Université de Bordeaux

Master Bio-informatique

Projet de 4TBI904U Données : De l'Entrepôt à l'Analyse

Visualisation d'Information - Analyse critique d'algorithme de réduction de dimension

Auteurs :

Elsa Coutaud
Mallory Le Corre
Noah Vanney
Corentin Serain

Année 2023-2024

Table des matières

Introduction

L'algorithme t-SNE (t-Distributed Stochastic Neighbor Embedding) est une technique permettant la réduction de dimensionnalité afin d'observer des données de grande dimension dans des espaces de deux ou trois dimensions. Cette méthode a été introduite par Laurens van der Maaten et Geoffrey Hinton en 2008. Cette technique est dérivée de la technique SNE (Stochastic Neighbor Embedding) présentée par Hinton et Roweis en 2002. Malgré de bons résultats de visualisation, l'algorithme SNE présentait certains défauts tels qu'une fonction de coût difficile à optimiser et des problèmes d'encombrement qui est un problème de conservation de la distance avec tous les voisins. La technique t-SNE permet d'atténuer ces problèmes.

Lorsque les données traitées présentent de grandes dimensions, des problèmes de dimensionnalité sont souvent associés ce qui complexifie la visualisation et la compréhension de celle-ci. La détection des groupes ou encore des différentes structures devient alors plus difficile. La réduction de dimensionnalité de t-SNE permet donc la représentation des données dans un espace de dimension inférieure tout en sauvegardant au mieux les relaxions entre les points d'origine. La nouvelle représentation obtenue, même si elle respecte la proximité qui existe entre les points, ne respecte pas forcément les distances et les densités de distribution des données originales. L'algorithme va minimiser ces changements lors de son application. Différents paramètres sont importants lors de l'utilisation de t-SNE, en commençant par la perplexité qui est un paramètre très important de l'algorithme, car il permet de faire varier l'équilibre entre l'importance des structures locales et celle des structures globales. Plus elle sera élevée plus t-SNE lors de son exécution accordera de l'importance aux structures globales et inversement si elle est plus faible, en fonction des données traitées elle sera adaptée aux besoins de l'utilisateur.

En résumé, cet algorithme est un outil puissant permettant une visualisation significative des données de haute dimension tout en conservant les relations entre les points. La maîtrise ainsi que la compréhension de ces différents paramètres est un enjeu majeur pour pouvoir exploiter tout le potentiel de ces algorithmes. Lors de ce projet plusieurs jeux de données seront utilisés en faisant varier certains paramètres tels que la dimension de ceux-ci, puis l'algorithme t-SNE sera, lui aussi, utilisé de plusieurs manières en fonction des paramètres qui lui seront attribués.

Chapitre 1

Explication de l'algorithme

1.1 Fonctionnement

Le but de t-SNE est de construire une nouvelle représentation des données de telle sorte que les données proches dans l'espace initial aient une probabilité élevée d'avoir une représentation proche dans l'espace réduit et inversement. Pour ce faire, on peut décomposer l'algorithme en trois étapes.

Dans un premier temps t-SNE commence par convertir les distances euclidiennes entre les données haute dimension en probabilité conditionnelle qui représentent les similitudes. Pour cela, Il va centrer autour de chaque point x_i une distribution gaussienne. Cette distribution modélise la probabilité que d'autres points soient situés à différentes distances de x_i. Pour chaque autre point que l'on nommera x_j, l'algorithme va mesurer la densité de probabilité sous la distribution gaussienne centrée sur x_i. La similarité du point x_j avec le point de données x_i est la probabilité conditionnelle, $p_{j\,|\,i}$, que x_i choisisse x_j comme voisins si les voisins étaient choisis proportionnellement à leur densité de probabilité sous la gaussienne centrée sur x_i. Ces mesures sont ensuite normalisées afin d'obtenir une somme des probabilités conditionnelles égale à 1 pour chaque point. Ainsi, pour les points proches, $p_{j\,|\,i}$ sera relativement élevé, alors que pour des points très éloignés, $p_{j\,|\,i}$ sera extrêmement petit. Mathématiquement, cette probabilité conditionnelle est donnée par l'expression :

$$p_{i,j} = \frac{\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)}{\sum_{k \neq l} \exp\left(-\frac{\|x_k - x_l\|^2}{2\sigma^2}\right)}$$

où σ est la variance de la gaussienne centrée sur le point de données x_i.

Dans un deuxième temps, l'idée va être de représenter nos données dans un espace de dimension plus réduite. Au début, nous ne connaissons pas les coordonnées idéales pour ces nouveaux espaces. Par conséquent, pour chaque point de données, nous attribuons initialement des coordonnées aléatoires dans cet espace réduit. Ensuite on calcule les similarités comme dans l'espace haute dimension, mais utilisant cette fois ci une distribution t-student. Nous obtenons la probabilité $q_{j\,|\,i}$ qui est la probabilité que y_i choisisse y_j comme voisin si les voisins étaient choisis proportionnellement à leur densité de probabilité sous la distribution t-student centrée sur y_i. Mathématiquement, cette probabilité se traduit :

$$q_{i,j} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}$$

Si les points de y_i et y_j modélisent correctement la similarité entre les points de données à haute dimension x_i et x_j, les probabilités conditionnelles $p_{j\,|\,i}$ et $q_{j\,|\,i}$ seront égales. Une métrique qui permet d'évaluer la fidélité avec laquelle $q_{j\,|\,i}$ modélise $p_{j\,|\,i}$ est la divergence de Kullback-Leibler. La divergence de Kullback-Leibler est une mesure de la différence entre deux distributions de probabilités. La forme générale de la divergence KL entre deux distributions de probabilité P et Q est donnée par :

$$D_{KL}(P \,||\, Q) = \sum_i P(i) \log \left(\frac{P(i)}{Q(i)} \right)$$

Et le gradient de la divergence de KL par rapport à y_i se calcule par :

$$\frac{\partial D_{KL}(P \,||\, Q)}{\partial y_i}$$

t-SNE va minimiser la somme des divergences de Kullback-Leibler sur tous les points à l'aide d'une méthode de descente de gradients. Tant que la divergence de Kullback-Leibler (valeur seuil définie) ou que l'algorithme n'a pas convergé ou que le nombre maximal d'itérations n'est pas atteint, le point y_i sera déplacé selon :

$$y_i = y_i - \alpha \frac{\partial L(p, q)}{\partial y_i}$$

avec L correspondant à la divergence. Si L -> 0 alors $p_{j\,|\,i} = q_{j\,|\,i}$

1.2 Influence des paramètres

La perplexité σ est un paramètre à rentrer lors de l'utilisation de t-SNE. Il permet de définir la taille du voisinage considéré pour le calcul de similarité. En fait, si la valeur de perplexité est faible seuls les points très proches seront considérés comme voisins et auront une valeur de similarité non-nulle. Ainsi, la structure proche du point étudié sera bien conservée dans l'espace réduit. A l'inverse, une valeur de perplexité élevée définit un voisinage beaucoup plus grand et va donc mieux conserver la structure globale des données. En résumé, une perplexité faible va mieux conserver la structure locale en se souciant moins de la structure globale, tandis qu'une perplexité élevée fera l'inverse. Dans cette étude, il va donc être intéressant de tester différentes valeurs afin de comparer les résultats au niveau local et au niveau global.

Le nombre d'itérations n_iter est le nombre de fois que t-SNE va itérer. A chaque itération, t-SNE déplace les points dans l'espace réduit et ainsi réduit progressivement la divergence de KullBlack-Leibler. Le but est donc de choisir un nombre d'itérations qui va conduire à la divergence de Kullblack-Leibler la plus faible. Il est intéressant de tester différents nombres d'itérations parce que si le nombre d'itérations est trop faible alors on obtiendra un résultat incomplet. En revanche, si ce nombre est trop grand, alors cela signifie que l'algorithme aura convergé plus tôt et les itérations qui se situeront après seront inutiles.

Le paramètre n_components va définir le nombre de dimensions de l'espace réduit dans lequel seront représentés les points. Cette valeur doit être inférieure aux nombres de dimensions de l'espace initial pour observer la réduction de dimension. Ici, nous allons utiliser une valeur fixe égale à 2 ce qui nous permettra de visualiser en 2D nos points dans l'espace réduit.

Le paramètre métrique correspond à la technique utilisée pour mesurer la distance entre deux points dans l'espace en haute dimension. Par défaut, t-SNE utilise la distance euclidienne. Dans cette étude, nous allons garder cette technique qui est la plus répandue. Il est important de noter qu'étant donné que nous avons normalisé nos données entre 0 et 1, la différence entre ces techniques de mesure est négligeable.

Chapitre 2

Implémentation

L'implémentation de ce projet se présente en 4 fichiers python :
— *generate_ data.py* : Génération de données en haute dimension.
— *tsne.py* : Application de l'algorithme t-SNE sur les données en haute dimension et du calcul de différentes métriques.
— *metrics.py* : Implémentation des métriques.
— *main.py* : Execution du code.
Un fichier requierements.txt contient toutes les dépendances utiles au projet : NumPy, Pandas et Scikit-learn.

2.1 Générateur de données

L'objectif de cette partie est de générer des données à haute dimension de manière aléatoire, qui pourront ensuite être traitées avec t-SNE. Ces données suivent trois distributions différentes : normal, en groupe et uniforme, afin de simuler différentes situations. Les paramètres de ces générateurs peuvent être modifiés suivant la distribution, pour avoir une plus grande diversité dans les jeux de données. Pour cela, le code utilise les bibliothèques scikit-learn et NumPy.

2.1.1 Fonction de Normalisation

Afin que les valeurs des données soient à la même échelle, elles sont normalisées entre 0 et 1 avec la méthode MinMaxScaler de scikit-learn. Elle prend en argument le dataset à normaliser et retourne le nouveau jeu de données dans un tableau multidimensionnel. Cette fonction sera appelée dans les fonctions de génération.

2.1.2 Génération de Données suivant une Loi Gaussienne

La distribution normale (ou gaussienne) est l'une des distributions les plus répandues. En effet, la loi normale fait partie des lois de probabilité les plus utilisées dans la modélisation des phénomènes naturels. Elle peut être caractérisée par une courbe en forme de cloche, symétrique par rapport à sa moyenne.

La distribution en groupe fait référence à la structure des données. Dans ce cas, les données seront générées de telle façon qu'elles peuvent être regroupées entre elles suivant leurs caractéristiques. Dans la fonction suivante, chaque groupe est généré en suivant une distribution normale.

La fonction `data_generation_mc` génère des données avec la fonction `make_classification` de scikit-learn. Make_classification utilise. Le but de cette fonction est de pouvoir générer des données avec plusieurs paramètres différents. Pour cela, elle prend en arguments des listes de valeurs pour chaque paramètre pouvant être changé : `samples` correspondant au nombre d'échantillons, `features` correspondants au nombre de caractéristiques, `classes` correspondants au nombre de groupes, et `informative` la proportion de caractéristiques qui sont prises en compte dans la création des groupes. Ces arguments sont sous forme de listes afin que la fonction puisse parcourir chaque valeur afin de créer toutes les combinaisons de paramètres possibles grâce à la fonction product de la bibliothèque itertools. Elle prend aussi en arguments le dossier qui contiendra les données et le nombre de répétitions pour chaque combinaison de paramètres. Pour tous les jeux de données qui n'ont qu'une classe, ils sont classés comme une distribution normale.

2.1.3 Générateur de Données avec Distribution Uniforme

Cette distribution suit une loi de probabilité dans laquelle chaque valeur possible ont la même probabilité.

De la même façon que pour la fonction précédente, la fonction `data_generation_uni` prend en arguments le dossier dans lequel les CSV seront sauvegardés et les listes de valeurs pour les paramètres pouvant être changés : `rep`, `samples` et `features`. Les données uniformes sont générées grâce à la bibliothèque NumPy et la fonction `uniforme`.

2.1.4 Sauvegarde des fichers

Pour les deux fonctions de génération de données, les ensembles de donnée sont sauvegardés au format CSV dans des dossiers spécifiques à la distribution. De plus, ces CSV sont nommés suivant les combinaisons de paramètres utilisées et les noms des fichiers crées sont enregistrés dans un dataframe. Cela permet de fragmenter le pipeline : les données étant sauvegardées sur l'ordinateur, il est possible en cas de problème de reprendre à cette étape.

2.2 Algorithme t-SNE

Cette partie consiste à implémenter des fonctions qui permettent d'appliquer l'algorithme t-SNE aux données générées précédemment.

Une première fonction, nommée `tsne`, permet d'appliquer l'algorithme t-SNE sur un fichier de données selon différents paramètres et de sauvegarder les résultats dans des fichiers au format CSV. Cette fonction prend en argument les données sous forme de dataframe, le dossier où seront créés les fichiers CSV, ainsi que les listes de valeurs de paramètres de l'algorithme t-SNE (perplexité, nombre d'itérations). Grâce à la fonction `product` de la bibliothèque itertools, toutes les combinaisons de paramètres sont parcourues. L'algorithme t-SNE est ainsi appliqué sur un même fichier avec chaque combinaison de paramètres. Pour cela, la fonction `TSNE` de la bibliothèque scikit learn est utilisée.

La seconde fonction, appelée `tsne_metrics`, permet d'appliquer la fonction précédente sur l'ensemble des données suivant la distribution donnée en paramètre. Pour cela, la fonction prend également en paramètre un dataframe avec les noms des fichiers CSV de données à haute dimension. Cette fonction permet également de calculer des métriques sur les fichiers de données à hautes dimensions et les fichiers de résultats t-SNE qui leurs sont associés. Les résultats de ces métriques sont ensuite enregistrés dans un fichier CSV avec les paramètres utilisés dans la génération de données et dans l'algorithme t-SNE (voir section 2.3 Benchmark).

2.3 Benchmark

L'objectif de cette section est d'effectuer une étude qualitative de la réduction de dimensionnalité après l'application de t-SNE.

2.3.1 Nombre moyen de voisins communs

Cette étude utilise le nombre de voisins communs. Elle permet de comparer les relations entre les points avant et après déformation, fournissant ainsi un indice de la qualité de t-SNE. Cette fonction prend en entrée le dataframe des valeurs initiales, le dataframe des résultats t-SNE ainsi que la valeur de k. Pour les deux ensembles de données, la fonction va chercher les k plus proches voisins avec la fonction *"NearestNeighbors"* de sklearn. Ensuite, une moyenne du nombre de voisins communs entre l'espace initial et l'espace réduit est calculée et retournée par la fonction.

2.3.2 Ratios de distances euclidiennes

Cette fonction calcule l'écart-type des ratios entre les distances euclidiennes des paires de points dans l'espace initial et l'espace réduit. Elle prend en arguments le tableau de données initial et le tableau de données des résultats t-SNE. La fonction utilise d'abord la fonction "*euclidean_ distance*" de *sklearn* pour

calculer les distances euclidiennes entre chaque paire de points dans les ensembles de données initiaux. Le ratio des distances entre l'espace réduit et l'espace initial est ensuite calculé. Une valeur égale à 0.00000000001 est ajoutée à l'ensemble pour éviter le cas peu probable d'une division par 0. L'écart-type est finalement calculé grâce à la fonction *std* de la bibliothèque NumPy.

Cette métrique peut donner une mesure de la variabilité entre les points dans l'espace réduit comparé à l'espace initial.

2.3.3 Utilisation du benchmark

Dans le script principal, ces fonctions sont appelées et appliquées pour chaque distribution directement après la générétion des résultats t-SNE pour un fichier de données. Pour la métrique du nombre moyen de voisins communs, elle est calculée avec un $K = 10$ et $k= 20$. Les résultats de chaque métrique sont stockés dans des fichiers CSV contenant chaque paramètre utilisé pour la génération de données et pour le t-SNE, ainsi que les résultats des métriques associées.

Chapitre 3

Résultats

3.1 Paramètres et nombre de fichiers générés

La section suivante présente les paramètres clés et le processus de génération des données aléatoires utilisées pour évaluer l'algorithme t-SNE.

3.1.1 Générateur de données

Les données utilisées pour l'évaluation de t-SNE ont été générées à l'aide d'un générateur de données aléatoires. Les paramètres du générateur ont été définis de manière à couvrir une grande quantité de possibilité, offrant une diversité dans le nombre d'échantillons, de caractéristiques, de classes et de niveaux d'informativité. Les paramètres spécifiques incluent :

Nombre d'échantillons : [100, 250, 500, 750, 1000, 2500, 5000]
Nombre de caractéristiques : [10, 20, 30, 40, 50]
Nombre de classes : [1, 2, 3, 4, 5, 8, 10, 12, 15]
Niveau d'informativité : [0.5, 0.75, 1]

Les données générées sont réparties en trois catégories : données suivant une distribution normale (data_norm), données en groupes (data_cluster), et données suivant une distribution uniforme (data_uni). En ce qui concerne les données en groupes ainsi que les données de distribution normale, tous les paramètres vue précédemment sont pris en compte lors de l'exécution de t-SNE. Le nombre de classes permet de définir si la distribution est normale (une classe) ou en groupes (plus d'une classe). Pour les données de distribution uniforme, uniquement le nombre d'échantillons ainsi que le nombre de caractéristiques sont prise en compte. Pour chaque combinaison de paramètres générée une répétition de 5 est appliquée, ce qui signifie que à partir des paramètres spécifiques il y a 5 fichiers générés.

Il est possible de calculer le nombre de fichier CSV en sortie du générater grâce au produit cartésien effectuer à partir des paramètres et de la répétition. Ainsi, avec les paramètres choisis nous avons comme produits cartésiens :

— Pour les données suivant une distribution normale : $7 \times 5 \times 1 \times 3 \times 5 = 525$ fichiers générés
— Pour les données suivant une distribution en groupes : $7 \times 5 \times 8 \times 3 \times 5 = 4200$ fichiers générés
— Pour les données suivant une distribution uniforme : $7 \times 5 \times 5 = 175$ fichiers générés

Le générateur de données produit alors au total 4900 fichiers de données à haute dimension.

3.1.2 Résultats t-SNE

L'algorithme t-SNE a été appliqué aux données générées en utilisant différentes combinaisons de paramètres. Les résultats ont été enregistrés sous forme de fichiers CSV dans des dossiers spécifiques pour chaque distribution. Les paramètres t-SNE ajustables étaient les suivants :

Perpléxité : [10,30,50]
Nombre d'itérations : [300,500,1000]

Les fichiers CSV résultants représentent une combinaison unique de paramètres. De la même façon que pour le générateur de données, avec le produit cartésien nous avons le nombre de fichiers générés après l'algorithme t-SNE : 4725 pour les données suivant une distribution normale, 37800 pour les données en groupes et 1575 pour les données suivant une distribution uniforme.

3.2 Métriques

Distribution Normale

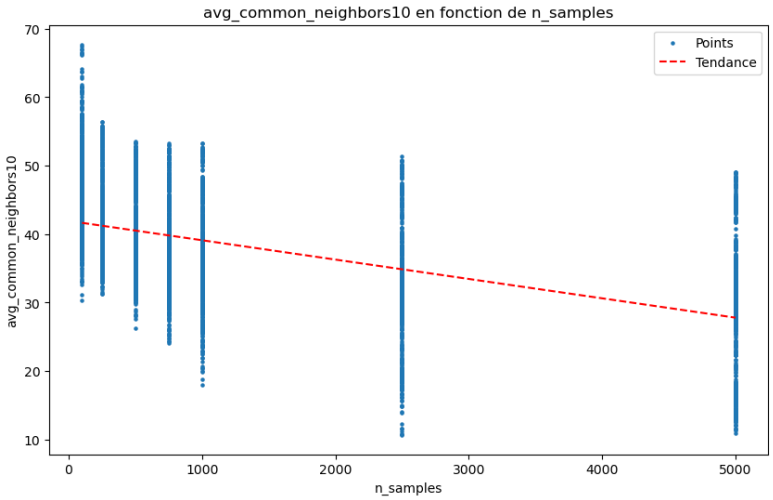

Figure 1 : Pourcentage de plus proches voisins en commun en utilisant k=10 en fonction du nombres de samples

On commence par observer le nombre de voisins en communs entre l'espace initial et l'espace réduit pour k=10 voisins en fonction du nombre de samples. On observe que le pourcentage de voisins en commun à tendance à diminuer au fur et à mesure que le nombre de samples augmente. En effet on voit que pour 100 samples on a un pourcentage de voisins commun situé entre 30 et 60% tandis que pour 5000 samples ce pourcentage est entre 10% et 40%(voir figure1). Cette tendance semble se confirmer avec l'utilisation de k plus proche voisins de k=40. Pour un nombres de samples de 100 on observe des pourcentages de voisins communs entre 45 pourcent et 65 tandis que pour un nombre de sample de 5000 ce pourcentage est de l'ordre de 5 à 30 pourcent (voir annexe).

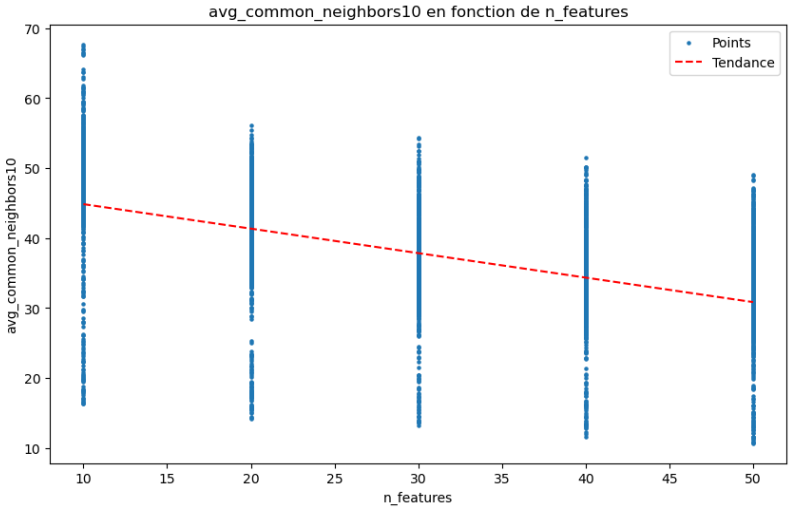

Figure 2 : Pourcentage de plus proches voisins en commun en utilisant k=10 en fonction du nombres de features

Avec la même idée que pour le nombre de samples, nous voulons observer comment varie le pourcentage de plus proche voisins communs en fonction nombre de dimensions dans l'espace d'origine. On observe que le pourcentage à tendance à diminuer au fur et à mesure que l'on augmente de le nombres de features allant d'un intervalle de 45 à 60 pourcent pour 10 features à 10 à 40 pourcent pour 50 features (voir figure 3). Nous observons des résultats similaires pour k=40 (voir annexe)

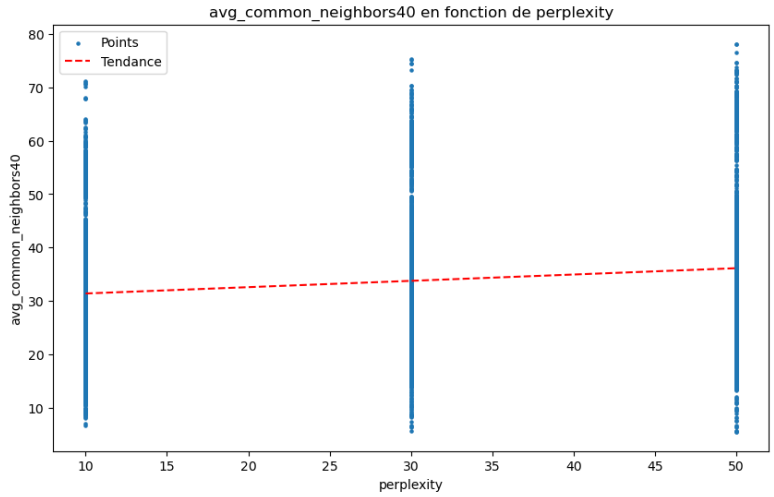

Figure 3 : Pourcentage de plus proches voisins en commun en utilisant k=40 en fonction de la perplexité

Lorsque l'on regarde l'evolution du pourcentage de voisins (avec k=40) communs en fonction de la perplexité, on voit que celui-ci à tendance à augmenter légèrement lorsqu'on eleve la valeur de perplexité. Si on regarde pour k=10 cette augmentation n'est pas nette et le pourcentage de voisins en commun reste stable. (voir annexe)

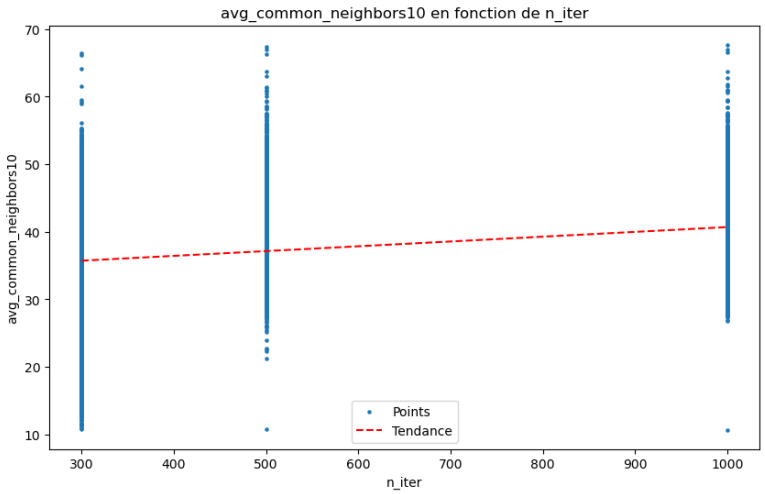

Figure 4 : Pourcentage de plus proches voisins en commun en utilisant k=10 en fonction du nombre d'itérations

Enfin sur ce graphique on observe l'évolution du pourcentage de voisins en communs (avec k=10) en fonction du nombre d'itérations. On voit que le pourcentage de conservation des voisins à tendance

à augmenter au fur et à mesures que l'on augmente le nombre d'itérations. Pour 300 itérations nous avons un minimum de pourcentage de voisins communs de 10% alors qu'il est presque de 30% pour 1000 itérations.

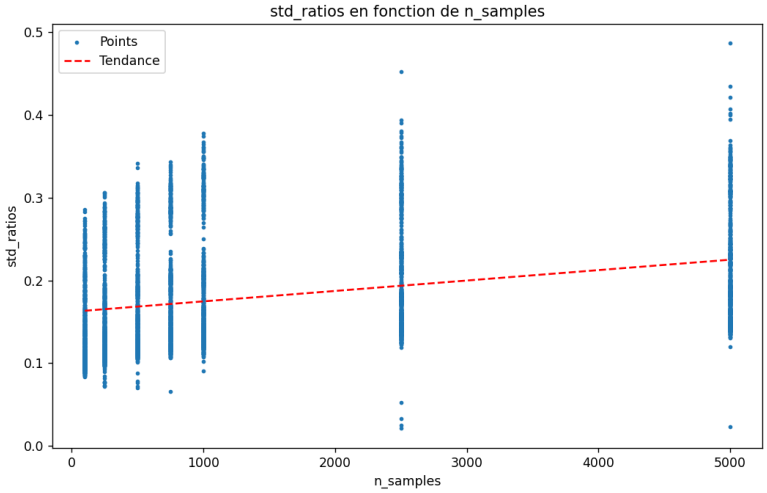

Figure 5 : écart type des ratios des distances euclidiennes en fonction du nombre de samples

On également calculer l'écart type des ratios des distances euclidiennes dans l'espace initial et dans l'espace réduit. D'après les observations, plus le nombre de samples augmentent et plus cette ecart type augmente

Distribution Uniforme

Les résultats obtenus pour la distribution uniforme sont très similaires à ceux obtenus pour la distribution normale. Vous pourrez trouver ces résultats en annexes.

Distribution en groupes

L'interêt de cette distribution est d'observer si nombres de classes présent dans l'espace d'origine va joué un rôle dans la performance de TSNE.

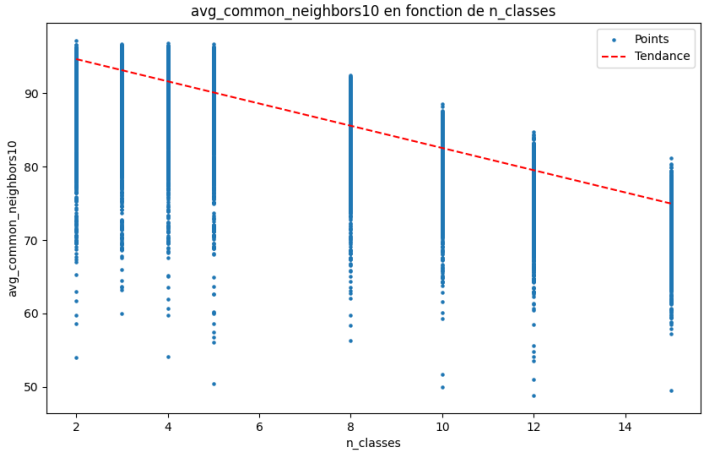

Figure 6 : Pourentage de voisins en commun en fonction du nombres de classes

Dans ce graphique on observe la conservation du voisinage (k=10= en fonction du nombres de classes dans l'espace d'origine. On remarque que lorsque le nombre de classes est plus grand le pourcentage de voisins en communs diminue.

3.3 Déformation

Afin de visualiser la déformation causée par l'algorithme t-SNE sur nos données, on peut tout d'abord observer les résultats de t-SNE sur un espace à deux dimensions (voir figure 5).

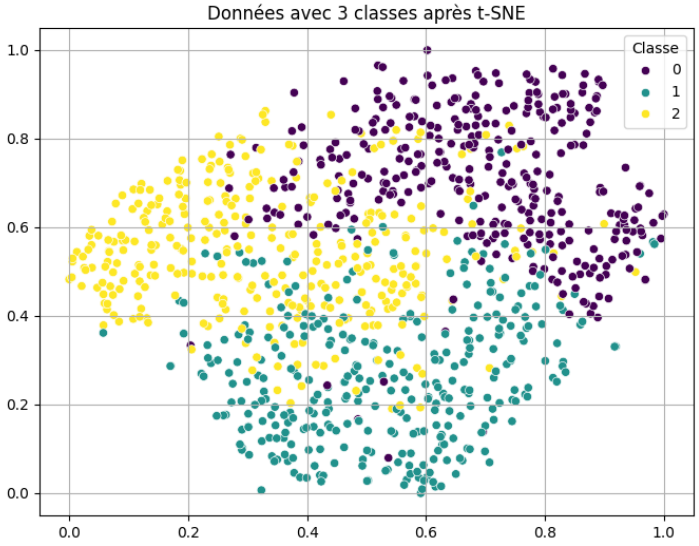

Figure 5 : Visualisation des données générées avec 3 classes, après l'algorithme t-SNE

Ce graphique présente le résultat de t-SNE sur des données générées en groupes (ici au nombre de trois). On peut observer que les classes sont divisées dans l'espace en 3 groupes bien distincts.

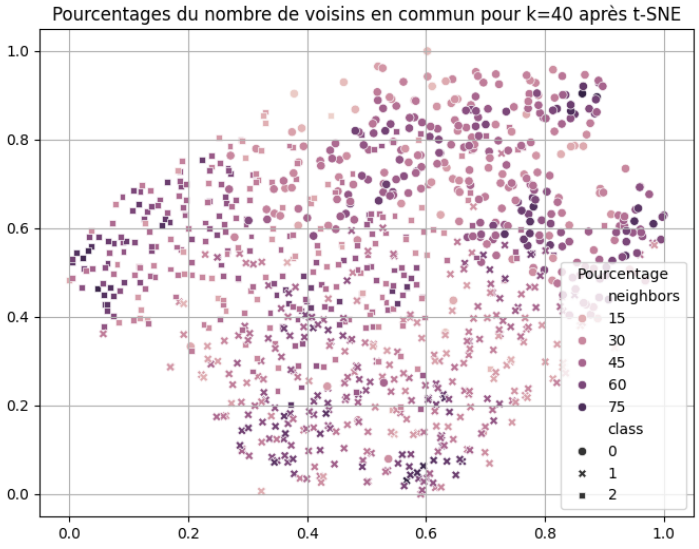

Figure 6 : Pourcentage de plus proches voisins en commun, pour k=40

Une autre façon de visualiser la déformation causé par l'algorithme t-SNE est d'observer la métrique des k plus proches voisins sur l'espace en deux dimensions créé par t-SNE. Sur ce graphique, on observe les mêmes données que dans la figure précédente mais cette fois ci on s'intéresse également au pourcentage de voisins en communs (avec k=40) pour chaque donnée. On peut voir une tendance qui se crée légèrement au niveau des endroits où les groupes se chevauchent. En effet on peut voir que à ce niveau les pourcentages de voisins en commun vont être autour de 15 à 30% alors qu'au centre des groupes on a une tendance plutot aux alentours de 75%.

Discussion

D'après les résultats obtenus le nombre de samples semble avoir un impact significatif sur la conservation des voisins. Cela paraît plutôt logique puisque plus il y a de points plus le nombre de voisins en commun sera faible. Le défaut de cette métrique est que si les voisins ne sont pas dans les k=10 mais dans les k=20 alors nous nous retrouverons avec un pourcentage de voisins en commun faible alors que la structure était conservé. Le nombres de samples n'est pas le seul paramètre qui va jouer un rôle dans la conservation de cette structure de voisinage. En effet pour 40 voisins, une perplexité haute semble améliorer la conservation du voisinage. Ceci peut s'expliquer par le fait que plus la valeur de perplexité est grande plus elle considère de voisins au moment des calculs de probabilités. Le nombre d'itérations influence également le conservation des voisins. Cependant cette influence semble se réduire après 500 itérations. Nous pouvons imaginer que l'algorithme converge à peu près à ce nombre d'itérations.

L'écart type des ratios des distances euclidiennes apporte également des informations complémentaires. D'après les résultats, ils semblent que plus le nombre de samples est important, plus cet écart type augmente. Or nous savons que plus cette écart type est petit, plus cela signifie que la représentation que réalise t-SNE dans l'espace réduit est stable. On peut donc dire que l'augmentation du nombres de samples reduit la précision de t-SNE. C'est également le cas pour l'augmentation du nombre de features.

Si on regarde comment se comporte t-SNE sur le jeu de données en groupes on voit que le nombre de classes présentes dans l'espace d'origine va également avoir un impact significatif. En effet plus le nombres de classes est important moins le voisinage est conservée. Cette observation va dans le sens ce que qu'on observe sur la figure 6. En réalité, plus il y a de classes dans un jeu de données plus il y a de chances que des points appartenant à des classes différentes se chevauchent. C'est précisement ce chevauchement qui semble baisser les performance de l'algorithme.

Chapitre 4

Conclusion

A travers cette étude, nous avons montré que l'algorithme reducteur de dimension est correct à conditions d'utiliser les paramètres adequat. La perplexité et le nombre d'itérations impact enormément la projection finale mais les paramètres de l'espace d'origine également. En effet on a vu que le nombres de samples, le nombres d'échantillons et le nombres de classes influencé significativement la qualité de l'algorithme. Il est important de noter que la stochasticité de la méthode t-SNE rend son évaluation difficile. Malgré la répétitions, les resultats sont parfois decevant. Pour palier à ce problèmes il aurait été interessant de réaliser cette études sur plus de valeurs de paramètres et en calculant plus de métriques afin de contrer ce biais

Bibliographie

[1] G.E. van der Maaten, L.J.P. ; Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9, 2008.

[2] L.J.P. van der Maaten. t-distributed stochastic neighbor embedding. https ://lvdmaaten.github.io/tsne/. Accès online le 12/12/2023.

[3] L.J.P. van der Maaten. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9, 2008.

[4] Ciccolella C. O. Anno R. Halpert R. Spidlen J. Snyder-Cappione J. E. Belkina, A. C. Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. *Nature Communications*, (10) :1–12, 2019.