State of the art

How to evaluate the faithfulness of visual data projection ?

June 10, 2022

Students :

Cyril	GARDENAT	cyril.gardenat9@etu.univ-lorraine.fr
Nathan	METZGER	nathan.metzger6@etu.univ-lorraine.fr
Muhamed	SILIC	muhamed.silic3@etu.univ-lorraine.fr

Tutors :

Lydia BOUDJELOUD-ASSALA

Keywords: projection; méthodes; critères; évaluation; qualité; réduction de dimension

Abstract :

Dans cet état de l'art, nous nous sommes concentrés sur une approche de visualisation de données : la projection. Ainsi, différentes méthodes de projection (linéaires et non linéaires, supervisées et non supervisées) ont été présentées de façon non exhaustive. Une perte de donnée étant inévitable lors de la projection de grands jeux de données, la problématique de la qualité de cette dernière se pose. De ce fait, nous avons également cité les biais qui peuvent découler d'une projection ainsi que les critères qui permettent de juger de sa qualité.

Contents

Introduction

La représentation visuelle des grands jeux de données est un enjeu de plus en plus important dans de nombreux domaines scientifiques. En effet, une immense partie des analyses de données se fait dans des domaines de recherches dont les acteurs ne sont pas des “data analyst”. Cela rend l’analyse compliquée et peut engendrer certains biais lors de l’interprétation des données [**?**]. C’est pour cela, que le domaine de la visualisation des données se doit d’être accessible à ce genre de profil.
La visualisation permet aux utilisateurs d’avoir un aperçu global des données et de choisir en conséquence quel type de traitement et quel paramétrage utiliser dans leurs analyses. Elle a donc pour rôle d’être une interface interactive pour représenter et naviguer à travers les données extraites, et ce dans le but de les rendre compréhensibles et donc exploitables[**?**]. C’est exactement ce vers quoi tend ce domaine : *“Visual analytics combines automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making on the basis of very large and complex data sets”* [**?**]. Différentes techniques de visualisation existent [**?**], mais c’est sur la projection que nous allons nous concentrer.

Dans une projection, toutes les données sont projetées selon une projection linéaire ou non des axes dans un plan en 2D. Cela permet de représenter les données sous la forme d’un nuage de points (on parle de "scattered representation") en essayant de respecter au mieux la structure des données [**?**].
Le principe de base d’une lecture projection est le suivant : les points qui sont proches les uns des autres sont censés être “similaires” tandis que ceux qui sont éloignés sont censés être "différents”.
Les projections permettent également d’inférer des propriétés sur les données et de les vérifier. Par exemple, il est possible de vérifier si les clusters sont séparables linéairement (notamment à l’aide d’une projection linéaire) [**?**] .

Dans le cas de la visualisation, acquérir les données et les traiter sont deux étapes qui sont de plus en plus simples. Le défi est de représenter fidèlement les jeux de données. En effet, dans une projection les jeux de données peuvent atteindre jusqu’à des dizaines de milliers de dimensions et doivent être réduit en 2 ou 3 dimensions pour être visualisable et compréhensible par l’homme. Or ce processus de dimension a un coût : des pertes d’informations. Cependant, ce processus a un coût, étant donné qu’il est lié à une perte d’information causée par l’étirement ou la compression de la plupart des distances entre paires, car la dimension de l’espace couvert par les données d’origine est généralement supérieure à la dimension de l’espace de projection. ce qui cause l’apparition d’artefacts dans la projection. Ces artefacts compromettent ainsi grandement la fiabilité et la bonne interprétation de ces graphiques[**?**]. Dans une projection, les axes ne jouent pas forcément un rôle : c’est principalement la distance entre les points qui donne un sens et une interprétabilité à la projection[**?**]. Néanmoins, représenter fidèlement une distance euclidienne après réduction et projection est tout bonnement impossible.

Par conséquent, la mesure et la visualisation de ces distorsions appelées "artefacts" sont cruciales pour l’analyse. Elles se doivent d’être analysées afin de détecter quelles distances entre paires ont été préservées, et donc d’évaluer si les caractéristiques observées dans l’espace de projection sont des images fidèles de certaines caractéristiques de l’espace originel, ou simplement des artefacts de la projection.

Les artéfacts sont définis comme des points "mal placés”(en comparaison aux points d’origine). Ils sont dus au processus de réduction qui peine à respecter fidèlement les distances. Il existe deux types d’artefacts : les artefacts géométriques et les artefacts topologiques [**?**].

- Les artefacts géométriques sont causés par de légères distorsions des distances, dans ce cas précis les distances sont fausses, mais le voisinage des points est conservé, l’interprétation n’est donc pas (ou très peu) perturbée.

- Les artefacts topologiques sont causés par des distorsions trop importantes des distances. Par conséquent, cela engendre forcément des problèmes d'interprétation.

Parmi ces types d'artefacts, nous pouvons distinguer 4 sous-types d'artefacts :[?] :

- Les compressions : Les distances par paire de projections sont inférieures aux distances par paire d'origine correspondante, mais la topologie du voisinage d'origine est préservée.
- Les étirements : Les distances par paires projetées sont plus grandes que les distances par paires d'origine correspondantes, mais la topologie du voisinage d'origine est préservée.
- Les collages : Une compression particulière dans laquelle les points très éloignés dans l'espace d'origine deviennent des voisins proches dans l'espace de projection, changeant ainsi radicalement la topologie du voisinage.
- Les déchirements : Un étirement particulier où des points proches dans l'espace d'origine sont projetés loin les uns des autres, changeant drastiquement la topologie du voisinage.

Il y aurait deux causes principales à l'origine de ces artefacts [?] :

- Les causes structurelles(ou intrinsèques) : Les structures géométriques et topologiques des collecteurs d'origine et celles de l'espace de projection ne sont pas compatibles, de sorte que la projection ne peut se faire sans modifier les distances par paires entre les données.
- Causes techniques(ou extrinsèques) : Si la technique de projection est non linéaire, elle peut créer des artefacts qui n'existent pas à l'optimum global et donc modifier la forme initiale.

Les artefacts ayant des causes techniques peuvent être annulés alors que ceux ayant des causes structurelles ne le peuvent pas. Cependant, dans la pratique, il n'est guère possible de distinguer les deux causes d'artefacts dans le cas des projections non-linéaires.

Pour tenter de pallier les difficultés induites par les artefacts, des techniques ont été mises en place pour assister l'utilisateur dans la visualisation. Parmi ces techniques, nous pouvons noter l'utilisation d'échelles colorimétriques[?], qui permettent de mesurer les différents artefacts, mais ne permettent pas de mettre en valeur les clusters cachés ni de donner une idée claire ou de préciser de la qualité de la projection. Dans ce cas, il est compliqué d'exploiter une projection de n dimensions de façon fiable sans prendre en compte les artefacts qu'elle présente[?]. Dans l'article "Visualizing Dimensionality Reduction Artifacts : An evaluation"(Heulot) une technique colorimétrique est utilisée pour voir s'il est possible de surmonter les difficultés dues aux artefacts dans d'interprétation d'échelles multidimensionnelles. Ils ont donc mis en place une méthode s'appelant "ProxiViz". Grâce à celle-ci, sur un jeu de test , les informations locales d'un item sont beaucoup mieux représentées. Cette méthode permettrait de détecter plus facilement les outlier et les clusters. Elle pourrait donc, en partie, aider les non "spécialistes des data" à mieux distinguer certaines informations. Même si encore une fois cela est limité par la qualité de l'écran, le contraste et la vision de l'humain qui peut peiner à distinguer les nuances entre les couleurs[?] [?] [?]. L'idée a notamment été reprise par Heulot[?], il se base sur le concept de la "visualisation interactive des proximités" afin de mettre en évidence les artefacts. Cela permet de visualiser sur la projection les proximités d'origines des différents points en fonction d'une référence choisie par l'utilisateur.

1 Les méthodes de projection

Lorsque l'on souhaite représenter en 2 ou 3 dimensions de grands jeux de données multidimensionnels(au-delà de vingt dimensions[?][?]), il faut passer par une réduction de dimension. Cette étape intervient dans le processus de projection et est indispensable dans ce cas précis où il y a un nombre important de dimensions(plus d'une vingtaine [?]). Néanmoins, une réduction implique inexorablement une perte d'information et donc une perte de précision, ce qui peut biaiser la projection ainsi que son interprétation. Il est donc important, malgré des pertes inévitables, d'essayer de conserver au maximum les structures et les données utiles. Le facteur numérique est donc un facteur clé, qui se doit d'être compris et maitrisé.

Dans l'espace des données, il est possible d'avoir recours à différent types de mesures(différents critères) pour déterminer à quel point deux points de la projection sont similaires. Cela dépend grandement de la sémantique sous-jacente à la notion de similarité, celle-ci étant liée au domaine scientifique d'où proviennent les données et à leur nature [?]. Comme mentionné plus haut, la projection se base sur un encodage des similarités par le biais de la variable de position: plus les objets se ressemblent, plus ils seront proches sur la projection, et inversement. Dans une projection, une dissimilarité métrique dans l'espace des données correspond à la définition mathématique d'une distance [?] [?]

$$d : \mathbb{R}^m * \mathbb{R}^m \to \mathbb{R}$$

, qui pour

$$\forall (u, v, w) \in \mathbb{R}3 * m$$

satisfasse les conditions suivantes [?] :

- La définition : $d(u, v) = 0 \Leftrightarrow u = v$
- La positivité : $d(u, v) \geq 0$
- La symétrie : $d(u, v) = d(v, u)$
- L'inégalité triangulaire : $d(u, v) \leq d(u, w) + d(w, v)$

La distance de Manhattan et la distance euclidienne peuvent subir la domination d'une dimension qui aurait des valeurs réparties sur un spectre plus large que les autres[?]. Pour corriger cela, on normalise ou on pondère les dimensions (il faut cependant faire attention à bien prendre en compte les poids dans l'interprétation de la similarité).

Il est possible de séparer les méthodes de projections en plusieurs catégories. Elles peuvent être non supervisées ou supervisées : c'est-à-dire qu'elles nécessitent l'étiquetage des données et/ou l'intervention de l'utilisateur pour positionner les points sur la projection[?] [?]. Les algorithmes de projections peuvent également être séparés en deux catégories : les algorithmes de projections linéaires et non linéaires. Dans le cas des algorithmes de projection linéaire, l'espace de plus faible dimension(celui de la projection) est obtenu par la combinaison linéaire des dimensions de l'espace de départ.

Les algorithmes de projection non linéaires partent du principe qu'il existe une variété non linéaire de plus faibles dimensions que l'on peut projeter sans trop de déformations dans un espace 2D(ou 3D). Les données sont ainsi projetées en conservant les propriétés de leur variété. Pour cela, elles utilisent des mesures de stress métriques(selon les distances) ou Non-Métriques(selon le rang des distances). Ensuite plusieurs méthodes d'optimisation sont utilisées telles que la descente de gradient, les réseaux de neurones[?] ou bien le placement par force(pour les jeux de données les moins gros)[?]. Cela permet soit de trouver un optimum global(qui préserve la structure globale) ou un optimum local(dans le cas où les méthodes préservent les voisinages locaux).

1.1 Analyse en Composantes Principales(ACP)

Ce type de projection linéaire vise à préserver de la façon la plus optimale la variance dans les données en transformant des variables corrélées en variables non corrélées(également appelées "composantes principales") qui expliquent la variabilité dans les données. L'ACP mesure l'inertie des données pour réduire au maximum le nombre de variables. Géométriquement parlant, elle a recourt à une projection orthogonale des données dans un sous-espace principal obtenu par combinaison des dimensions dans le jeu de données. Le sous-espace est calculé de sorte que le nuage de points obtenu à partir de la projection maximise la variance des données [**?**].
Cette méthode est beaucoup utilisée dans de nombreux domaines d'application. Elle fournit une pondération des composantes principales, ce qui permet de déterminer les directions de dispersions des données les plus importantes. La projection résultante est directement interprétable,car les vecteurs propres décrivent la contribution de chaque variable [**?**]. Néanmoins, elle ne permet pas de séparer des relations non linéaires et est plus susceptible d'introduire du faux voisinage entre les points.

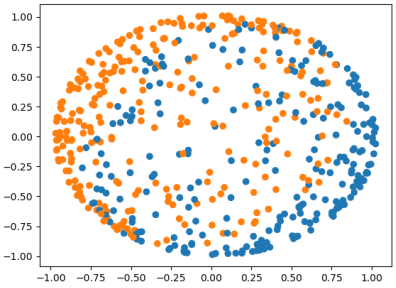

Figure 1: Réduction du dataset Globe

Cette méthode est très performante en termes de temps de calcul et respecte fortement la distance au voisinage de chaque point, cependant elle ne représente pas correctement les classes. La méthode est implémentée dans la librairie scikit-learn.

1.2 Multidimensional Scaling (MDS)

Le principe de fonctionnement de cette méthode est de projeter les données dans un sous-espace linéaire[**?**]. L'objectif de cette approche est de conserver de la meilleure façon possible les paires de distances au carré provenant de l'espace des données. Pour y parvenir, l'algorithme cherche pour chaque métrique de l'espace des données, une combinaison linéaire optimale. Globalement cette technique a les mêmes défauts que l'ACP quand il s'agit de projeter des données linéairement séparables. Néanmoins, dû au fait que la cette technique prend une matrice de similarité en entrée, les axes de projection de cette technique ne sont pas directement interprétables[**?**].

Cette méthode est performante en termes de temps de calcul et de mémoire. Dans le cas du jeu de données utilisé (GLOBE), les classes sont préservées, néanmoins une d'entre elles est projetée à l'extérieur.

La méthode est implémentée dans la librairie scikit-learn. `https://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html`

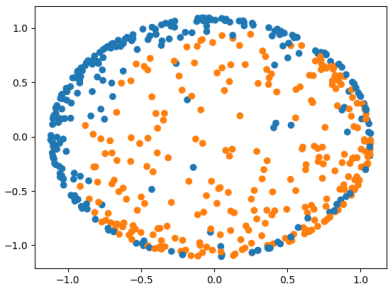

Figure 2: Réduction du dataset Globe

1.3 ClassNerv

Cette méthode a été présentée lors de la prestigieuse conférence NeurIPS de 2020 [?].

C'est une méthode de projection qui permet de réduire la dimension de l'espace de décision, et de réduire la dimension des données en conservant les classes et leur voisinage. Elle présente les avantages des méthodes supervisées et non supervisés, c'est à dire la combinaison du traitement par classe et par distance au voisinage. On peut y faire varier les paramètres

$$\tau^* = \frac{\tau^{\in} + \tau^{\notin}}{2}$$

$$\varepsilon = \frac{\tau^{\in} - \tau^{\notin}}{2}$$

$\tau^* \in [0, 1]$ avec 0 pour un minimum de faux voisin et 1 pour un minimum de voisin raté.
$\varepsilon \in [0, 0.5]$ 0 pour réduire la supervision, et 0.5 pour l'augmenter.
Par exemple sur le Dataset Globe (deux classes séparées en deux hémisphères, représentation d'origine en 3D), on obtient les résultats suivants :

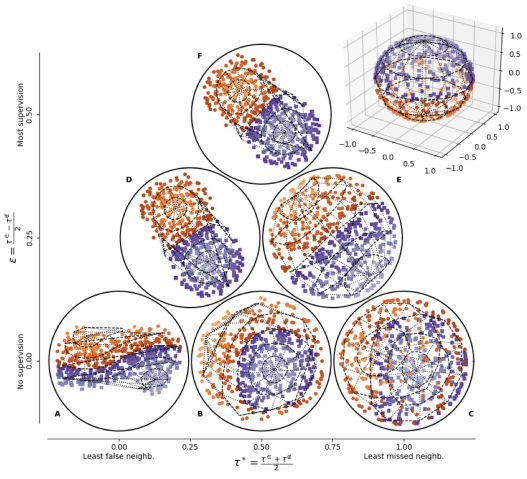

Figure 3: Réduction du dataset Globe

On peut remarquer que l'on obtient un meilleur résultat avec un ε de 0.25 et un τ^* de 0.75, c'est-à-dire une supervision moyenne et en privilégiant les faux voisins plutôt que les voisins manqués.

Implementation

Il est également important de noter que cette méthode est très coûteuse en temps de calcul.

1.4 t-SNE

Cette méthode non linéaire se base sur un algorithme de projection: le SNE [?] celui-ci permet de bonnes visualisations, mais a une fonction de coût qui est difficile à optimiser. C'est notamment de ce problème dont le t-SNE s'affranchit en utilisant une fonction coût "symétrique" et une t-distribution plutôt qu'une distribution Gaussienne. Ces deux changements lui permettent d'enlever les problèmes d'optimisation du SNE. Cette méthode est donc plus simple et plus rapide à exécuter. De plus, les maps produites seraient même de meilleure qualité[?]. Avec le t-SNE, il est possible de restituer fidèlement la structure d'un grand jeu de données au niveau local et au niveau global. Cette méthode permet en effet de repérer la présence de clusters à différentes échelles. Il est par ailleurs possible de retrouver les clusters "séparés", et de faire une approximation du spectral clustering avec un certain paramétrage. [?] [?] [?].

La démarche de cette méthode est la suivante : elle commence par calculer la probabilité P que deux instances puissent être voisines. Ainsi, la valeur de P sera haute pour des voisins proches et très faible pour des voisins éloignés.
La variance de la distribution Gaussienne est différente à chaque fois, ce qui permet de capturer la densité pour différents voisinages multidimensionnels. Le calcul va s'effectuer de façon itérative jusqu'à ce que la perplexité (au préalable définie par l'utilisateur) soit atteinte.
Une fois, toutes les probabilités calculées le but est de trouver une distribution de probabilité Q qui représente fidèlement le jeu dans un espace de plus faible dimension. Cette fois-ci, au lieu d'utiliser une distribution Gaussienne, c'est une distribution du t de Student qui est utilisée, avec un degré de liberté de 1.
Contrairement à P, Q n'est pas paramétré avec une variable de densité de voisinage, par conséquent des voisinages avec des densités très différentes dans l'espace original peuvent être "transformés" en voisinages de tailles équivalents dans la représentation à plus faible dimension. La recherche de Q qui représente fidèlement P se fait en optimisant la fonction coût suivant la divergence de Kull-Leibler des deux distributions. Cette étape est effectuée de façon itérative, en faisant une descente de gradient pour un nombre d'itérations déterminé par l'utilisateur [?].

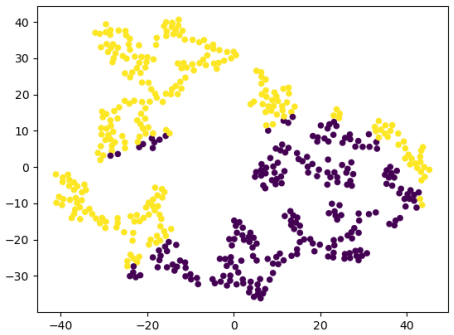

Figure 4: Réduction du dataset Globe

Cette méthode est très performante en temps de calcul, mais cela se fait au détriment des résultats qui sont beaucoup moins précis. Les classes sont relativement bien séparées, mais la distance au voisinage n'est absolument pas respectée.

Celle-ci est implémentée dans des projets github par Laurens Van Der Maaten dans une grande diversité de langages (Matlab, Python/Cuda, R, ...).

1.5 Autres méthodes de projection non linéaires

La liste des méthodes de projections non linéaires ne se limite évidemment pas à celles citées. Nous pouvons également retrouver le Sammon mapping[?], le curvilinear components analysis(CCA)[?], le Maximum Variance Unfolding(MVU)[?] [?], ou le Laplacian Eigenmaps[?]. Ces techniques sont performantes, mais peinent à être efficaces quand il s'agit de visualiser des jeux de données avec beaucoup de dimensions. Il nous était également compliqué de restituer correctement les structures locales ET globales dans une seule projection[?] [?]. C'est en partie pour ces raisons et le manque d'implémentation disponibles qu'elles n'ont pas été retenues pour notre projet.

1.6 Conclusion partie

Comme nous avons pu le voir, il existe de nombreux algorithmes de projections qui se basent tous sur différentes hypothèses mathématiques, différents critères et différents paramétrages. Ainsi, avec un même jeu de données, il est possible d'obtenir plusieurs projections différentes en fonction des algorithmes choisis(le nuage de points ne sera pas le même) . Le choix du type d'algorithme de projection à utiliser dépend de la nature des données que l'on veut étudier, du type de résultat que l'on souhaite observer et de nombreuses caractéristiques propres aux données qui peuvent être très complexes à appréhender sans de bonnes connaissances en projection de données.

Par exemple, nous avons vu pour L'ACP et Isomap, qu'elles ne sont pas stables face aux variations des données. La raison est que l'ajout ou la suppression de quelques instances peut modifier de manière significative les vecteurs et les valeurs propres, affectant ainsi considérablement la cartographie. Même de petites perturbations dans les données peuvent avoir un impact conséquent sur la cartographie finale en raison de l'inversion des vecteurs propres. Ce qui entraîne des configurations très différentes avant et après la perturbation [?]. Le t-SNE quant à lui est basé sur des procédures d'optimisation dont la solution optimale dépend des conditions initiales. En général, des initialisations différentes conduisent à des solutions différentes. La stabilité n'est pas garantie même en fixant la condition initiale, de sorte qu'une modification du nombre d'éléments peut conduire à des agencements sensiblement différents [?]. Quant au MDS, c'est une technique qui repose sur des points de repère ou de contrôle et qui a tendance à être stable tant que les points de contrôle et de repère restent fixes. Comme la position de chaque instance ne dépend que des points de contrôle (points de repère), si ces points ne changent pas, le mappage ne change pas non plus [?].

Nous pouvons donc affirmer qu'aucune méthode n'est parfaite : peu importe la réduction de dimension appliquée une perte de données est inévitable ce qui cause forcément l'apparition d'artefacts. Ce qui soulève la question suivante : Comment s'assurer de la qualité d'une projection ? Pour appréhender cela, nous passerons en revue plusieurs critères et méthodes d'évaluation de la qualité d'une projection dans le chapitre suivant.

2 Les critères d'évaluation des projections

Comme nous l'avons vu dans le chapitre précédent, la projection d'un jeu de données multidimensionnel passe par un processus de réduction de dimension. Or, celui-ci a un coût : il implique une perte de données qui va se répercuter lors de la représentation finale de celles-ci.La réduction induit une erreur : le stress qui a un impact sur l'approximation des distances au sein de la représentation.

Par définition, une bonne projection est une projection qui représente le plus fidèlement possible la structure sous-jacente du jeu de données. Par conséquent, une bonne configuration de projection est une configuration qui minimise le stress de la projection. En d'autres termes: plus les distances entre le jeu de données et la projection seront proches, moins il y aura d'erreurs. Ainsi, le but dans une projection est de minimiser le stress.
Mathématiquement parlant, le stress peut être défini comme la somme quadratique des écarts de distances :

$$\epsilon_\psi = \sum_{i,j}^{n} [(d_m(x_i, x_j) - d_p(y_i, y_j)]^2$$

2.1 Visualisation du stress

Le stress va produire des artefacts lors de la projection de données. Ceux-ci vont avoir un impact sur l'approximation et l'interprétation de la représentation graphique. Il y a plusieurs façons de le quantifier :

Les mesures peuvent être locales, c'est-à-dire qu'elles sont visualisées en chaque point et aident à expliquer la projection. C'est possible en utilisant un diagramme de Shepard [?] [?]. D'autres méthodes utilisent un jitter disc autour de chaque point. Cela permet de visualiser le stress, mais ne donne pas des informations sur les origines de celui-ci [?].

Il est également possible de le visualiser en utilisant un encodage colorimétrique, qui en fonction d'une couleur et de son intensité indiquera les points les moins fiables de la projection [?]. On peut aussi avoir recours à différents types d'interpolation de la couleur entre les points, selon différents paramètres, ce qui forme une carte de précision [?].

Une méthode pour mettre en avant les artefacts topologiques(faux voisinages et déchirures) a été mise en place par Lespinats et Aupetit [?] : l'idée est de visualiser deux mesures de stress simultanément à l'aide d'une échelle de couleur 2D uniforme. Cela permet d'utiliser la projection malgré les erreurs de positionnement des différents points: cependant la mesure de qualité ne garantit pas qu'on puisse exploiter la projection(surtout pour les tâches de clustering visuel) .
Il y a également, des méthodes interactives avec différentes vues possibles(stress) globaux, faux voisinages, déchirures ...) selon les différents niveaux de granularité (point seul ou groupes de points) [?] [?]. De plus, en fonction du type d'artefacts, différentes mesures d'erreurs sont proposées : coloration interpolée ou edge bundling(arc compact) pour, par exemple relier les points et faire le parallèle pour montrer les déchirures sur la projection [?] [?] [?]. Les arcs sont aussi colorés de façon différente en fonction de l'intensité des erreurs.
Ce système est interactif, car l'utilisateur peut choisir manuellement un groupe de point à étudier afin de visualiser seulement les artefacts qui lui sont relatifs. Il y a aussi beaucoup de vues statiques qui peuvent l'aider à filtrer.

2.2 Les critères de mesure

Pour évaluer la qualité d'une projection, plusieurs approches se basant sur différents ritères existant. Ces critères peuvent être relatifs à la structure des données projetées avec le taux d'oublier de la projection (outlying), le taux de cluster et à leurs qualités(visual clustering, class consistency, cluster separator, class density), et à la forme de la projection (clumpy, skinny). Nous allons les détailler au cours de ce chapitre.

Clustering

Le clustering sert à partitionner les données en groupe homogènes. En d'autres termes, cela consiste à évaluer si le jeu de données contient des "groupes"(ou clusters) ainsi que leur qualité (c'est-à-dire s' ils sont séparés et facilement identifiables). Pour cela, il existe plusieurs critères.

- **Class consistency :** Tout d'abord, il y a la mesure de la *class consistency* [**?**]. Dans un jeu de données,chaque classe est composée d'un sous-ensemble de l'espace de données qui lui est assigné. Ainsi chaque point est assigné à une classe. L'ensemble des classes du data-space est appelé *class structure*. L'objectif est donc de vérifier si la séparation des classes est correcte en fonction des dimensions étudiées, sachant qu'une bonne représentation visuelle d'une "Class structure" doit être fidèle à cette même structure de classe. Pour déterminer cela, nous pouvons le calculer selon la règle de la *distance to centroid*: c'est-à-dire que chaque point d'une classe doit être à une distance inférieure de son centroïde(i.e le centre de sa classe) qu'à celui d'une autre classe. Or c'est une des propriétés qui est très souvent perdue lors de la projection. Par conséquent, si la *class consistency* est bonne, les classes seront situées à des régions visuellement séparées dans la projection graphique, les clusters seront donc plus facilement discriminables et la projection potentiellement de meilleure qualité.

- **Class density :** Pour justifier de la qualité du clustering d'une projection, nous pouvons également mesurer la densité de classe (ou *class density*). Le principe est de retenir les graphes avec le moins de chevauchement [**?**]. Ainsi, lors d'une projection, les points appartenant à un cluster forment une image, c'est -à -dire que chaque classe forme une image. L'algorithme se fait en fonction de la densité basée sur le voisinage : pour chaque pixel, la distance de son voisin le plus proche de la même classe est enregistrée. Puis, la densité locale est calculée dans une sphère de rayon de la distance maximum possible. Le chevauchement global des classes est ensuite estimé en calculant la somme de la différence de chaque paire de pixels à la valeur absolue. Puis la visualisation avec le chevauchement le plus faible sera retenue.

$$CDM = \sum_{k=1}^{m-1} \sum_{l=k+1}^{m} \sum_{i=1}^{P} \|P_k^i - P_l^i\|$$

- **Histogram density :** Il y a aussi la mesure de l'*histogram density* [**?**] [**?**] qui est une mesure de l'entropie de la projection (i.e l'information moyenne de différentes portions de la projection découpée en grille). La projection est séparée en plusieurs barres (à l'image des histogrammes). Dans chacun de ces bins on comptabilise le nombre de points et leur étiquette de classe. L'entropie de chaque bin se calcule de cette façon.

$$H(p) = -\sum_{c} * \frac{P_c}{\sum_c * P_c} * \log_2 * \frac{P_c}{\sum_c * P_c}$$

  Ainsi, si tous les points d'une barre sont de la même classe, alors l'entropie est égale à zéro. Par conséquent le clustering sera de bonne qualité.

Outlying

Le taux d'outliers est important à définir pour justifier de la qualité du clustering. Un *outlier* est une donnée aberrante. Pour déterminer si un point est un outlier, il y a plusieurs méthodes.

Critères utilisant les graphes :

- Une des façons de raisonner en graphe et d'utiliser le concept de *Minimum Spanning Tree*(MST) [**?**] , c'est-à-dire l'arbre le moins long que l'on peut créer à partir du nuage de points de la

projection. En prenant comme référentiel le MST, les outliers sont soit des points qui sont localisés aux extrémités de l'arbre ou en intérieur dans des régions relativement vides. Ainsi, on peut les définir comme des nœuds de degrés 1, et dont la somme des poids des arêtes adjacentes est supérieure à 'w' selon la formule suivante

$$\omega = q_{75} + 1.5(q_{75} - q_{25})$$

. Puis, nous pouvons calculer le taux d'outliers dans la projection (outlying) en divisant la longueur totale de tous les outliers par la longueur totale de l'arbre selon la formule suivante :

$$c_{outlying} = length(T_{outliers})/length(T)$$

.

- Il y a également la méthode de l'*isolation Forest*. Dans celle-ci, l'algorithme choisit une caractéristique du jeu donnée et une valeur *split* comprise entre le maximum et le minimum des valeurs. Cette étape est effectuée pour toutes les observations. Par la suite la moyenne des tous les arbres est faite pour construire la forêt.L'apprentissage de l'algorithme consiste à comparer les observations avec la *splitting value* dans un nœud. Ce nœud comprendra également deux sous-noeuds dans lesquels on fera aussi la même comparaison. Ainsi le nombre de *splitting* est égal à la longueur du chemin. Toutes les comparaisons auront un score allant de 0 à 1. 0 signifiant "normalité", et 1 signifiant qu'il y a un grand nombre d'outlier[**?**]. Elle est implémentée en python avec scikit Learn.

Puis, nous pouvons calculer le taux d'outliers dans la projection (outlying) en divisant la longueur totale de tous les outliers par la longueur totale de l'arbre selon la formule suivante.

$$c_{outlying} = length(T_{outliers})/length(T)$$

.

Le Local Outlier Factor(LOF): Pour ce critère, on considère un outlier selon son voisinage local[**?**]. Le LRD(Local Reachability Density) de chaque point est comparé avec le LRD de ses voisins. Puis on calcule le ratio de la moyenne des LRD des voisins du point sur le LRD du point lui-même. Si le Local Outlier Factor est supérieur à 1 alors le point est un outlier. Cette mesure est très utile pour détecter des outliers dans les clusters extrêmement denses. Néanmoins cette valeur étant un ratio et non un seuil, elle est parfois compliquée à interpréter en fonction de la problématique de recherche.

Critères de formes

Les informations sur la forme de la projection sont également très importantes.[**?**] La forme est souvent définie par le critère *skinny*. Ce critère représente grossièrement la finesse de la courbe. Celui-ci est défini en calculant le ratio aire/périmètre d'une projection (avec normalisation)

$$C_{skinny} = 1 - \sqrt{4\pi area(A)}/perimeter(A)$$

Dans le cas où le chemin le plus court d'un Minimum Spanning Tree est quasiment aussi long que la somme des arêtes de l'arbre d'origine, la courbe peut être qualifiée de *stringy*

$$C_{stringy} = diameter(T)/length(T)$$

Critère de densité

Dans une configuration éparpillée (scattered) il y a également le critère de densité qui importe. C'est la distribution des arêtes du Minimum Spanning Tree qui va donner des informations sur

la densité relative de points. Le calcul de cette densité se base sur la mesure des quantiles de la longueur des arêtes :

$$c_{skew} = (q_{90} - q_{50})/(q_{90} - q_{10})$$

Ainsi, si la dimension d'un Minimum Spanning Tree est extrêmement asymétrique, les clusters y seront mal représentés. Dans ce cas, elle est qualifiée de *clumpy*. [**?**]

Score de Projection *(seulement pour l'ACP)*

Les informations d'une projection par ACP sont mesurées en comparant la variance totale du jeu de données et celle qui a été retenue après la réduction de dimension. Ce qui la rend très dépendante du nombre de dimensions du jeu de données de base.
Ainsi, contrairement aux petits jeux de données, il est quasiment impossible que les trois composantes principales d'un grand jeu de données retiennent 100 pourcents(ou quasiment 100 pourcents) de la variance. Et ce, même si le jeu de données est composé de variables non aléatoires avec des structures très facilement interprétables. Le but de cette méthode est de quantifier l'informativité de la projection non pas par la variance finale, mais par l'excès de variance de ce qui est attendu après réduction d'un dataset aléatoire. Pour calculer le score de projection: on calcule la variance totale qui est retenue par les 3 principaux composants. Ensuite, on estime la valeur de ces mêmes entités, mais pour un jeu de variables complètement aléatoire.
Enfin, on calcule le score de projection en faisant différence entre la racine carrée de la quantité observée et de la racine carrée de la valeur attendue (celle du random dataset). Ainsi, si le résultat est grand(s'éloigne positivement de zéro) cela veut dire qu'il y a plus d'informations (relatives au calcul de la variance) pour le vrai jeu de donné que pour celui du jeu de données aléatoire. Par conséquent, la projection est plus susceptible de proposer des structures fidèles et intéressantes, qui ne sont pas dues au hasard[**?**].

2.3 Conclusion de la partie

Ces mesures de qualité sont donc utiles pour les personnes non spécialistes, ce qui leur permet d'appréhender au mieux la visualisation obtenue après projection. Elles peuvent également être utilisées de façon automatique dans des algorithmes qui permettent de trier et de déterminer si la projection qui va être affichée est fidèle au jeu de données[**?**].
Comme nous l'avons brièvement abordé dans l'introduction, les facteurs numériques à eux seuls ne suffisent pas à l'interprétation d'une projection : il faut rajouter le facteur humain à l'équation. Malgré l'utilité de toutes ces mesures, le jugement humain reste toujours le plus important[**?**]. De plus, c'est également l'être humain qui a pour rôle d'interpréter la projection. C'est pour cela qu'il existe des évaluations à faire passer aux humains, qui servent à mesurer l'impact des techniques de projection, la qualité des graphiques et leur compréhension par l'utilisateur. Par exemple ,nous pouvons tester directement les utilisateurs avec plusieurs types de tâches telles que le "data outlier validation", le "clustering validation", le "cluster énumération", et le "class outliers validation" [**?**]. Ces tests consistent en la détection des structures et des données aberrantes dans les projections. Ainsi si une projection a une bonne qualité (suivant les critères numériques) et permet aux utilisateurs de discriminer rapidement et correctement les structures,elle tend alors vers ce que l'on peut considérer comme étant une "bonne projection".

Conclusion

Comme ça a été mentionné dans l'état de l'art la visualisation des grands jeux de donnée souffre de cette malédiction de la dimensionalité. Le problème des projections dans ce cas précis n'est pas celui de projeter les données, mais celui de les projeter fidèlement. C'est-à-dire, la difficulté pour la méthode à respecter le rapport des distances des points entre le jeu de donné original (à n-dimensions) et la projection finale (qui réduit ce dernier à deux ou 3 dimensions). Plus le nombre de dimensions augmente, plus il devient compliqué de respecter les distances après réduction.
Le choix d'une méthode de projection par rapport à une autre dépend d'autant plus de la sémantique et de la structure des données. Ce choix dépendra aussi de ce que l'on souhaite observer : voulons-nous mettre en valeur les classes ? Voulons-nous détecter des valeurs aberrantes? En fonction de cela, le choix de la méthode et de son paramétrage peut grandement différer. Une fois le jeu de données projeté, il faut s'assurer de la qualité de cette projection. Pour cela nous disposons de nombreux critères qui peuvent donner un indice sur le clustering, outlying et la densité de la projection. Certains critères sont même propres à certaines méthodes(comme le projection score pour l'ACP). Encore une fois, en fonction de ce que nous voulons observer certains critères seront plus utiles que d'autres. De plus, ces critères ne font pas tout : d'une certaine façon, une projection se doit d'être à la fois la plus compréhensible par l'Homme et la plus fidèle possible au jeu de données qu'elle représente.

C'est dans ce but, que dans la partie réalisation de cette UE, nous souhaitons mettre en place une évaluation des différentes méthodes de projections citées. Pour faire cela, nous envisageons de procéder de la façon suivante:
Au vu de nos nombreuses lectures et des résultats sur notre jeu de données tests (Globe), nous avons déduit que la méthode ClassNeRV est la plus performante et donne les résultats les plus satisfaisants. Par conséquent, nous souhaitons effectuer un étalonnage des différentes autres méthodes de projection, en l'utilisant en tant que référentiel.
Notre idée est donc la suivante : lorsqu'un jeu de donnée sera projeté avec une autre méthode de projection, le résultat obtenu sera comparé à la projection de ce même jeu de donnée obtenue via ClassNerV. Puis, en fonction des divergences entre ces deux jeux de données projetés, certains biais seront mesurés et affichés. De plus dans l'objectif d'être le plus précis possible, nous envisageons également de mesurer les différents critères de la projection(clustering, outlying, forme...) et de croiser les résultats obtenus à ceux de la comparaison des deux projections. Nous pensons aussi qu'il serait judicieux de mettre en place un système de "scoring", qui permettrait de rendre plus intuitive l'appréhension des résultats relatifs à la qualité de la visualisation.
Le but de cette démarche est de proposer un outil capable de comparer sa méthode de projection à d'autres via un tableau de scoring pour se rendre compte de la pertinence et de la fidélité des diverses méthode de projection.

Comme ça a été mentionné dans l'état de l'art la visualisation des grands jeux de donnée souffre de cette malédiction de la dimensionalité. Le problème des projections dans ce cas précis n'est pas celui de projeter les données, mais celui de les projeter fidèlement. C'est-à-dire, la difficulté pour la méthode à respecter le rapport des distances des points entre le jeu de donné original (à n-dimensions) et la projection finale (qui réduit ce dernier à deux ou 3 dimensions). Plus le nombre de dimensions augmente, plus il devient compliqué de respecter les distances après réduction. // Puis suite du paragraphe.

Méthodologie : Nous avons donc mis en place un programme avec un système de score qui classe l'efficacité des différentes méthodes de réduction/projection pour un même jeu de donné. C'est-à-dire que le jeu de donnée va passer une fois dans chaque algorithme, puis le résultat sera mesuré. Pour mesurer ces résultats nous avons utilisé différents critères issus de la librairie R « clusterCrit » et scikit-learn. ClusterCrit est une librairie R qui fournit une liste de critéres permettant d'attester de la qualité interne des clusters et une liste de critères qui permet de mesurer la similarité entre deux

partitions. Ces critères prennent seulement en compte la répartition des points dans les différents cluster et ne permettent pas de mesurer la qualité de la distribution.

3 Implémentation

3.1 Jeux de données

Pour comparer les différentes méthodes nous avons créer plusieurs jeux de donnés qui présentent les caractéristiques suivantes :

- deux clusters superposés avec un seul
- trois clusters bien distincts
- trois clusters superposés
- un cluster entouré de quelques outliers
- trois cluster entourés d'outliers

Pour les créer nous avons utilisé la méthode $make_blobs de la librairie sklearn. Les paramtres fixes sont : n_samples qui repr sente le nombre de points total qui sera galement r partient les diff rents clusters. Nous l'avons fix mille$. Nous avons fait varier les paramètres suivants pour obtenir la répartition souhaitée :

- *centers* qui détermine le nombre de centres à génerer

- $cluster_std$ qui d termine l'cart type des clusters

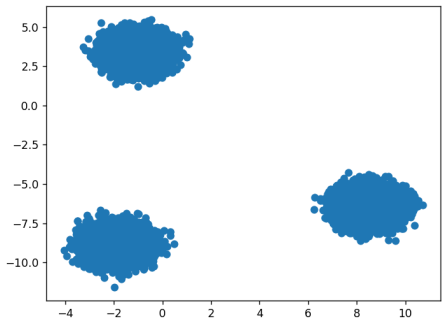

Figure 5: Trois clusters distincts

•

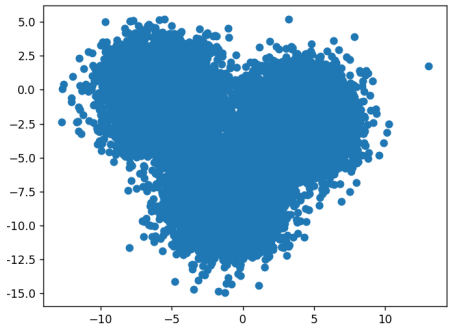

Figure 6: Trois clusters se chevauchant

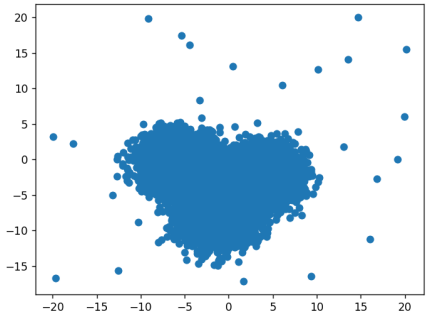

Figure 7: Trois cluster distincs avec des outliers

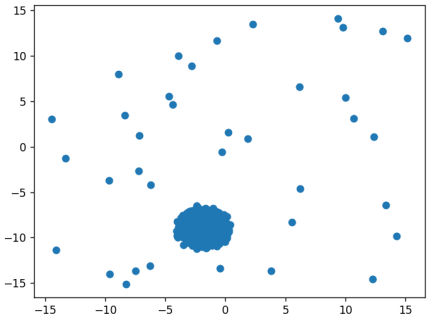

Figure 8: Un cluster avec plusieurs outliers

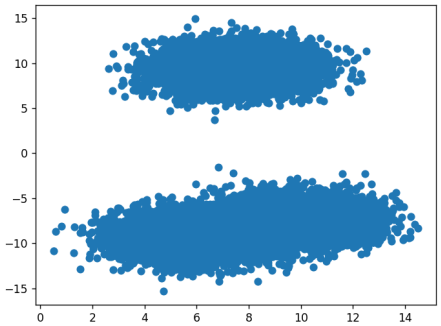

Figure 9: Deux clusters se chevauchant et un cluster seul

3.2 Méthodes

La librairie *rpy2* est utilisée pour pouvoir charger et utiliser des librairies R sous sous Python. Comme précisé plus haut nous appliquons cela à la librairie *clusterCrit* . Tous les critéres d'intérêts sont listé dans la variable globale *crit*, une liste.

La fonction *calculate* va utiliser chaque méthode de réduction sur le jeu de donnéen, le resultat de cette réduction sera stockée dans la variable*p2* à laquelle on appliquera la fonction *IntVector*. Comme éléments de comparaison, on applique Kmean sur le jeu de donnée originel. Une fois ces deux opération effectuées, il ya deux variables $r_dsquireprsentelersultatdelamthodederduction, et$

Les résultats seront stockés sous la forme d'un tableau de la forme suivante...

4 Experimentation