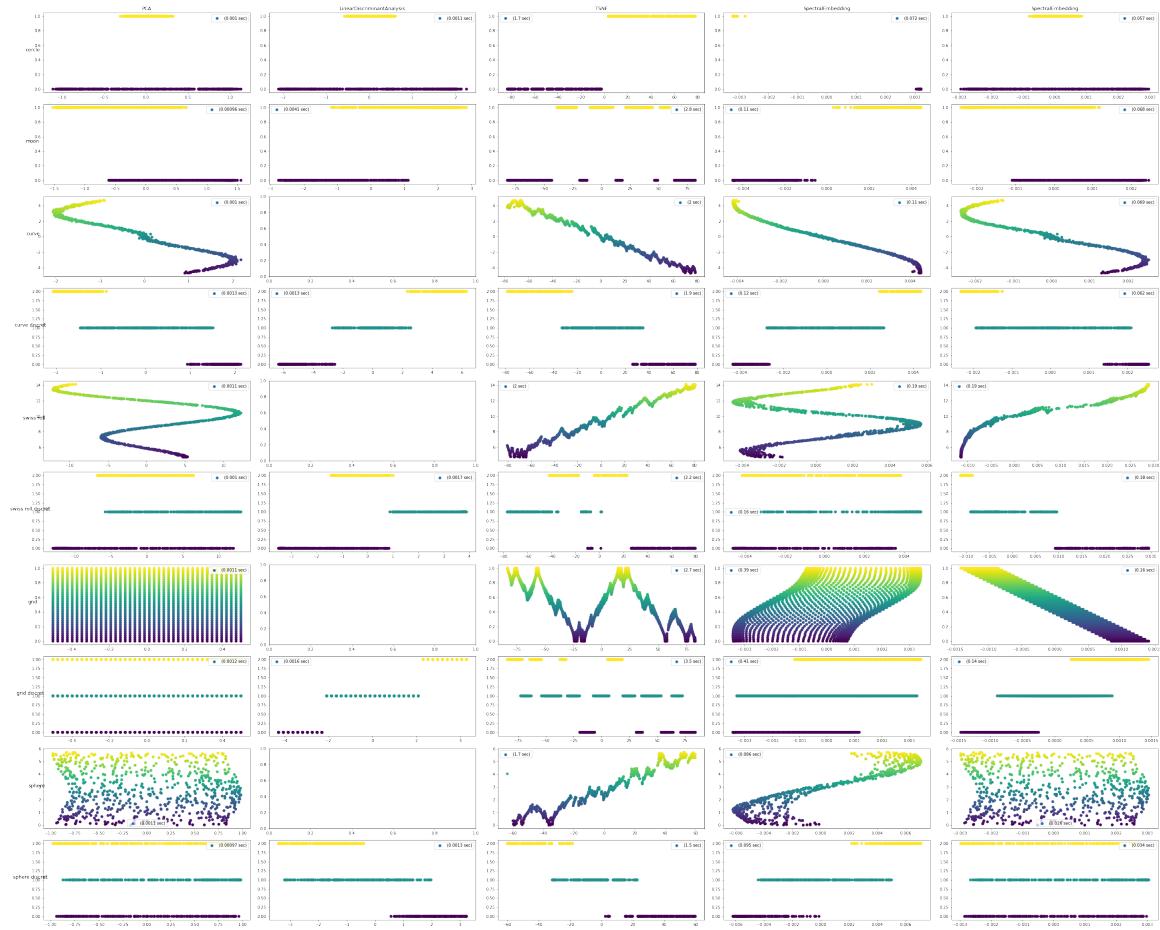


Figure 11: Résultats des différentes méthodes

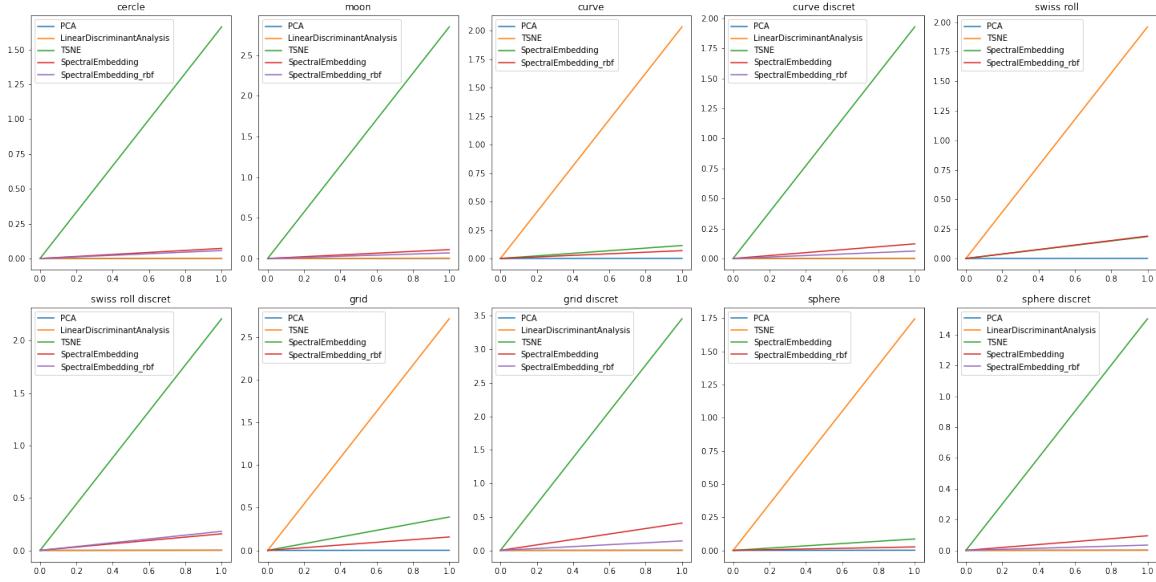


Figure 12: Temps d'exécution

D'après le graphique nous pouvons observer :

- On remarque que le cas de la conservation d'une seule component et dans le cas où les labels ne sont pas continues, nous avons une séparation distante de chaque classes.
- Abaisser la dimension des données, peut être une bonne pratique pour simplifier la classification et le clustering.
- t-SNE est la méthode la plus lente.

5.2 Datasets Réels

Dans cette partie nous allons appliquer nos méthodes sur un dataset réel. Pour cela nous allons utiliser le dataset digits et iris.

Dataset	Observation	Remarques
Iris	<p>A scatter plot showing the relationship between petal length (y-axis, 1 to 7) and sepal length (x-axis, 4.5 to 8.0). The data points are color-coded by species: setosa (blue), versicolor (orange), and virginica (green). The plot shows clear separation between the species based on these two measurements.</p>	Le dataset reste assez simple.
Digits_all	<p>A 28x28 grid of handwritten digits from 0 to 9, each enclosed in a small square box. This represents the full Digits dataset.</p>	Ici nous prenons le dataset en entier. Les chiffres de 0 à 9.
Digits_6	<p>A 28x28 grid of handwritten digits from 0 to 6, each enclosed in a small square box. This represents a subset of the Digits dataset containing only digits 0 through 6.</p>	Ici nous prenons que les chiffres de 0 à 6
Digits_4	<p>A 28x28 grid of handwritten digits from 0 to 4, each enclosed in a small square box. This represents a subset of the Digits dataset containing only digits 0 through 4.</p>	Ici nous prenons que les chiffres de 0 à 4

Table 6: Real Datasets

5.2.1 Observations: 2 Composantes

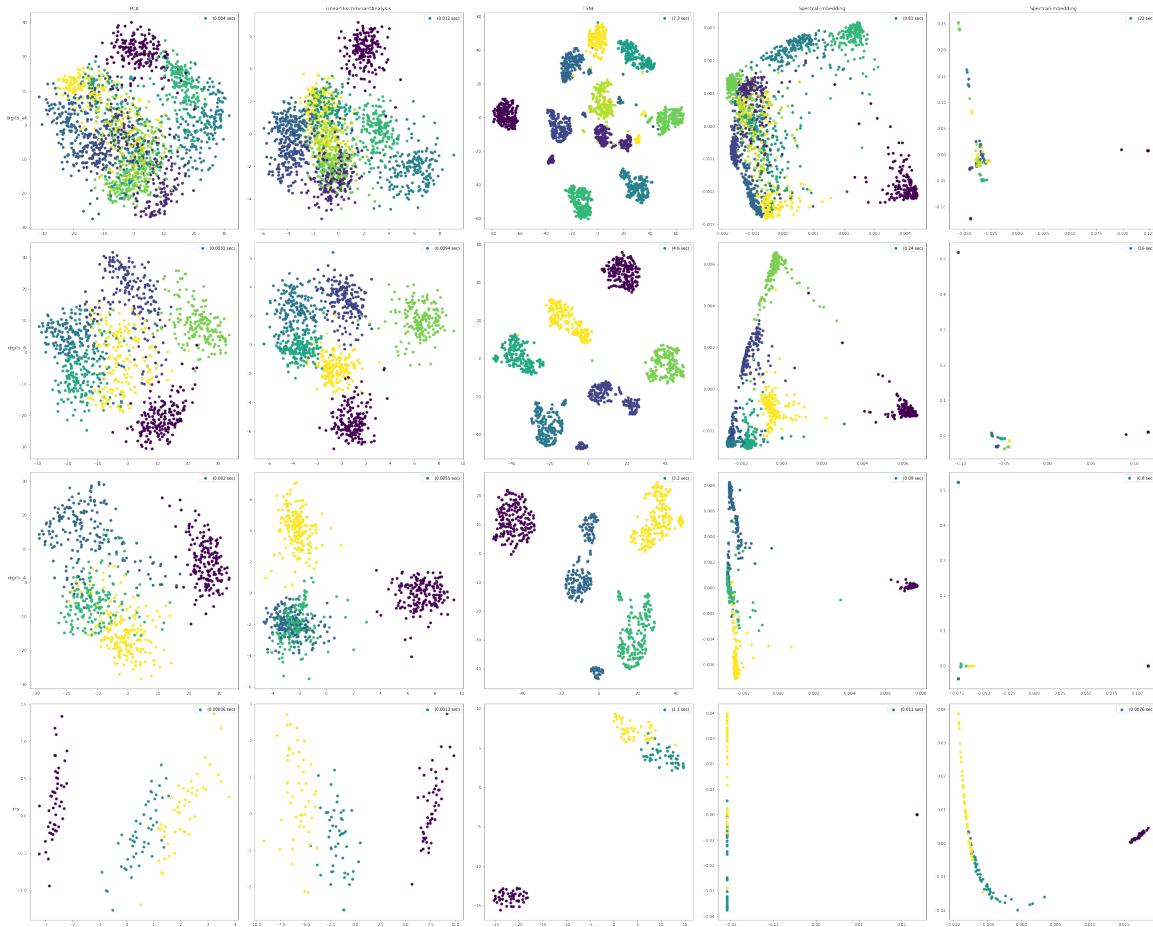


Figure 13: Résultats des différentes méthodes avec 2 composantes

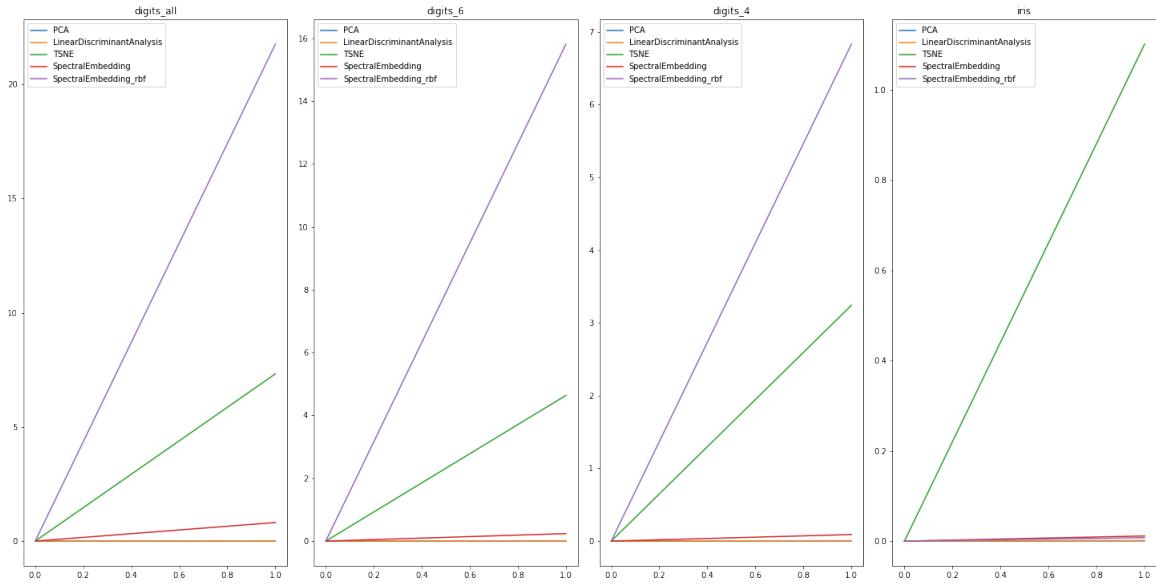


Figure 14: Temps d'exécution

D'après le graphique nous pouvons observer :

- t-SNE arrive le mieux à séparer les données
- Spectral Embedding prend énormément de temps pour au final avoir un résultat moins précis que t-SNE. (À part pour le dataset Iris qui reste assez simple.)

5.2.2 Observations: 1 Composante



Figure 15: Résultats des différentes méthodes avec 1 composante

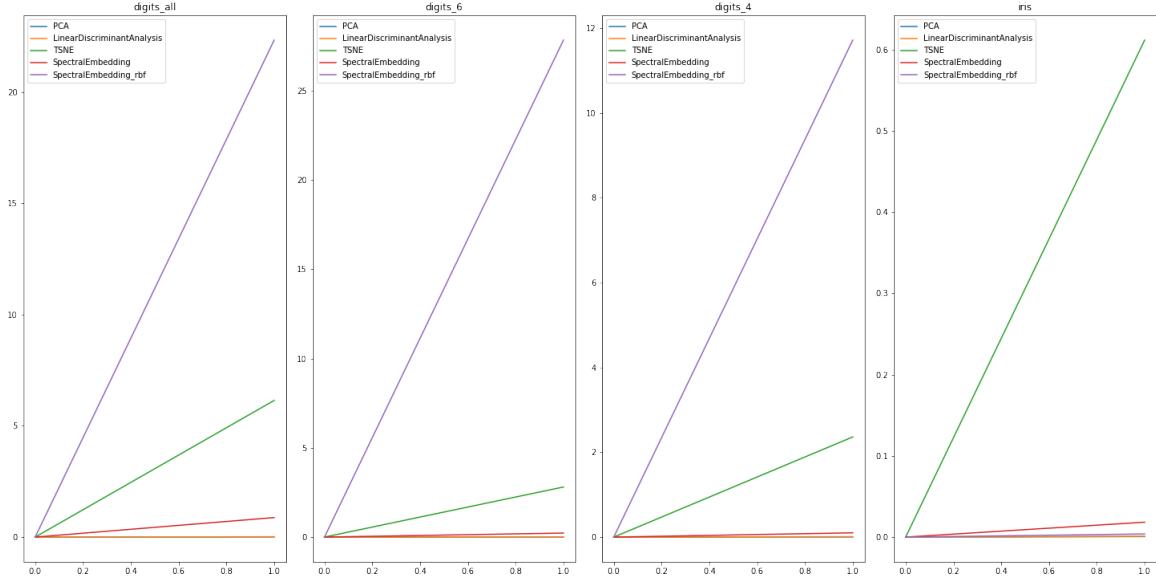


Figure 16: Temps d'exécution

D'après le graphique nous pouvons observer :

- La représentation en 1D est beaucoup plus clair sur tout les datasets.
- Spectral Embedding prend énormément de temps pour au final avoir un résultat plus ou moins équivalent que t-SNE. (À part sur le dataset Iris).

6 Evaluation

Dans cette partie nous allons évaluer les différentes méthodes présentées en utilisant plusieurs classifiants et avec de la cross validation sur 3 folds pour éviter d'overfitter.

6.1 Gen Datasets

6.1.1 Sans Reduction

	cercle	moon	scurve	swiss_roll	grid	sphere
KNN	1.00	1.00	0.97	0.99	0.75	0.97
SVC_lin	0.67	0.88	0.97	0.85	0.73	0.98
SVC	1.00	1.00	0.98	0.99	0.76	0.98
DT	1.00	0.99	0.98	1.00	0.78	0.96
RF	1.00	0.99	0.98	0.98	0.77	0.97
MLP	1.00	0.97	0.99	0.99	0.75	0.99
AdaBoost	1.00	1.00	0.76	0.62	0.78	0.90
GNB	1.00	0.88	0.92	0.99	0.80	0.97
QDA	1.00	0.88	0.99	0.99	0.79	0.99

Figure 17: Scores Classification sans reduction

On peut remarquer ici que les datasets génériques ne sont pas très complexe et que sans réduction nous arrivons déjà à obtenir un bon score de classification (à part pour le dataset Grid dans certains cas).

6.1.2 Reduction PCA

	cercle	moon	scurve	swiss_roll	grid	sphere
KNN	1.00	1.00	0.98	0.92	0.75	0.74
SVC_lin	0.56	0.86	0.98	0.71	0.67	0.71
SVC	1.00	1.00	0.98	0.91	0.75	0.73
DT	0.99	0.99	0.97	0.91	0.80	0.72
RF	0.99	0.99	0.98	0.91	0.80	0.73
MLP	1.00	0.97	0.98	0.90	0.75	0.72
AdaBoost	1.00	1.00	0.61	0.76	0.80	0.48
GNB	1.00	0.86	0.90	0.81	0.80	0.72
QDA	1.00	0.88	0.98	0.83	0.79	0.73

Figure 18: Scores Classification avec reduction PCA

On remarque que la réduction PCA dégrade un peu les scores sur les datasets swiss_roll, grid et sphère. À part pour quelques exceptions (DecisionTree (DT)).

6.1.3 Reduction LDA

	cercle	moon	scurve	swiss_roll	grid	sphere
KNN	NaN	NaN	0.98	0.90	0.75	0.98
SVC_lin	NaN	NaN	0.98	0.84	0.75	0.98
SVC	NaN	NaN	0.98	0.91	0.80	0.98
DT	NaN	NaN	0.98	0.92	0.80	0.97
RF	NaN	NaN	0.98	0.92	0.80	0.98
MLP	NaN	NaN	0.98	0.90	0.78	0.99
AdaBoost	NaN	NaN	0.98	0.87	0.80	0.89
GNB	NaN	NaN	0.99	0.91	0.80	0.98
QDA	NaN	NaN	0.98	0.91	0.80	0.98

Figure 19: Scores Classification avec reduction LDA

On remarque que la réduction LDA n'améliore pas significativement les scores.

6.1.4 Reduction t-SNE

	cercle	moon	scurve	swiss_roll	grid	sphere
KNN	1.00	1.00	0.98	0.98	0.75	0.98
SVC_lin	1.00	1.00	0.98	0.98	0.78	0.98
SVC	0.99	0.99	0.78	0.95	0.50	0.82
DT	1.00	1.00	0.98	0.97	0.78	0.97
RF	1.00	1.00	0.99	0.98	0.76	0.97
MLP	1.00	1.00	0.98	0.99	0.74	0.98
AdaBoost	1.00	1.00	0.96	0.95	0.61	0.85
GNB	1.00	1.00	0.97	0.97	0.79	0.93
QDA	1.00	1.00	0.96	0.97	0.79	0.97

Figure 20: Scores Classification avec reduction t-SNE

On remarque que la réduction t-SNE améliore significativement les scores.

6.1.5 Reduction Spectral Embedding

	cercle	moon	scurve	swiss_roll	grid	sphere
KNN	1.00	1.00	0.98	0.94	0.75	0.87
SVC_lin	0.67	0.66	0.43	0.35	0.50	0.36
SVC	0.67	0.66	0.43	0.35	0.50	0.36
DT	1.00	1.00	0.98	0.87	0.79	0.87
RF	1.00	1.00	0.98	0.87	0.75	0.86
MLP	0.50	0.50	0.43	0.34	0.50	0.36
AdaBoost	1.00	1.00	0.98	0.66	0.51	0.76
GNB	1.00	1.00	0.99	0.76	0.73	0.80
QDA	1.00	1.00	0.97	0.77	0.84	0.85

Figure 21: Scores Classification avec reduction SPE

6.1.6 Reduction Spectral Embedding RBF

	cercle	moon	scurve	swiss_roll	grid	sphere
KNN	1.00	0.99	0.98	0.99	0.75	0.75
SVC_lin	0.56	0.58	0.43	0.35	0.50	0.36
SVC	0.56	0.58	0.43	0.68	0.50	0.36
DT	0.99	0.91	0.98	0.99	0.73	0.73
RF	1.00	0.92	0.98	0.99	0.74	0.74
MLP	0.50	0.50	0.43	0.50	0.50	0.36
AdaBoost	1.00	0.92	0.49	0.98	0.52	0.55
GNB	1.00	0.75	0.89	0.94	0.73	0.72
QDA	1.00	0.73	0.98	0.98	0.79	0.73

Figure 22: Scores Classification avec reduction SPE_rbf

On remarque que les réductions SPE et SPE_rbf n'améliorent pas tous les scores comme t-SNE. On observe même une dégradation des scores sur les datasets grid et sphere.

6.2 Datasets Réels

6.2.1 Sans Reduction

	digits_all	digits_6	digits_4	iris
KNN	0.94	0.97	0.98	0.94
SVC_lin	0.95	0.98	0.98	0.89
SVC	0.10	0.17	0.26	0.95
DT	0.64	0.83	0.88	0.96
RF	0.77	0.87	0.93	0.95
MLP	0.95	0.97	0.97	0.97
AdaBoost	0.26	0.48	0.59	0.95
GNB	0.78	0.87	0.85	0.94
QDA	0.80	0.86	0.88	0.97

Figure 23: Scores Classification sans reduction

On observe ici que le dataset Iris est assez simple, on obtient de bon scores sur plupart des classificateurs. Par contre sur le dataset digits les scores varient.

6.2.2 Reduction PCA

	digits_all	digits_6	digits_4	iris
KNN	0.60	0.80	0.83	0.97
SVC_lin	0.60	0.83	0.84	0.91
SVC	0.50	0.66	0.69	0.97
DT	0.58	0.78	0.83	0.95
RF	0.62	0.82	0.85	0.93
MLP	0.63	0.83	0.84	0.97
AdaBoost	0.34	0.48	0.64	0.85
GNB	0.62	0.81	0.86	0.91
QDA	0.63	0.81	0.85	0.97

Figure 24: Scores Classification avec reduction PCA

On remarque que la réduction PCA dégrade les scores sur les datasets digits.

6.2.3 Reduction LDA

	digits_all	digits_6	digits_4	iris
KNN	0.63	0.92	0.78	0.97
SVC_lin	0.69	0.93	0.74	0.98
SVC	0.69	0.93	0.77	0.95
DT	0.67	0.91	0.76	0.95
RF	0.69	0.92	0.78	0.95
MLP	0.70	0.93	0.77	0.98
AdaBoost	0.30	0.81	0.74	0.94
GNB	0.70	0.93	0.77	0.98
QDA	0.69	0.93	0.77	0.98

Figure 25: Scores Classification avec reduction LDA

On remarque que la réduction semble mieux marcher sur le dataset digits_6. Pour le reste, la réduction semble dégrader les scores.

6.2.4 Reduction t-SNE

	digits_all	digits_6	digits_4	iris
KNN	0.98	1.00	1.00	0.97
SVC_lin	0.93	1.00	1.00	0.97
SVC	0.70	0.71	0.70	0.97
DT	0.69	1.00	0.99	0.97
RF	0.90	1.00	1.00	0.97
MLP	0.96	0.99	1.00	0.96
AdaBoost	0.32	0.50	0.75	0.86
GNB	0.95	1.00	1.00	0.93
QDA	0.95	1.00	0.99	0.97

Figure 26: Scores Classification avec reduction t-SNE

On observe ici une nette amélioration significatives des scores.

6.2.5 Reduction Spectral Embedding

	digits_all	digits_6	digits_4	iris
KNN	0.73	0.91	0.90	0.95
SVC_lin	0.13	0.22	0.34	0.32
SVC	0.13	0.22	0.34	0.32
DT	0.65	0.90	0.92	0.93
RF	0.74	0.91	0.91	0.91
MLP	0.10	0.17	0.25	0.35
AdaBoost	0.38	0.54	0.84	0.93
GNB	0.69	0.91	0.92	0.93
QDA	0.68	0.90	0.91	0.94

Figure 27: Scores Classification avec reduction SPE

6.2.6 Reduction Spectral Embedding RBF

	digits_all	digits_6	digits_4	iris
KNN	0.98	1.00	1.00	0.89
SVC_lin	0.11	0.23	0.27	0.32
SVC	0.37	0.41	0.54	0.32
DT	0.72	0.99	0.98	0.88
RF	0.97	0.99	1.00	0.93
MLP	0.51	0.45	0.51	0.32
AdaBoost	0.30	0.61	0.75	0.95
GNB	0.72	0.98	1.00	0.92
QDA	0.88	0.98	1.00	0.94

Figure 28: Scores Classification avec reduction SPE_rbf

On remarque ici que les réductions SPE et SPE_rbf n'améliorent pas autant les scores que la réduction t-SNE.

7 Conclusion

Il existe plusieurs méthodes de réduction de dimensions, certaines linéaires d'autres non. Historiquement ces méthodes ont fortement évolué. PCA qui est une technique assez ancienne, qui tend à être remplacée par de nouvelles méthodes comme t-SNE ou UMAP. Cependant elles restent toujours utilisées car elles sont assez "simple" d'utilisation ce qui garantit un temps d'exécution convenable.