

# Etat de l'art:réduction de la dimension

## M2 Machine Learning for DataScience

Joseph Gesnouin & Yannis Tannier

Université Paris Descartes

Novembre 2018

### Abstract

Les jeux de données multidimensionnels peuvent être un casse-tête à visualiser afin d'en recueillir des informations facilement identifiables. Tandis que les jeux de données de faibles dimension sont très facilement affichables car il est aisément de représenter la structure inhérente de leurs données, il n'est pas facile de réaliser une visualisation intuitive pour des jeux de données à grande dimension. Afin d'améliorer la capacité de visualisation de ces grands jeux de données multidimensionnels, la dimension doit être réduite, tout en essayant de garder le maximum d'information. L'objectif des méthodes factorielles présentées dans ce rapport sont doubles: réaliser une réduction de la dimension mais également assister à l'interprétation des informations conservées.

Théoriquement, la famille des méthodes d'analyse factorielle peut se subdiviser en deux familles de méthodes:

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

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

Dans un premier temps, nous présenterons de manière assez brève une liste non exhaustive des algorithmes utilisés pour cet état de l'art. Puis nous réaliserons une étude

comparative de ces algorithmes de réduction de dimension linéaire et non-linéaires sur des jeux de données très utilisés dans la littérature: USPS, PenDigits et Mnist.

### Algorithmes de réduction de dimension

#### Approches Linéaires:

- **Random indexing/Random Projection** est un moyen simple et assez rapide pour réduire la dimension d'un jeu de données. Les dimensions et les distributions des matrices de projections aléatoires sont contrôlées de telle sorte qu'elles conservent approximativement les distances entre deux échantillons du dataset. Cette méthode repose sur le fait que pour des points dans un espace vectoriel de dimension élevée, il existe un espace vectoriel à dimension réduite de telle sorte que cet espace conserve approximativement les distances entre les points.
- **L'ACP** consiste à transformer des variables corrélées en axes décorrélés les uns des autres. Ces axes sont composés uniquement de combinaisons linéaires des variables précédentes. Cette méthode est une des premières méthodes d'analyse factorielle et moteur central de toute autre méthode d'analyse factorielle. Dans notre cas, nous projeterons les points du jeu de données sur les deux premiers axes factoriels créant ainsi un nouvel espace vectoriel, les points du jeu de données initial projetés sur un espace réduit, nous espérons pouvoir grâce à ces nouveaux axes et en fonction de l'inertie expliquée, faire parler notre jeu de données et avoir une idée plus claire de la structure en classe de celui-ci.

#### Approches non-linéaires:

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

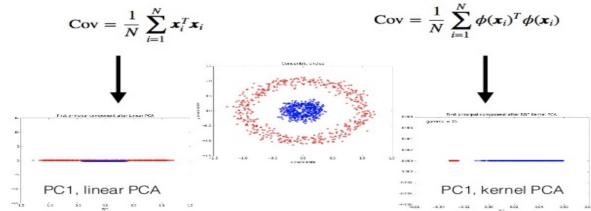


Figure 1: Exemple de K-ACP et ACP sur dataset:Kernel

Ci-dessus, un exemple où K-ACP capture bien cette non-linéarité, le jeu de données Kernel: il suffit de créer une nouvelle variable composante non linéaire des deux autres variables précédentes.dans ce cas particulier la fonction kernel de transformation  $\phi$  sera  $\phi(x,y)=x^2 + y^2$ .

On passe alors d'un jeu de données aux classes inséparables linéairement en dimension 2 à un jeu de données aux classes séparables linéairement en dimension 3.

- **Multi-Dimensional Scaling:** MDS cherche une représentation des données dans un espace vectoriel réduit, pour lequel, la projection des individus respecte bien les distances et sont équivalentes à celle de l'espace vectoriel d'origine de dimension bien supérieure. En général, c'est une technique utilisée pour analyser la similarité ou la dissimilarité entre les données. Ces dissimilarités sont représentées sous forme de distances dans un espace géométrique.
- **Isomap:** Isomap peut être considéré comme une extension de MDS ou de K-PCA. Cet algorithme cherche un espace de dimension réduite qui conserve les distances "géodésiques" entre tous les points. Cette méthode se base sur les plus proches voisins de chacune des instances afin de conserver cette distance. Premièrement, Isomap génère un graphe de voisinage entre toutes les instances du jeu de données. Une fois ce graphe de voisinage généré, une table de dissimilarité est créée grâce à l'algorithme de distance de Djikstra. De cette manière, Isomap dispose de toute l'information nécessaire afin de voir quels sont les points les plus proches des autres. Pour finir, on utilise MDS sur cette table de dissimilarité.
- **Locally Linear Embedding:** LLE peut se définir comme une multitude de méthodes factorielles linéaires comparées et concaténées une-à-une dans des zones locales. On pourrait le comparer à une succession d'ACP lancées dans des zones locales du jeu de données puis comparées globalement afin de garder la meilleure qualité des données non-linéaires. Il se base sur une intuition géométrique: on suppose que tous les points du jeu de données ainsi que ces voisins sont séparables ou disposés d'une manière linéaire dans une zone locale. Une variante Hesseenne de LLE existe: également connue sous le nom de **Hessian Eigenmapping**, il semblerait que LLE soit sensible au problème de la régularisation. Quand le nombre de voisins est supérieur au nombre de dimensions, la matrice gardant en mémoire chacun des champs locaux linéaires n'a pas suffisamment de rangs. Pour éviter ce problème de régulari-

sation, on utilise une forme hessienne pour chaque groupe local afin de bien définir la structure linéaire.

- **Local Tangent Space Alignment:** LTSA est similaire algorithmiquement parlant à LLE. Cependant, au lieu de se concentrer à préserver les distances locales comme le fait LLE, LTSA essaie de caractériser chaque groupe local par son espace tangent. Il se base sur l'intuition que si la réduction est correcte, tous les hyperplans tangents au jeu de données seront alignés.

- **t-distributed stochastic neighbor embedding:** 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.

Un paramètre très important de t-SNE est la perplexity: elle sert de balance entre l'attention que l'on porte entre les zones locales et les zones globales du dataset. Utiliser une perplexité élevée, entraînera un nombre plus grands de voisins à prendre en compte et donc une sensibilité amoindrie pour les structures plus petites. À l'inverse, une perplexité faible prendra en compte un nombre réduit de voisins dans ses calculs et donc de ce fait ignorera probablement quelques informations globales en faveur d'un pattern local non important.



Figure 2: t-SNE: Exemple de variation du paramètre perplexity sur un jeu de données à deux classes

Plus un dataset aura un nombre d'instances élevées, plus le nombre de voisins utilisés pour avoir une bonne idée de représentation locale sera élevé, et par définition une perplexité plus élevée sera nécessaire. De même, les jeux de

données avec beaucoup de bruit, nécessiterons une perplexité élevée également afin d'outrepasser le bruit de chaque groupe local.

Un autre paramètre important à optimiser est le learning rate. Si celui-ci est trop bas, l'algorithme de descente du gradient restera bloqué dans un optimum local, si celui ci est trop haut la divergence de Kullback-Leibler augmentera durant l'optimisation.

Il est important de noter que pour t-SNE, la taille des clusters ne peut avoir aucune valeur:

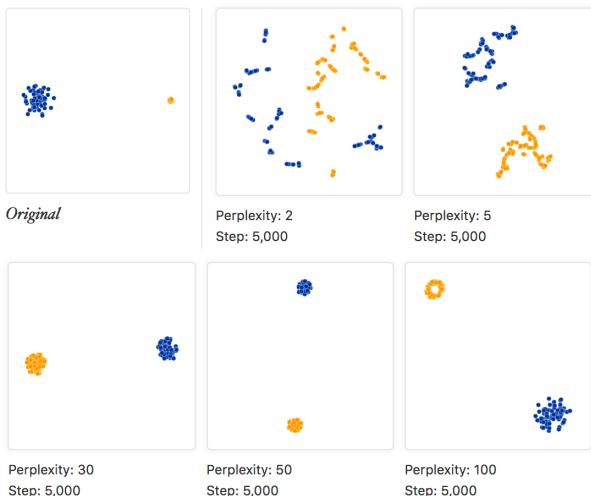


Figure 3: t-SNE:Exemple à deux classes de tailles variantes

Ci-dessus, un exemple de l'algorithme de t-SNE à deux classes, de tailles différentes. Pourtant les clusters ont à peu près la même taille après l'utilisation de l'algorithme t-SNE. L'algorithme adapte sa notion de distance pour chaque région dont la densité varie dans le jeu de données. Il en résulte donc, une croissance de la taille des clusters denses, et une contraction de la taille des clusters plus "sparses". En d'autre termes, t-SNE égalise la densité par défaut. C'est un point à prévoir lors de l'utilisation de l'algorithme.

Parallèlement, la distance entre les clusters ne peut conserver aucune information importante et ne peut pas être considérée comme information acceptable: pour être conservée, la géométrie globale du jeu de données requiert un fine-tuning de la perplexity.

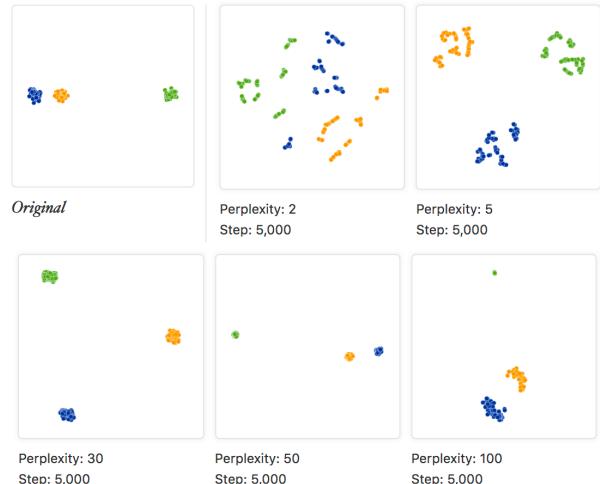


Figure 4: t-SNE:Exemple à trois classes de distances inégales

Les conditions d'utilisation et d'interprétation de cet algorithme pour un data scientist sont multiples. On pourrait le qualifier de boîte noire. Contrairement aux autres algorithmes de réduction non linaires: il rend difficile au processus de fournir des inférences et des points de vue fondés sur les résultats. C'est un algorithme très puissant lorsqu'il s'agit de découvrir les structures cachées à l'intérieur des données. Ce qui en fait un excellent algorithme de visualisation, cependant, il convient de bien comprendre toutes les facettes négatives de son utilisation afin de ne pas se laisser séduire par les résultats visualisés.

## Présentation des jeux de données utilisés et premières visualisations

Afin de réaliser notre étude comparative des différents algorithmes de réduction matricielle présentés, nous avons choisi de porter notre attention sur trois jeux de données portant sur le même type de données mais de taille différentes: PenDigits, USPS, MNIST.



Figure 5: Code couleur utilisé pour les méthodes de visualisation

### PenDigits

Le jeu de données PenDigits (Pen-Based Recognition of Handwritten Digits) est un jeu de données multiclass à 16 attributs et 10 classes: (0 ... 9). Ce jeu de données a été créé en collectant 250 échantillons de 44 personnes différentes. Dans ce jeu de données, toutes les classes ont une fréquence similaire: chaque classe a une fréquence d'apparition de 10%.

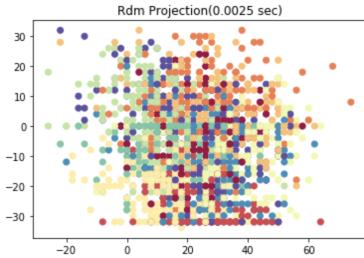


Figure 6: Visualisation aléatoire dans un espace de dimension réduite des premiers exemples du jeu de données PenDigits

La manière la plus facile et rapide de réduire la dimension d'un jeu de données est de réaliser une projection aléatoire des données sur un espace vectoriel réduit. Cela permet d'obtenir une première visualisation instantanée du dataset. Même si utiliser ce type de projection fait obligatoirement perdre le format des structures inhérentes des données les plus importantes, cette première visualisation ne nécessite rien de particulier et ne peut apporter qu'une plus value aux autres méthodes de visualisation lorsqu'il s'agira de jouer avec les paramètres: les méthodes de projection aléatoires préservent assez bien les distances.

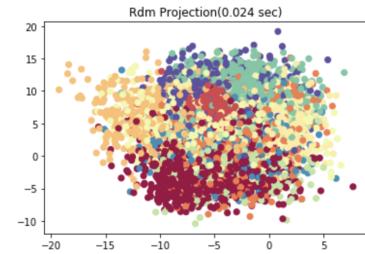


Figure 8: Visualisation aléatoire dans un espace de dimension réduite des premiers exemples du jeu de données USPS

On peut déjà remarquer un "groupement" en fonction de la classe de chaque instance. En ce qui concerne la fréquence et la répartition de chaque classe dans le jeu de données, comme cela n'était pas précisé, nous avons vérifié si la fréquence était équiprobable afin d'avoir plus d'informations sur le jeu de données:

deciles	
10	0.0
20	1.0
30	1.0
40	2.0
50	4.0
60	5.0
70	6.0
80	7.0
90	8.0
100	9.0

### Jeu de données USPS

Publié en 1994, ce jeu de données USPS contient 7291 images d'apprentissage et 2007 de test. les images sont représentées sous forme de 16\*16 pixels en niveaux de gris. Comme PenDigits, les classes disponibles dans le dataset varient de 0 à 9.

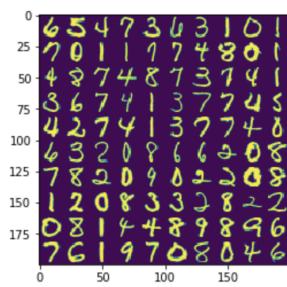


Figure 7: Visualisation des premiers exemples du jeu de données USPS

Figure 9: répartition des classes USPS

### MNIST

Probablement le jeu de données le plus connu lorsqu'il s'agit de faire de l'OCR, MNIST est de loin le plus gros jeu de données des trois utilisés: 60 000 individus. Comme les jeux de données précédents MNIST dispose de 10 classes différentes: de 0 à 9. Les trois jeux de données ayant 10 classes ni plus ni moins.

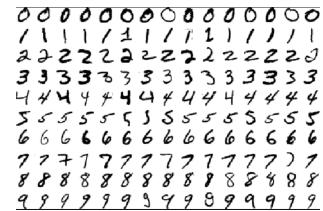


Figure 10: Visualisation des premiers exemples du jeu de données MNIST

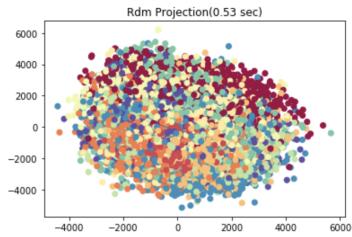


Figure 11: Visualisation aléatoire dans un espace de dimension réduite des premiers exemples du jeu de données MNIST

Nous disposons déjà d'informations non négligeables pour visualiser nos données dans un espace réduit lorsqu'il nous faudra faire varier les paramètres d'utilisation des algorithmes: qu'importe l'optimisation locale/globale, nous cherchons des visualisation avec 10 groupes différents représentant les 10 classes disponibles de fréquences égales dans les jeux de données testés.

De plus la simple utilisation de random projection nous permet de remarquer une certaine différence entre la répartition des distance des classes dans chacun des jeux de données: bien qu'ils portent sur le même sujet ils sont fondamentalement différents sur leur représentation des digits, mais également leur taille. De ce fait un algorithme qui pourrait bien marcher sur un jeu de donné pourrait potentiellement moins bien marcher sur un autre: les distances entre les classes n'étant pas toutes aussi visibles. USPS reste le jeu de données qui semble le plus "facile" à visualiser des trois: les distances interclasses étant plus nettes que les autres.

## Étude comparative

### Jeu de données PenDigits

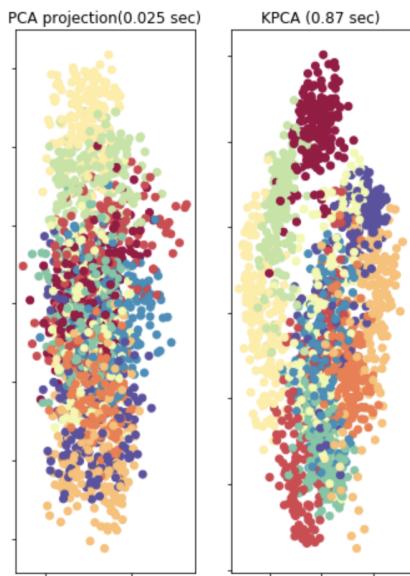


Figure 12: Visualisation PCA / K-PCA du jeu de données PenDigits

Comme prévu, les structures non-linéaires ne sont pas bien gérées par l'ACP dont la définition mathématique correspond à une combinaison linéaire des variables de base, il n'est donc pas étonnant de voir que la réduction de dimension via l'ACP n'arrive pas à conserver ces structures. Néanmoins nous pouvons remarquer que certains des nombres sont tout de même assez bien regroupés: le 5 et le 4.

A l'inverse, le k-ACP bien que peu concluant avec un kernel linéaire, arrive à capturer un peu plus des structures de chaque classes et ainsi reste plus précis que l'ACP seule. Nous pouvons par exemple remarquer des "groupes" de classes: Le 4 et le 5 restent bien définis, à ceci nous pouvons ajouter le 6. Le 3 et le 1 de part leur représentation très proche restent très fortement proches dans la visualisation. Néanmoins k-ACP arrive à conserver une certaine cohérence. Nous avons fait fluctuer plusieurs valeurs du paramètre gamma afin de trouver un bon compromis entre la variance et le biais afin d'avoir une visualisation un tant soit peu interprétable.

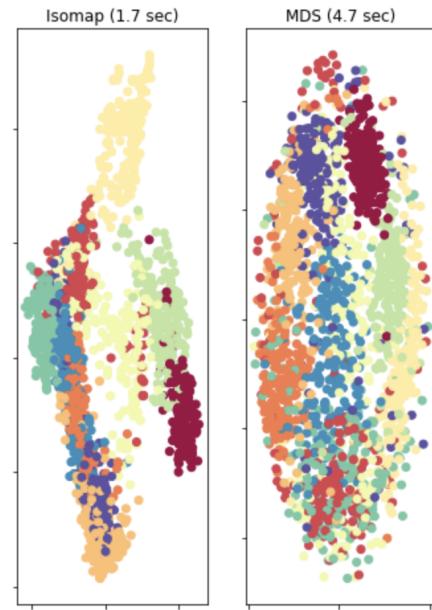


Figure 13: Visualisation ISOMAP / MDS du jeu de données PenDigits

En prenant un jeu de données aussi complexe que des nombres écrits à la main, qui typiquement n'ont pas des formes géométriques claires, Isomap réussit non seulement à trouver des coordonées globalement intéressantes, mais de plus, isole certaines structures que PCA ou MDS ne détectent pas. Si l'on regarde la visualisation générée, l'apparence des interactions linéaires entre les points, ainsi que leurs distances dans l'espace vectoriel réduit confirment que cet algorithme a réussi à capturer certaines structures inhérentes au dataset que les algorithmes présentés précédemment n'ont pas réussi à trouver.

En ce qui concerne MDS, la distance inter-instances est bien respectée entre l'espace vectoriel d'origine et l'espace

vectoriel réduit. Nous remarquons une représentation visuelle assez bien "découpée", malheureusement l'écart entre chacun des groupes ne nous permettrait pas de déterminer la classe d'une instance: les points sont représentés de manière intéressante, mais les distances sont telles qu'il n'est pas envisageable de clusteriser ces résultats. Cet algorithme est surtout utilisé pour mesurer la similarité (resp. dissimilarité) dans la data via leur distance, dans notre cas la dissimilité n'est pas suffisante.

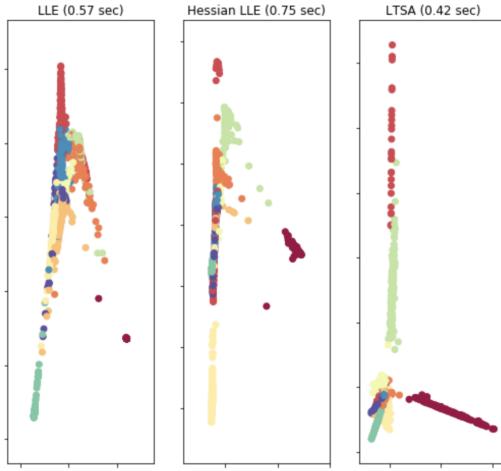


Figure 14: Visualisation LLE / HLLE / LTSA du jeu de données PenDigits

Nous avons ensuite choisi de regrouper la famille des algorithmes non-linéaires qui cherchent à optimiser de manière locale les distances de chaque points.

Contrairement aux projections linéaires de la PCA qui, in fine, reste une projection d'une distribution uniforme autour de sa moyenne, LLE, ainsi que les algorithmes suivants possèdent une structure "épineuse", les pointes de chaque épine correspondent aux configurations extrêmes de chacune des classes.

Pour LLE, qui cherche à préserver les distances dans chaque voisinage local, la distinction de chaque classe reste incertaine. chaque classe est bien regroupée mais nous n'arriverons pas à les départager.

Pour H-LLE, qui est une amélioration de LLE en introduisant plusieurs poids indépendants pour chaque voisinage afin d'éviter le problème de régularisation. Les classes sont plus distinctes: Les structures géométriques déterminées grâce à l'introduction de ces poids locaux sont bien plus stables et permettent donc une amélioration de la visualisation comparée à LLE.

Pour LTSA, qui repose sur le fait que beaucoup de dataset à de grande dimension peuvent être modélisés comme des data points proches les un des autres dans un espace de dimension réduit non-linéaire. L'algorithme distingue encore mieux les classes que les deux autres. Un avantage de LTSA comparé aux méthodes LLE, est que en utilisant LSTA, on pourrait mieux détecter la dimension intrinsèque du dataset en analysant les espaces tangents locaux, ainsi avoir une

meilleure idée de la représentation finale du jeu de données.

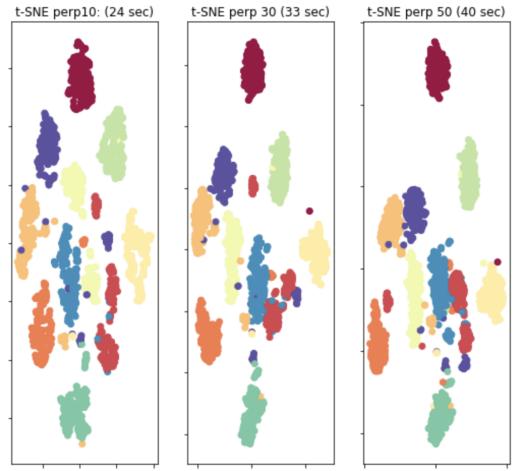


Figure 15: Visualisation t-SNE perp 10/30/50 du jeu de données PenDigits

Contrairement à Isomap, LLE et toutes les variantes présentées jusque-ici, t-SNE va se concentrer sur la structure locale des données et va tendre vers une extraction de chacun des groupes locaux. Cette capacité à grouper les échantillons en fonction des structures locales voir topologiques, peut donner de très bon résultats, et ainsi démeler visuellement un dataset comprenant plusieurs structures inhérentes simultanément. Comme c'est le cas dans les datasets de digits présentés.

Nous avons approfondi l'étude de l'algorithme t-SNE en faisant varier la perplexité: la taille du jeu de données étant très importante dans le choix de ce critère, nous espérons pouvoir comparer ces résultats avec les autres jeux de données bien plus conséquents. PenDigits étant un jeu de données assez restreint, nous remarquons que plus la perplexité est élevée, moins nous arrivons à distinguer les 10 classes.

Quoiqu'il arrive t-SNE reste une méthode de visualisation capable de représenter la structure des données et on peut facilement constater que notre jeu de données est constitué d'une dizaine/douzaine de groupes. Cependant comme dit précédemment, il faut raison garder et ne pas oublier que t-SNE n'est pas réellement un algorithme de reduction de dimension mais plutôt de visualisation: t-SNE n'effectue pas un mapping point par point de l'espace vectoriel original à l'espace de dimension réduite. De ce fait il est inutile d'appliquer des méthodes sur les points générés par t-SNE en espérant garder de l'information de l'espace initial comme pour les autres méthodes. t-SNE essaie simplement de placer les points dans un espace réduit de telle sorte que les points ayant une dissimilarité (resp. similarité) élevée soient éloignés(resp. proches). La seule relation que garantit t-SNE contrairement aux autres algorithmes, est que un point p sera le même dans les deux espaces vectoriels. De ce fait on aura plusieurs de ces points "ancrés" qui seront directement liés dans les deux espaces.

pour illustrer, imaginons un point p équidistant de a et

$b$  dans l'espace initial, l'algorithme le placera également à distance égale entre  $a$  et  $b$  dans l'espace réduit. Le problème étant que t-SNE ne garantit pas que ces points voisins soient proches les uns des autres dans l'espace réduit. Il ne peut pas non plus garantir que 3 points alignés dans l'espace initial ressembleront également à une droite dans l'espace réduit. Même en 3d, il existe une infinité de possibilités de plan de points  $p$  qui seront équidistants et qui auront  $a$  et  $b$  comme voisins les plus proches.

### Jeu de données USPS et MNIST

Les jeux de données présentés étant tous représentatifs du même type de problème et de structure de données: un jeu de données complexe avec plusieurs structures de données locales non linéaires, nous avons préféré regrouper les deux autres jeux de données, les résultats n'étant pas différents de l'analyse précédente: certains algorithmes peinent à capturer les structures complexes des jeux de données proposés.

La plus grosse différence entre les jeux de données présentés reste le nombre de cas d'usage, qui influe fortement sur le choix du nombre de voisins à déterminer pour les algorithmes: un nombre de plus proches voisins trop grand et l'on perd les structures locales, un nombre de plus proches voisins trop petit et l'on commence à représenter le bruit. Il aura donc fallu jouer avec ce paramètre en plus des hyper-paramètres de chaque algorithme afin de trouver une représentation stable.

Les résultats des algorithmes ne diffèrent pas de ceux obtenus sur le jeu de données réduit PenDigits: les analyses des résultats des algorithmes hors t-SNE ne diffèrent pas de l'analyse proposée précédemment pour le jeu de données PenDigits.

Ainsi, les algorithmes linéaires tel que la PCA ou encore k-PCA avec un kernel linéaire n'arrivent toujours pas à capturer les structures non-linéaires du dataset. Isomap et MDS, quand à eux, ont toujours cette forme de distribution normale centrée autour de leur moyenne. Ils arrivent à capturer une petite partie des structures inhérentes au jeu de données. Cependant, il nous est impossible de distinguer les classes. MDS semble "regrouper" les classes de manière circulaire contrairement à précédemment où les clusters étaient plus ou moins séparables, du moins visuellement. LLE/HLL/LA arrivent assez bien segmenter visuellement chaque groupe. Cependant sans les informations de retour apprentissage nous ne serions capable de trouver que trois groupements représentés par les épines des points extrêmes de chaque classes générées par les algorithmes.

Finalement, en ce qui concerne t-SNE, nous aurions espéré plus de variance entre les différents appels avec une perplexité variante, le papier présentant t-SNE proposait de toujours choisir des valeurs entre 5 et 50 pour la perplexité. C'est pourquoi nous ne sommes pas allés plus loin. Nous aurions du augmenter celle-ci afin de noter une réelle différence entre les visualisations. Cependant, nous avons fait le choix de respecter les recommandations du papier et les valeurs que nous nous étions fixer précédemment.

Au niveau des résultats, pour le moins visuels, t-SNE remplit sa mission aussi bien que pour le jeu de données PenDigits. Il nous permet d'avoir une petite idée du nombre de

classes de chaque jeu de données: une petite douzaine. En tant que data scientist, avoir une idée plus précise de la répartition de ses données avant même de lancer ses algorithmes d'apprentissage est non négligeable. C'est pourquoi les méthodes de réduction de dimension et de visualisation, bien que souvent boudées, voir ignorées ne doivent pas être prises à la légère: toute information relative au jeu de données est bonne à prendre afin de comprendre au mieux la data et de la faire parler. Ces familles d'algorithmes représentent donc une étape importante de la data science en étant une partie à part entière de l'analyse du jeu de donnée avant de vouloir en prédire des choses.

Manifold Learning with 7291 points, 10 neighbors

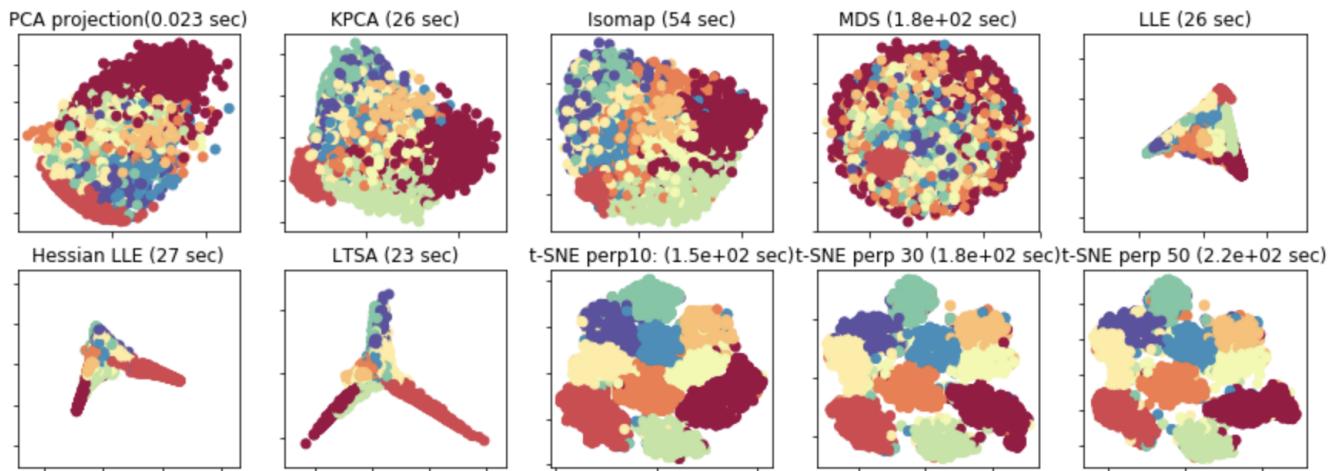


Figure 16: Visualisations USPS

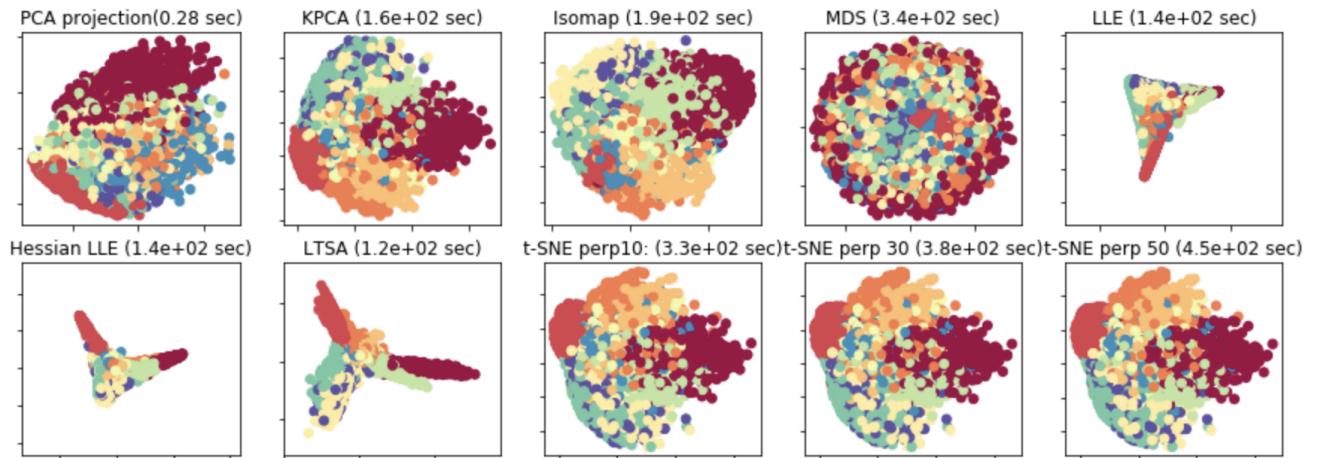


Figure 17: Visualisations MNIST

## Conclusion

Nous avons eu l'occasion de tester et visualiser les retours de plusieurs des algorithmes de réduction de dimension, ainsi que les utiliser dans des cas d'utilisation concrets. Comme prévu, les méthodes factorielles linéaires telles que l'ACP n'arrivent pas à capturer certaines structures de données complexes. D'autres méthodes non-linéaires telles que Isomap, MDS, peinent à capturer certaines de ces structures mais restent plus précises que la PCA. Pour les méthodes d'optimisation des distances euclidiennes, géodésiques ou d'espace tangent local telles que LLE, HLLE et LTSA, celles-ci regroupent en moyenne assez bien chaque classe mais la visualisation peut s'avérer compliquée. Pour finir t-SNE est une méthode de visualisation impressionante qui a permis d'approximer visuellement le nombre de classes de chaque jeu de données. Il ne faut cependant pas oublier toutes les contraintes d'utilisation liées à t-SNE expliquées précédemment. Il aurait également été intéressant de rajouter la factorisation matricielle non négative dans les algorithmes à traiter, afin de voir comment celle-ci se positionnait sur les différents datasets vis à vis des méthodes qui lui sont voisines et que nous avons proposé.

Pour aller plus loin, nous aurions aimé essayer ces algorithmes sur des jeux de données portants sur différents sujets. Les trois jeux de données proposés ne diffèrent pas trop les un des autres à part la taille de l'échantillon et leur expression des distances. Bien que relativement basiques: ces jeux de données continuent à poser des problèmes à l'état de l'art actuel en réduction de dimension. Dans les papiers que nous avons lu, il semblerait que pour une grosse partie des problèmes en reconnaissance des formes se situent au niveau du preprocessing des signaux multidimensionnels: par exemple pour les images de visage. Souvent le but de ce preprocessing est une sorte de réduction de la dimension pour compresser le signal en taille et découvrir des représentations compactes, des structures dans la donnée. Il aurait été formateur de tester ces algorithmes sur des datasets variés et dont la complexité des structures non-linéaires est progressive afin de voir une réelle évolution entre les familles d'algorithmes, mais c'est également un point que nous souhaiterions approfondir dans le cadre de nos missions en entreprise: réduire la dimension des images de visage et des iris pour la détection de faux visage, ainsi que de réduire la taille de l'amas d'informations générées par le parc Livebox et STB des différents pays Orange pour en extraire de la donnée exploitable. Ces deux missions se prêtent bien à l'utilisation d'algorithmes de réduction de dimension et nous ne manquerons pas d'en faire part à nos collègues à notre retour en entreprise.

## References

- [1] Tenenbaum, J.B.; De Silva, V.; & Langford, J.C. "A global geometric framework for nonlinear dimensionality reduction".
- [2] Roweis, S. & Saul, L. "Nonlinear dimensionality reduction by locally linear embedding".
- [3] Zhang, Z. & Wang, J. "MLLE: Modified Locally Linear Embedding Using Multiple Weights".
- [4] Donoho, D. & Grimes, C. "Hessian Eigenmaps: Locally linear embedding techniques for high-dimensional data".
- [5] Zhang, Z. & Zha, H. "Principal manifolds and nonlinear dimensionality reduction via tangent space alignment".
- [6] Kruskal, J. Psychometrika "Nonmetric multidimensional scaling: a numerical method".
- [7] Borg, I.; Groenen P. Springer "Modern Multidimensional Scaling - Theory and Applications".
- [8] van der Maaten, L.J.P.; Hinton, G "Visualizing High-Dimensional Data Using t-SNE".
- [9] van der Maaten, L.J.P. "t-Distributed Stochastic Neighbor Embedding".