

Reconnaissance des formes et méthodes
neuronales (RCP208)
Réduction de dimension

Nicolas Audebert et Michel Crucianu
(prenom.nom@cnam.fr)

<http://cedric.cnam.fr/vertigo/Cours/ml/>

EPN05 Informatique
Conservatoire National des Arts & Métiers, Paris, France

25 février 2026

Plan du cours

- 1 Motivation
- 2 Réduction linéaire de dimension
- 3 L'algorithme LLE
- 4 L'algorithme t-SNE
- 5 L'algorithme UMAP
- 6 Compléments : graphes, plongement spectral, complexe simplicial

Problèmes en grande dimension

Les jeux de données réels sont souvent non-structurées et de grande dimension (images, sons, séries temporelles...) :

- peu adaptés à la visualisation
- souffrent du **fléau de la dimensionnalité**
 - le nombre d'échantillons nécessaires pour représenter fidèlement l'espace de dimension d est impraticable,
 - Exemple : 100 points sur le segment $[0, 1]$ \implies 10^4 points sur le carré, 10^6 points sur le cube, 10^{2d} points sur l'hypercube en dimension d !
 - les composantes « bruitées » réduisent le pouvoir discriminant de la distance euclidienne
 - La distance minimale et la distance maximale entre un point de référence \mathbf{q} et les points d'un jeu de données $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ se rapprochent quand $d \nearrow$:

$$\lim_{d \rightarrow +\infty} \mathbb{E} \left(\frac{\max_{x_j} \|\mathbf{q} - \mathbf{x}_j\| - \min_{x_j} \|\mathbf{q} - \mathbf{x}_j\|}{\min_{x_j} \|\mathbf{q} - \mathbf{x}_j\|} \right) \rightarrow 0$$

La **réduction de dimension** a deux intérêts principaux :

- **visualiser** la structure du nuage des observations
- **simplifier** les données pour des traitements ultérieurs (par ex., classification)

Exemple pratique



- Une image est représentée par une matrice de 8×8 pixels
 - vecteur de dimension $d = 64$
 - peu visualisable

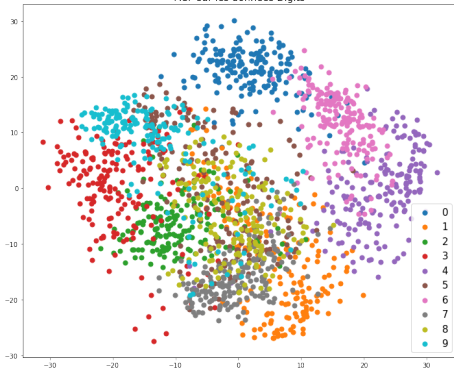
$$\begin{bmatrix} 0. & 0. & 11. & 12. & 0. & 0. & 0. & 0. \\ 0. & 2. & 16. & 16. & 16. & 13. & 0. & 0. \\ 0. & 3. & 16. & 12. & 10. & 14. & 0. & 0. \\ 0. & 1. & 16. & 1. & 12. & 15. & 0. & 0. \\ 0. & 0. & 13. & 16. & 9. & 15. & 2. & 0. \\ 0. & 0. & 0. & 3. & 0. & 9. & 11. & 0. \\ 0. & 0. & 0. & 0. & 9. & 15. & 4. & 0. \\ 0. & 0. & 9. & 12. & 13. & 3. & 0. & 0. \end{bmatrix}$$

Exemple pratique

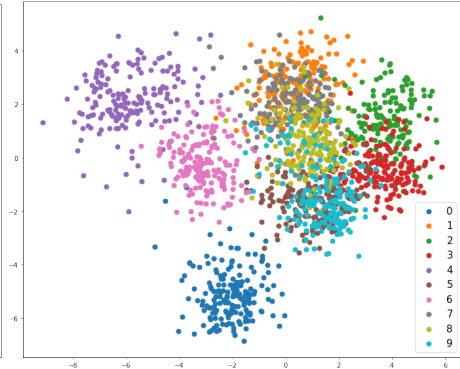


- Une image est représentée par une matrice de 8×8 pixels
 - vecteur de dimension $d = 64$
 - peu visualisable

ACP sur les données Digits



AFD sur les données Digits



Apprentissage de variété

(*manifold learning*)

- Les jeux de données en grande dimension sont difficiles à visualiser
- Hypothèse : la dimension observée des données est artificiellement élevée
 - Les données sont engendrées par un ensemble de variables **intrinsèques** (des « degrés de liberté »)
 - On suppose qu'il y a moins de degrés de liberté que de variables dans les observations
 - Autrement dit, les variables observées sont partiellement redondantes ou superflues

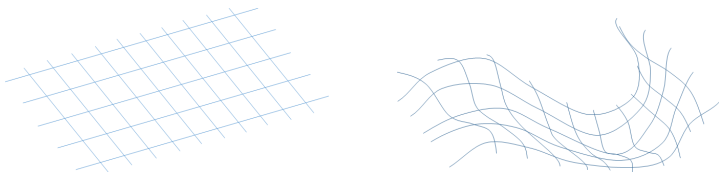
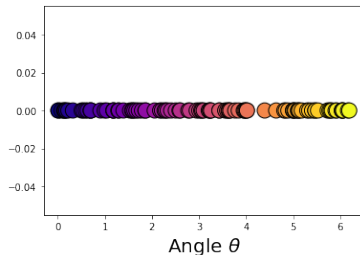
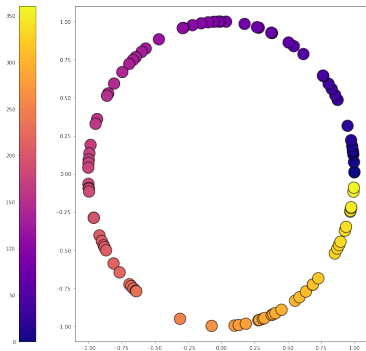


FIG. – Sous-espace intrinsèque linéaire (gauche) ou non linéaire (droite)

Apprentissage de variété : exemple

- Cercle : nuage d'observations bidimensionnel (x, y)
 - tous les points vérifient la contrainte : $x^2 + y^2 = 1$
- En reparamétrisant les observations
 - $x = \cos \theta, y = \sin \theta$

⇒ les observations se trouvent sur une **variété** unidimensionnelle obtenue en faisant varier l'angle dans l'intervalle $[-\pi, +\pi]$

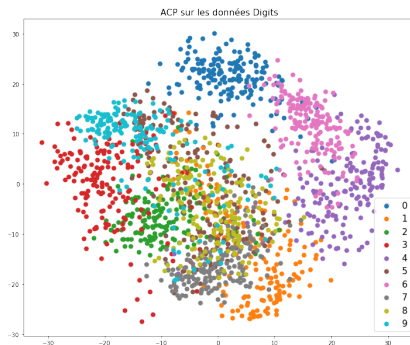


Plan du cours

- 1 Motivation
- 2 Réduction linéaire de dimension**
- 3 L'algorithme LLE
- 4 L'algorithme t-SNE
- 5 L'algorithme UMAP
- 6 Compléments : graphes, plongement spectral, complexe simplicial

Analyse en composantes principales

- \mathbf{R} : matrice de n lignes (une par observation) et d colonnes (une par variable)
- Analyse en composantes principales du nuage d'observations : rechercher le sous-espace de dimension k engendré par les k vecteurs propres \mathbf{u}_α associés aux k plus grandes valeurs propres λ_α de la matrice $\mathbf{X}^T\mathbf{X}$, avec
 - $\mathbf{X}^T\mathbf{X}$ est la matrice des covariances empiriques pour l'ACP centrée
 - $\mathbf{X}^T\mathbf{X}$ est la matrice des corrélations empiriques pour l'ACP normée



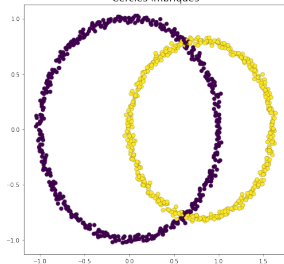
Analyse factorielle discriminante

- On suppose disposer d'une variable nominale « classe » $Y \in \{1, \dots, q\}$
- Les covariances entre variables se décomposent en deux termes : $\mathbf{S} = \mathbf{E} + \mathbf{D}$
 - \mathbf{E} est la covariance inter-classes, \mathbf{D} est la covariance intra-classes
- Analyse factorielle discriminante : rechercher le sous-espace des k premiers axes discriminants = le sous-espace de dimension k engendré par les k premiers vecteurs propres généralisés u_α qui vérifient $\mathbf{E}u_\alpha = \lambda \mathbf{S}u_\alpha$
- $\mathbf{S}^{-1}\mathbf{E}$ n'étant pas symétrique, les axes discriminants ne sont généralement pas orthogonaux

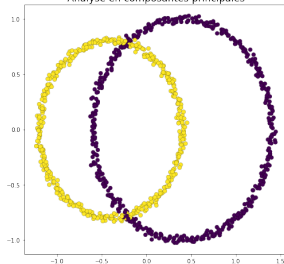


Limites de la réduction linéaire de dimension

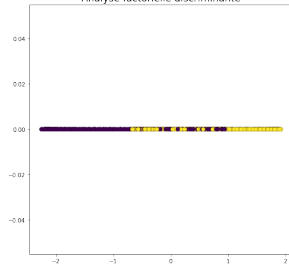
Cercles imbriqués



Analyse en composantes principales



Analyse factorielle discriminante



- Il n'existe pas de transformation linéaire permettant de séparer les deux cercles
- Quels algorithmes pour la réduction de dimension **non-linéaire** ?

Plan du cours

- 1 Motivation
- 2 Réduction linéaire de dimension
- 3 L'algorithme LLE**
- 4 L'algorithme t-SNE
- 5 L'algorithme UMAP
- 6 Compléments : graphes, plongement spectral, complexe simplicial

Cadre formel

Considérons une matrice \mathbf{X} ($n \times D$) de n observations de dimension D

- on cherche une matrice \mathbf{Y} ($n \times d$) qui « représente bien la structure des observations » de \mathbf{X} mais dans un espace de dimension $d < D$
- « représente bien la structure des observations » : les voisins y_j de y_i (lignes de \mathbf{Y}) dans l'espace **réduit** doivent être les mêmes que les voisins x_j de x_i (lignes de \mathbf{X}) dans l'espace **de départ**

Locally Linear Embedding [6]

Principe : localement, le nuage d'observations \mathbf{X} est engendré par un sous-espace affine

1 pour chaque point $\mathbf{x}_i \in \mathbf{X} \subset \mathbb{R}^D$:

- trouver les k plus proches voisins $V_i = \{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \dots, \mathbf{x}_{i_k}\}$ de \mathbf{x}_i

2 résoudre

$$\mathbf{W}^* = \arg \min_{\mathbf{W}} \sum_i \left\| \mathbf{x}_i - \sum_{\mathbf{x}_j \in V_i} w_{ij} \mathbf{x}_j \right\|^2 \quad \text{t.q.} \sum_j w_{ij} = 1$$

c'est à dire, chercher les paramètres \mathbf{W}^* (matrice $n \times k$) t.q. \mathbf{x}_i se reconstruit par la combinaison linéaire de ses voisins V_i

3 résoudre

$$\mathbf{Y}^* = \arg \min_{\mathbf{Y}} \sum_i \left\| \mathbf{y}_i - \sum_{j \text{ t.q. } \mathbf{x}_j \in V_i} w_{ij}^* \mathbf{y}_j \right\|^2 \quad \text{t.q.} \mathbf{y}_i \in \mathbb{R}^d$$

c'est à dire, chercher la matrice réduite \mathbf{Y}^* (matrice $n \times d$) t.q. \mathbf{y}_i se reconstruit par la même combinaison linéaire des voisins correspondants

Illustration

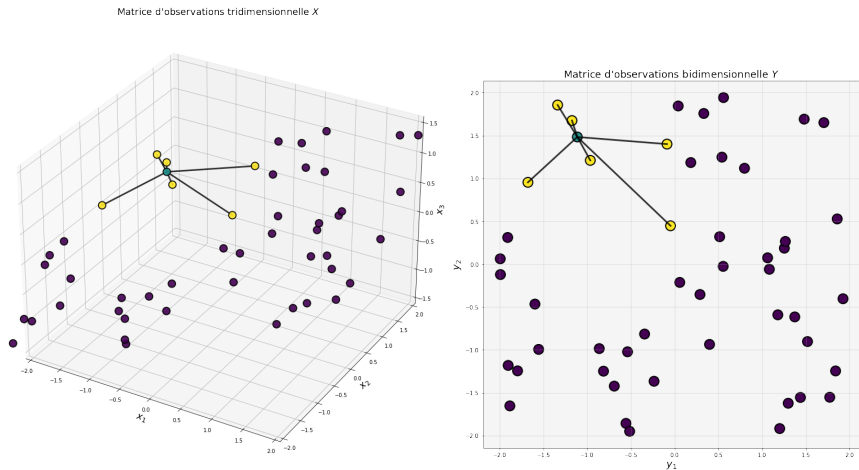
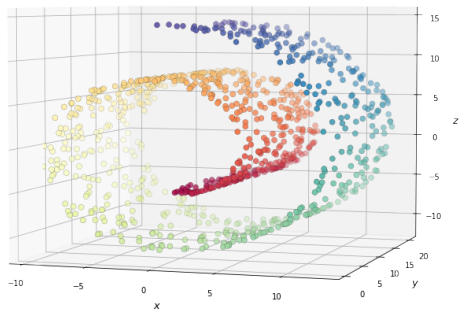


FIG. – LLE trouve les coefficients W tels que le point bleu clair peut se reconstruire par combinaison linéaire de ses 6 plus proches voisins (en jaune). LLE détermine ensuite les coordonnées des points équivalents dans l'espace réduit de sorte que le point réduit correspondant se reconstruit par la même combinaison linéaire de ses voisins.

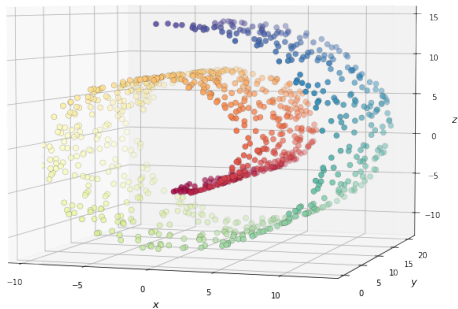
Exemple

Jeu de données « Gâteau roulé »



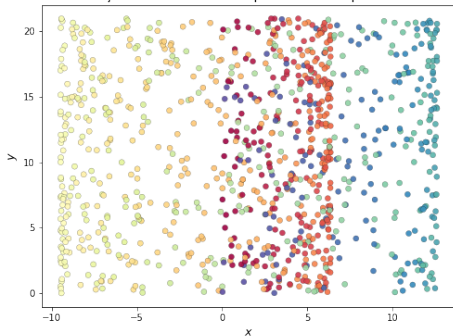
Exemple

Jeu de données « Gâteau roulé »



Projection sur les axes x et y

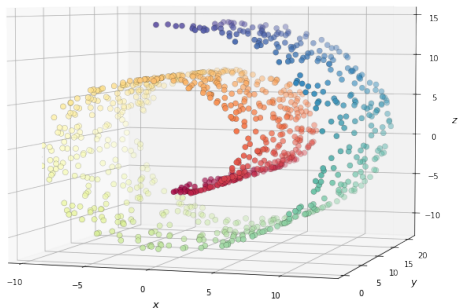
Projection selon les deux premières composantes



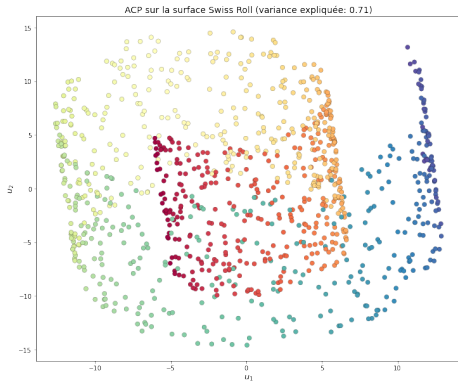
La projection selon les deux premières variables ne rend pas compte de la structure du jeu de données : les variables choisies sont arbitraires

Exemple

Jeu de données « Gâteau roulé »



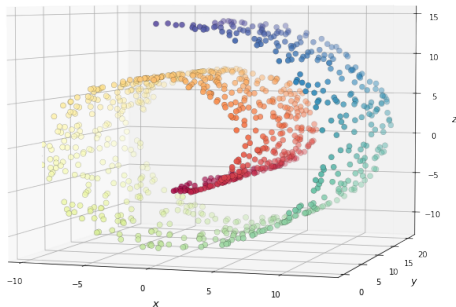
Analyse en composantes principales



L'analyse en composantes principales extrait les directions qui expliquent le plus la variance mais masque la troisième dimension

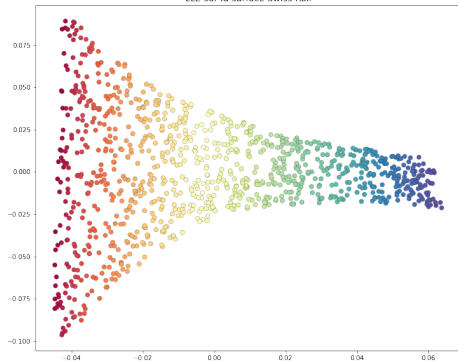
Exemple

Jeu de données « Gâteau roulé »



Locally Linear Embedding

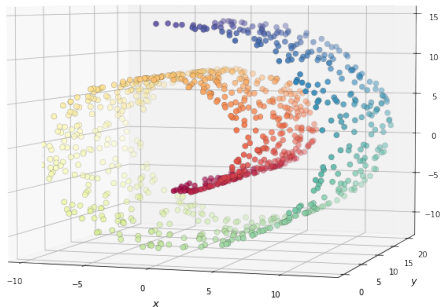
LLE sur la surface Swiss Roll



L'approche par voisinage de LLE permet de mieux respecter la structure locale et de « dérouler » la surface du jeu de données

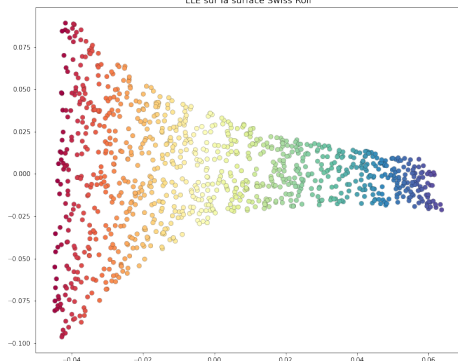
Exemple

Jeu de données « Gâteau roulé »



Locally Linear Embedding

LLE sur la surface Swiss Roll



L'approche par voisinage de LLE permet de mieux respecter la structure locale et de « dérouler » la surface du jeu de données

Voisinages \neq distances

LLE préserve les voisinages, **pas les distances**. Les reconstructions sont **locales** \implies les distances entre des points éloignés n'ont pas de signification !

Plan du cours

- 1 Motivation
- 2 Réduction linéaire de dimension
- 3 L'algorithme LLE
- 4 L'algorithme t-SNE**
- 5 L'algorithme UMAP
- 6 Compléments : graphes, plongement spectral, complexe simplicial

t-SNE : principe général

- *t-distributed Stochastic Neighbor Embedding* [4]
 - Algorithme de réduction de dimension non-linéaire
 - Principe général similaire à LLE : conserver, dans un espace de dimension réduite, les mêmes structures locales au voisinage de chaque point
- Difficultés de LLE :
 - La proximité des k plus proches voisins dépend de la densité locale, l'échelle des similarités conservées n'est pas la même partout
 - La séparation stricte entre voisins, pris en compte totalement, et non voisins, totalement ignorés, introduit des fragilités dans la réduction de dimension

→ Pour t-SNE le voisinage d'un point \mathbf{x}_i est représenté par la probabilité conditionnelle

$p_{j|i} = p(\mathbf{x}_j|\mathbf{x}_i)$ que \mathbf{x}_j soit considéré comme un « voisin » de \mathbf{x}_i

- On cherche dans l'espace réduit les points \mathbf{y}_i tels que $p(\mathbf{y}_j|\mathbf{y}_i) \simeq p(\mathbf{x}_j|\mathbf{x}_i)$
- Les distributions des voisinages sont semblables dans l'espace de départ et dans l'espace réduit

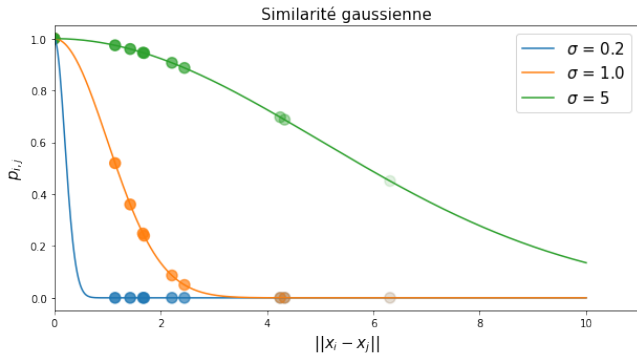
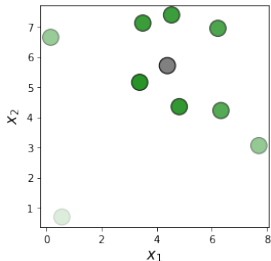
Similarité dans l'espace des observations (ou espace de départ)

- Considérons une matrice \mathbf{X} de n observations \mathbf{x}_i de dimension D
- À chaque paire $\mathbf{x}_i, \mathbf{x}_j$ on associe la probabilité jointe $p_{ij} = p_{ji}$ définie par :

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n} \text{ avec } p_{j|i} = \frac{\exp(-\|\mathbf{x}_j - \mathbf{x}_i\|^2 / 2\sigma^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_k - \mathbf{x}_i\|^2 / 2\sigma^2)}$$

avec par convention $p_{ii} = 0$

- σ choisi en fonction de la perplexité souhaitée (\sim nombre moyen de voisins)



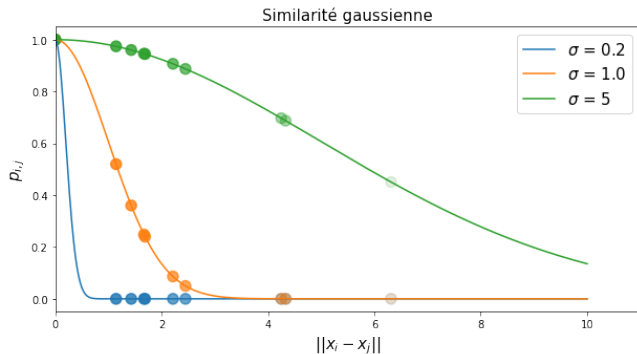
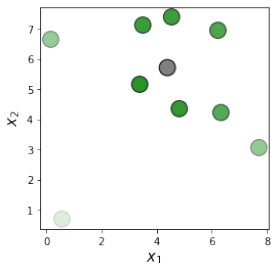
Similarité dans l'espace des observations (ou espace de départ)

- Considérons une matrice \mathbf{X} de n observations \mathbf{x}_i de dimension D
- À chaque paire $\mathbf{x}_i, \mathbf{x}_j$ on associe la probabilité jointe $p_{ij} = p_{ji}$ définie par :

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n} \text{ avec } p_{j|i} = \frac{\exp(-\|\mathbf{x}_j - \mathbf{x}_i\|^2 / 2\sigma^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_k - \mathbf{x}_i\|^2 / 2\sigma^2)}$$

avec par convention $p_{ii} = 0$

- σ choisi en fonction de la perplexité souhaitée (\sim nombre moyen de voisins)



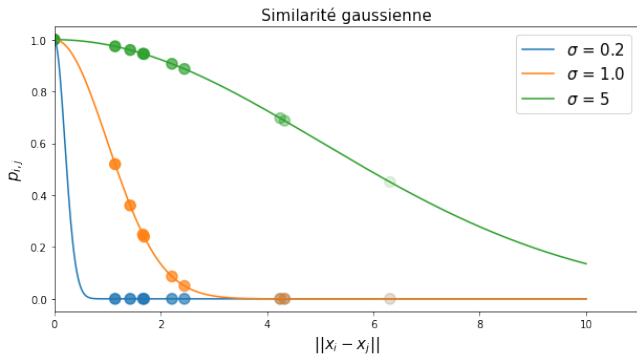
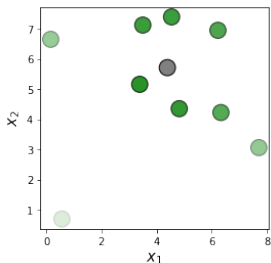
Similarité dans l'espace des observations (ou espace de départ)

- Considérons une matrice \mathbf{X} de n observations \mathbf{x}_i de dimension D
- À chaque paire $\mathbf{x}_i, \mathbf{x}_j$ on associe la probabilité jointe $p_{ij} = p_{ji}$ définie par :

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n} \text{ avec } p_{j|i} = \frac{\exp(-\|\mathbf{x}_j - \mathbf{x}_i\|^2 / 2\sigma^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_k - \mathbf{x}_i\|^2 / 2\sigma^2)}$$

avec par convention $p_{ii} = 0$

- σ choisi en fonction de la **perplexité** souhaitée (\sim nombre moyen de voisins)

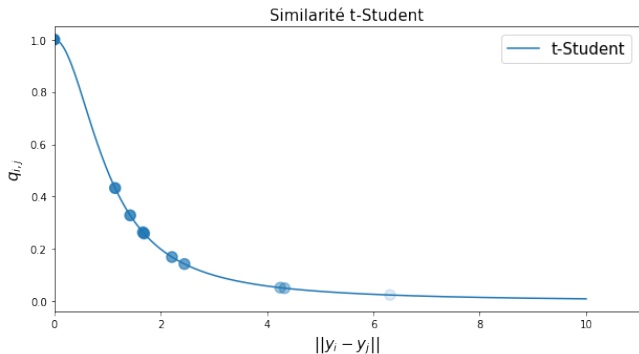
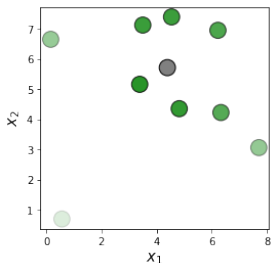


Similarité dans l'espace réduit (ou espace d'arrivée)

- L'objectif de t-SNE est d'obtenir une matrice réduite \mathbf{Y} de dimension $n \times d$, $d < D$, telle que les similarités q_{ij} s'approchent des p_{ij}
- Pour Y , la similarité est définie en utilisant une loi t -Student à 1 degré de liberté (ou loi de Cauchy) :

$$q_{ij} = \frac{(1 + \|y_j - y_i\|_2^2)^{-1}}{\sum_{k \neq i} (1 + \|y_k - y_i\|_2^2)^{-1}}$$

avec par convention, ici encore, $q_{ii} = 0$

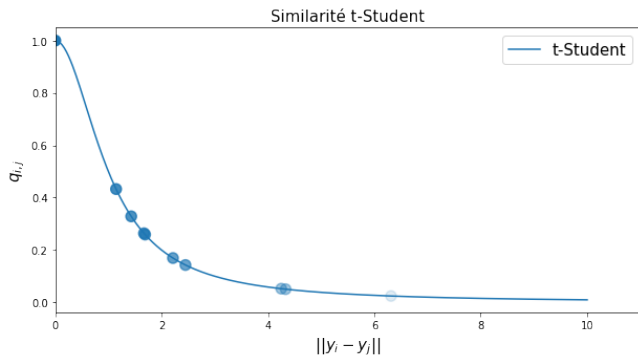
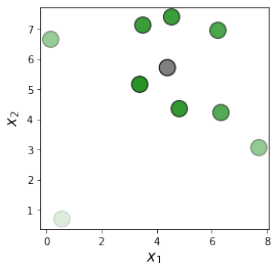


Similarité dans l'espace réduit (ou espace d'arrivée)

- L'objectif de t-SNE est d'obtenir une matrice réduite \mathbf{Y} de dimension $n \times d$, $d < D$, telle que les similarités q_{ij} s'approchent des p_{ij}
- Pour Y , la similarité est définie en utilisant une loi t -Student à 1 degré de liberté (ou loi de Cauchy) :

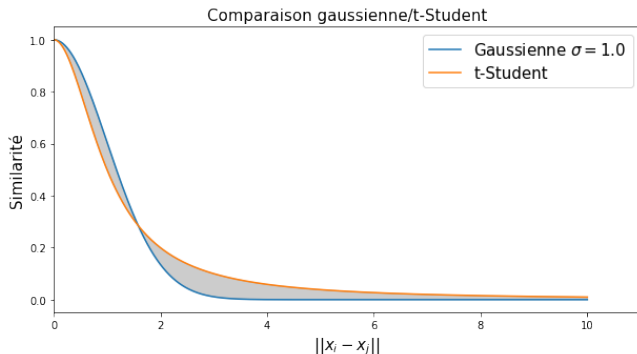
$$q_{ij} = \frac{(1 + \|\mathbf{y}_j - \mathbf{y}_i\|_2^2)^{-1}}{\sum_{k \neq l} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|_2^2)^{-1}}$$

avec par convention, ici encore, $q_{ii} = 0$



Choix des distributions

- Similarité gaussienne : $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$ avec $p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma^2)}{\sum_{k \neq i} \exp(-\|x_k - x_i\|^2 / 2\sigma^2)}$
 - Décroissance rapide : les voisins sont faciles à distinguer des non-voisins
 - Queue courte : les valeurs éloignées sont toutes $\simeq 0 \rightarrow$ tous les non-voisins sont à 0
- Similarité t (Student) : $q_{ij} = \frac{(1 + \|y_i - y_j\|_2^2)^{-1}}{\sum_{k \neq i} (1 + \|y_k - y_i\|_2^2)^{-1}} \rightarrow$ dans l'espace réduit
 - Décroissance rapide : les voisins sont faciles à distinguer des non-voisins
 - Queue longue : meilleure répartition dans l'espace, pas « d'agglutinement »

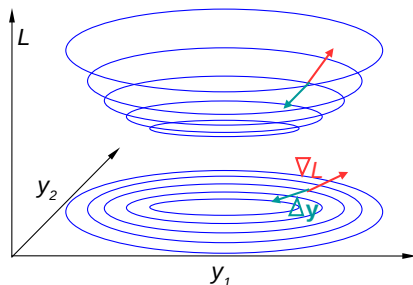


Optimisation

- Une mesure de dissimilarité entre distributions est la *divergence de Kullback-Leibler* :

$$L_{t-SNE} = D_{KL}(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

- Pour minimiser les écarts entre les q_{ij} et les p_{ij} il suffit de minimiser L_{t-SNE}
- Minimisation par **descente de gradient** : appliquer des modifications successives aux paramètres (ici les y_i) du critère optimisé (ici L_{t-SNE}), dans le sens **opposé** au gradient du critère par rapport aux paramètres, jusqu'à l'atteinte d'un minimum



Algorithme de t-SNE

t-SNE modifie les points \mathbf{y}_i pour minimiser L_{t-SNE} en utilisant une descente de gradient :

1 Pour chaque paire de points $(\mathbf{x}_i, \mathbf{x}_j)$:

- Calculer la similarité $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$ avec $p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_k - \mathbf{x}_i\|^2 / 2\sigma^2)}$

2 Placer aléatoirement les points $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n$ dans l'espace de dimension d

3 Répéter tant que $L_{t-SNE} \geq \varepsilon$ et l'algorithme n'a pas convergé et le nombre maximal d'itérations n'est pas atteint :

- Calculer le gradient de la divergence KL par rapport à chaque \mathbf{y}_i :

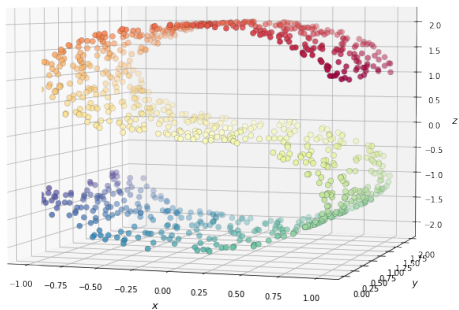
$$\frac{\partial L_{t-SNE}}{\partial \mathbf{y}_i} = 4 \sum_j (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j)(1 + \|\mathbf{y}_i - \mathbf{y}_j\|_2^2)^{-1}$$

- Déplacer chaque point \mathbf{y}_i dans le sens opposé au gradient :

$$\mathbf{y}_i := \mathbf{y}_i - \alpha \frac{\partial L_{t-SNE}}{\partial \mathbf{y}_i}$$

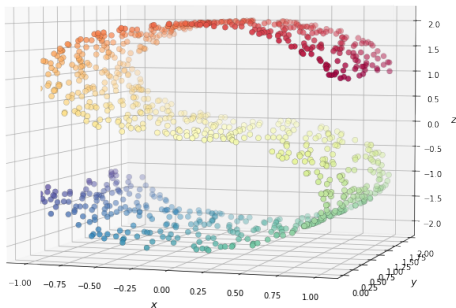
Exemple

Jeu de données « en S »

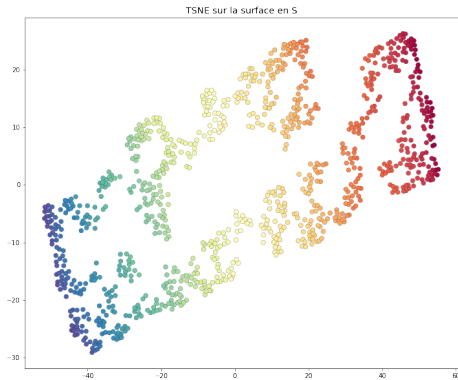


Exemple

Jeu de données « en S »



Projection 2D par t-SNE



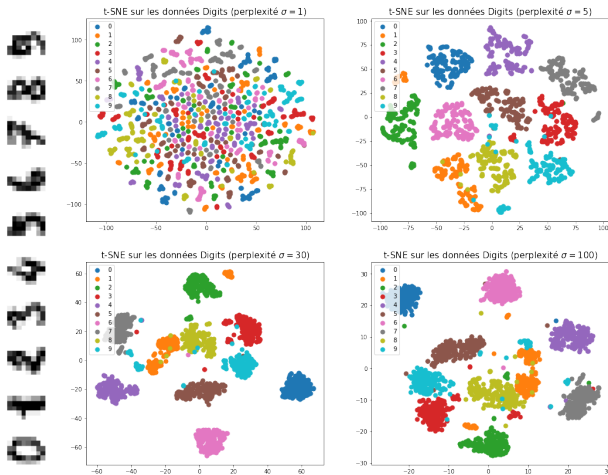
La projection par t-SNE permet de « dérouler » la surface en S.

Initialisation

- L'algorithme classique de t-SNE initialise les points y_i de façon aléatoire
 - Stochasticité élevée
 - Faible préservation de la structure globale
 - Dépendance forte des résultats à l'initialisation

- **Solution** : appliquer une ACP pour le calcul des y_i^0 initiaux
 - Préserve la structure globale ...sauf si la variété est très non-linéaire !
 - Meilleure reproductibilité
 - Plus robuste en pratique

Perplexité



- Perplexité trop faible : structure locale, seuls les voisins immédiats sont pris en compte, les groupes sont découpés en sous-groupes dispersés
- Perplexité trop élevée : structure globale, tous les points sont dans tous les voisinages

Optimisation

- La descente de gradient dans t-SNE fait intervenir deux phases :
 - **Phase 1** : exagération (*early exaggeration*)
 - les probabilités conditionnelles dans l'espace initial sont multipliées par un facteur γ pour augmenter artificiellement la séparation des groupes naturels dans les données (ainsi les écarts entre les valeurs p_{ij} les plus élevées, associées aux voisins les plus proches, et les autres augmentent, les p_{ij} les plus élevées ont donc encore plus d'impact lors des premières itérations)
 - **Phase 2** : optimisation finale
 - valeurs réelles de probabilités conditionnelles pour affiner le placement des observations localement dans les groupes
- Il existe des heuristiques pour choisir le facteur d'exagération et le pas de la descente de gradient (α) [1] :
 - $\alpha = \max(\frac{1}{4} \frac{n}{\text{exagération}}, 50)$
 - ou paramètre `learning_rate='auto'` dans scikit-learn

Approximation

- t-SNE est une méthode coûteuse due à sa complexité élevée
 - Calcul du gradient : $O(dn^2)$
 - Gradient pour un seul \mathbf{y}_i : $O(dn)$

$$\frac{\partial L}{\partial \mathbf{y}_i} = 4 \sum_j (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j)(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$$

- Approximation de Barnes-Hut [3]
 - Calcul du gradient : $O(dn \log n)$
 - Seulement pour la visualisation (espace réduit 2D ou 3D)

Limites

- Sensibilité aux paramètres et stochasticité de la méthode :
 - t-SNE est une méthode stochastique (initialisation aléatoire) : deux répétitions ne donnent pas nécessairement la même projection
 - L'initialisation a un impact significatif sur le résultat final
 - Les paramètres d'optimisation (exagération, pas d'apprentissage et surtout perplexité) peuvent significativement affecter la projection finale
- Interprétabilité sujette à caution [9]
 - La taille des groupes dans t-SNE n'a pas de signification (dépend de la perplexité)
 - La distance entre les groupes n'a pas (toujours) de signification
 - Avec une perplexité faible, le bruit dans les données peut relier artificiellement des groupes ou au contraire les surdiviser
- Inconvénients pratiques
 - Méthode non-inductive : impossible de projeter une nouvelle observation, il faut refaire l'optimisation !
 - Non-inversible : impossible de retrouver l'antécédent dans l'espace des observations d'un point arbitraire de l'espace réduit

Plan du cours

- 1 Motivation
- 2 Réduction linéaire de dimension
- 3 L'algorithme LLE
- 4 L'algorithme t-SNE
- 5 L'algorithme UMAP**
- 6 Compléments : graphes, plongement spectral, complexe simplicial

Principe général de UMAP

- *Uniform Manifold Approximation & Projection* (UMAP) [5]
 - Algorithme de réduction non-linéaire de dimension similaire à t-SNE
 - Paquet `umap-learn` en Python (interface compatible avec `scikit-learn`)
- Idée générale : trouver une représentation des données en plus faible dimension qui a la **même topologie** que le nuage des observations dans l'espace de départ
 - 1 Définir une matrice de similarités S appropriée
 - 2 À partir de S , construire un graphe de similarités
 - 3 Construire une représentation vectorielle des données dans un espace de dimension inférieure qui présente le **même** graphe de similarités
- Différences majeures avec t-SNE
 - Utilise une notion de **perplexité variable** pour définir une similarité **locale** qui dépend de la densité autour de chaque point
 - Considère une famille de fonctions de similarité plus large inspirée de la loi t de Student pour l'espace d'arrivée

Similarité adaptative

- UMAP définit une similarité adaptative qui varie selon la densité locale des données :

$$\rho_{i,j} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2 - \rho_i}{\sigma_i}\right)$$

- Similarité locale suivant densité locale :

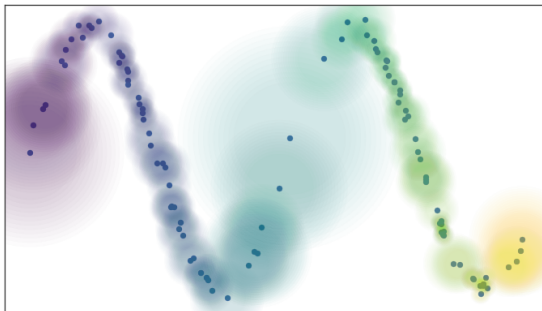


Figure de https://umap-learn.readthedocs.io/en/latest/how_umap_works.html

Similarité adaptative

- UMAP définit une similarité adaptative qui varie selon la densité locale des données :

$$p_{i,j} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2 - \rho_i}{\sigma_i}\right)$$

où ρ_i est la distance entre \mathbf{x}_i et son plus proche voisin

- noyau gaussien adaptatif avec une normalisation locale de la perplexité
 - σ_i est fixé de sorte que :

$$\sum_{j=1}^k \exp\left(-\frac{\max(0, \|\mathbf{x}_i - \mathbf{x}_j\| - \rho_i)}{\sigma_i}\right) = \log_2(k)$$

où k est un paramètre réglant le nombre de plus proches voisins à considérer

- Pour deux points \mathbf{x}_i et \mathbf{x}_j la similarité jointe p_{ij} devrait être symétrique ($p_{ij} = p_{ji}$), obtenu dans UMAP par $p_{ij} = p_{i,j} + p_{j,i} - p_{i,j} \cdot p_{j,i}$
 - la symétrisation est nécessaire car $\rho_i \neq \rho_j$
 - pour rappel, t-SNE utilise une autre symétrisation : $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$

Similarité dans l'espace d'arrivée

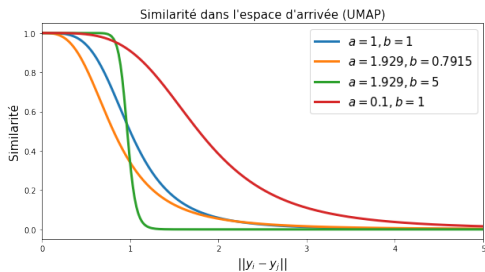
- La similarité dans l'espace d'arrivée s'inspire de la loi t de Student :

$$q_{ij} = \frac{1}{1 + a\|y_i - y_j\|^{2b}}$$

- traîne plus épaisse pour éviter le problème d'agglutinement dans l'espace réduit
- paramètres a et b choisis automatiquement de sorte que la similarité soit la plus proche possible de la fonction suivante définie par morceaux :

$$\begin{cases} 1 & \text{si } \|y_i - y_j\| \leq \text{min_dist} \\ e^{-\|y_i - y_j\|} & \text{si } \|y_i - y_j\| > \text{min_dist} \end{cases}$$

avec min_dist la distance minimale autorisée dans l'espace d'arrivée entre deux points



Optimisation

- Comme pour t-SNE, l'objectif est de faire tendre $q_{ij} \rightarrow p_{ij}$
- La fonction objectif à minimiser est l'**entropie croisée floue** entre les q_{ij} et p_{ij} :

$$\text{CE}(X, Y) = \sum_{i=1}^N \sum_{j=1}^N \left[p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right) + (1 - p_{ij}) \log \left(\frac{1 - p_{ij}}{1 - q_{ij}} \right) \right]$$

- En développant les logarithmes on obtient :

$$\text{CE}(X, Y) = \sum_{i=1}^N \sum_{j=1}^N [p_{ij} \log p_{ij} + (1 - p_{ij}) \log (1 - p_{ij})] \quad (1)$$

$$- \sum_{i=1}^N \sum_{j=1}^N [p_{ij} \log q_{ij} + (1 - p_{ij}) \log (1 - q_{ij})] \quad (2)$$

- premier terme dépend seulement des données de départ, ne peut donc pas être modifié
- seul le second terme est donc minimisé en modifiant les y_i (les projections dans l'espace d'arrivée) et donc les q_{ij}

Optimisation (2)

- Mise à jour des points dans l'espace d'arrivée : $\mathbf{y}_i := \mathbf{y}_i - \alpha \frac{\partial \text{CE}}{\partial \mathbf{y}_i}$
 - α correspond à la vitesse d'apprentissage de l'algorithme de descente de gradient
- UMAP optimise la fonction objectif en réalisant une descente de gradient **stochastique** :
 - Pour accélérer le calcul de CE sur un grand jeu de données, on ne considère qu'un échantillon limité pour en calculer une valeur approchée
 - L'échantillon contient un mélange de voisins ($p_{ij} \simeq 1$) et de non-voisins ($p_{ij} \simeq 0$) de sorte à optimiser simultanément les composantes de l'entropie croisée

Initialisation des points dans l'espace d'arrivée

- Comme pour t-SNE, l'initialisation des points dans l'espace d'arrivée peut avoir un impact significatif sur le résultat de UMAP
- Dans le cas de t-SNE avec *early exaggeration*, il a été observé que la première phase de l'algorithme revient approximativement à réaliser un plongement spectral [2]
 - L'initialisation par défaut de UMAP consiste donc à réaliser un **plongement spectral**
 - Moins d'aléatoire au moment de l'initialisation
 - Convergence plus rapide de la descente de gradient stochastique

Algorithme

Données : nuage de points $\mathcal{E} = \{\mathbf{x}_i\}_{1 \leq i \leq N}$, nombre de plus proches voisins k , dimension de l'espace d'arrivée d , paramètres (a, b) de la similarité q_{ij}

Sortie : nuage de points réduit $\mathcal{F} = \{\mathbf{y}_i\}_{1 \leq i \leq N}$

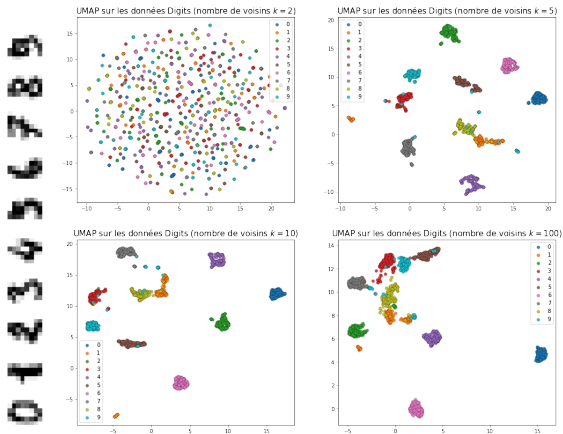
- 1 Déterminer pour chaque point \mathbf{x}_i ses k plus proches voisins
- 2 Calculer p_{ij} pour toutes les paires (i, j)
- 3 Initialiser les points $\mathcal{F} = \{\mathbf{y}_i\}$ à l'aide d'un plongement spectral dans \mathbb{R}^d
- 4 Répéter tant que l'entropie croisée $CE \geq \varepsilon$ et l'algorithme n'a pas convergé :
 - Tirer aléatoirement un échantillon uniforme de m observations \mathbf{x}_i
 - Calculer q_{ij} pour les points de l'échantillon des \mathbf{y}_i correspondants
 - Calculer $CE(p, q)$ sur cet échantillon
 - Déplacer chaque point \mathbf{y}_i par :

$$\mathbf{y}_i := \mathbf{y}_i - \alpha \frac{\partial CE}{\partial \mathbf{y}_i}$$

Paramétrage

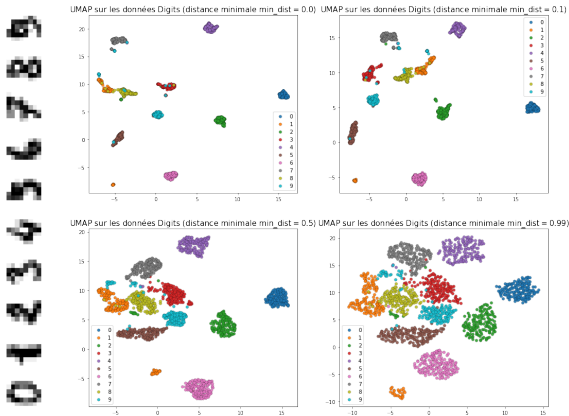
■ Principaux paramètres de l'algorithme UMAP

- 'n_components' : la dimension de l'espace (euclidien) d'arrivée
- 'n_neighbours' : le nombre de voisins à considérer pour définir la similarité adaptative
- 'min_dist' : la distance minimale autorisée entre deux points dans l'espace d'arrivée



Paramétrage

- Principaux paramètres de l'algorithme UMAP
 - 'n_components' : la dimension de l'espace (euclidien) d'arrivée
 - 'n_neighbours' : le nombre de voisins à considérer pour définir la similarité adaptative
 - 'min_dist' : la distance minimale autorisée entre deux points dans l'espace d'arrivée



UMAP (semi-)supervisé

- Considérons un jeu de données contenant des **étiquettes de groupes** :
 $\mathcal{X} = \{(\mathbf{x}_1, l_1), \dots, (\mathbf{x}_n, l_n)\}$, \mathbf{x}_i étant les obs. et l_i les étiquettes (variable nominale)
 - Comment inclure la connaissance des étiquettes dans l'algorithme UMAP ?
⇒ définir une distance mixte sur le couple (\mathbf{x}_i, l_i)

- La distance sur les étiquettes des groupes est définie comme étant :

$$d^{\text{classe}}(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 0 & \text{si } l_i = l_j \\ 0.5 & \text{si } l_i \text{ ou } l_j \text{ est inconnu} \\ 1 & \text{si } l_i \neq l_j \end{cases}$$

- La distance totale impliquée dans le calcul des similarités entre points dans l'espace de départ est une somme pondérée par un paramètre λ (`target_weight`) :

$$d(\mathbf{x}_i, \mathbf{x}_j) = (1 - \lambda)d^{\text{obs}}(\mathbf{x}_i, \mathbf{x}_j) + \lambda d^{\text{classe}}(\mathbf{x}_i, \mathbf{x}_j)$$

- $\lambda = 0$ → seule la distance entre les données \mathbf{x}_i est prise en compte
- $\lambda = 0.5$ → la distance totale est une moyenne de la distance entre les observations et entre les étiquettes
- $\lambda = 1$ → seule la distance entre les étiquettes est prise en compte

Limites

- Projection paramétrique de nouvelles données
 - Comme t-SNE, UMAP est une méthode non inductive, il n'y a pas d'expression explicite de la projection d'un point de l'espace de départ vers l'espace d'arrivée
 - Possible d'approcher cette projection à l'aide d'un modèle de régression, par ex. un réseau de neurones (perceptron multi-couche)
→ ParametricUMAP dans le paquet `umap-learn`
- Interprétabilité sujette à caution
 - Mêmes précautions que pour t-SNE concernant l'absence de signification des densités des groupes ou des distances inter-groupes
- Réglage des paramètres de l'algorithme
 - Le nombre de voisins `n_neighbors` de UMAP est moins sensible que la perplexité de t-SNE aux données isolées
 - Important de comparer plusieurs visualisations avec des paramètres différents pour observer les structures locales et la structure globale

Validation des méthodes de réduction de dimension

- Comment vérifier la pertinence d'une projection obtenue ?
 - Pas de critère simple d'interprétation comme pour les méthodes factorielles
- Solution 1 : performances des algorithmes de k plus proches voisins
 - A minima, les algorithmes de réduction non-linéaire de dimension doivent conserver les **relations de voisinage**
 - Calculer la précision d'un algorithme de kNN en classification sur $\mathcal{E} = \{\mathbf{x}_i, 1 \leq i \leq N\}$ dans l'espace de départ, puis la précision d'un kNN en classification sur $\mathcal{F} = \{\mathbf{y}_i, 1 \leq i \leq N\}$ dans l'espace d'arrivée
 - Les performances dans l'espace réduit devraient être équivalentes ou supérieures à celles dans l'espace de départ
 - Inconvénient : nécessite d'avoir des étiquettes de groupes sur les observations
- Solution 2 : critères spécifiques [7]
 - La réduction de dimension doit conserver l'ordonnancement des voisins :
 - *Trustworthiness* ($\in [0, 1]$, meilleure si plus proche de 1) :

$$T(k) = 1 - \frac{2}{nk(2n - 3k - 1)} \sum_{i=1}^n \sum_{j \in U_i^{(k)}} \max(0, (r(i, j) - k))$$

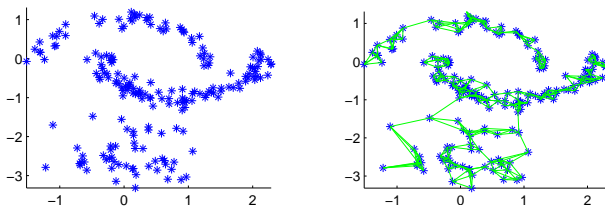
avec $U_i^{(k)}$ les indices des k plus proches voisins de i dans l'espace réduit et $r(i, j)$ le rang de j parmi les voisins de i dans l'espace de départ

Plan du cours

- 1 Motivation
- 2 Réduction linéaire de dimension
- 3 L'algorithme LLE
- 4 L'algorithme t-SNE
- 5 L'algorithme UMAP
- 6 Compléments : graphes, plongement spectral, complexe simplicial

Principe du plongement spectral

- Ensemble $\mathcal{E} = \{x_i, 1 \leq i \leq N\}$ de N données de dimension D décrites par une matrice de **similarités** \mathbf{S} telle que l'élément $s_{ij} \geq 0$ soit la similarité entre x_i et x_j
- Objectif : répartir des données $\mathcal{F} = \{y_i, 1 \leq i \leq N\}$ dans un espace de dimension $d < D$ tel que les distances $\|y_i - y_j\|$ soient faibles si les similarités s_{ij} sont fortes et inversement
- Approche (voir aussi **ces explications sur la classification spectrale**) :
 - 1 À partir de \mathbf{S} construire un graphe de similarités
 - 2 À l'aide du graphe, construire une représentation vectorielle « convenable » des données



(figure issue de [8])

Graphes : quelques définitions

- Graphe non orienté $G = (V, E)$, $V = \{v_1, \dots, v_N\}$ étant l'ensemble des N sommets et E l'ensemble des arêtes
- Arêtes pondérées : soit $w_{ij} \geq 0$ le poids de l'arête qui lie les sommets v_i et v_j
 - $w_{ij} = 0 \Leftrightarrow$ absence d'arête entre v_i et v_j
 - Matrice des poids : \mathbf{W} d'élément générique w_{ij}
- Degré du sommet v_i : $d_i = \sum_{j=1}^N w_{ij}$
 - Matrice diagonale des degrés : \mathbf{D} avec $\{d_1, \dots, d_N\}$ sur la diagonale
- Chaîne :
 - Un chemin (une chaîne) $\mu(v_i, v_j)$ entre les sommets v_i et v_j est une suite d'arêtes de E permettant de relier les deux sommets
- Vecteur indicateur d'un sous-ensemble $C \subset V$: vecteur \mathbf{c} à N composantes binaires tel que $c_i = 1$ si le sommet $v_i \in C$ et $c_i = 0$ sinon
- Composante connexe :
 - $C \subset V$ forme une composante connexe si $\forall v_i, v_j \in C$ il existe une chaîne $\mu(v_i, v_j)$ les reliant et $\forall v_i \in C, \forall \mu(v_i, v_p), v_p \in C$

Matrice laplacienne d'un graphe

- Pour un graphe non orienté à arêtes pondérées, la matrice **laplacienne** (non normalisée) est $\mathbf{L} = \mathbf{D} - \mathbf{W}$
- Propriétés :
 - $\forall \mathbf{x} \in \mathbb{R}^N, \mathbf{x}^T \mathbf{L} \mathbf{x} = \frac{1}{2} \sum_{i,j=1}^N w_{ij} (x_i - x_j)^2$
 - \mathbf{L} est symétrique et semi-définie positive $\Rightarrow \mathbf{L}$ a N valeurs propres réelles $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_N$ (valeurs propres λ et vecteurs propres $\mathbf{u} : \mathbf{L} \mathbf{u} = \lambda \mathbf{u}$)
 - La valeur propre la plus faible de \mathbf{L} est 0, le vecteur propre correspondant étant le vecteur constant $\mathbf{1}$; justification : $d_i = \sum_{j=1}^N w_{ij}$, donc $d_i + \sum_{j=1}^N (-w_{ij}) = 0$
- \mathbf{L} et les composantes connexes de G :
 - La multiplicité de la valeur propre 0 de \mathbf{L} est égale au nombre de composantes connexes de G
 - Les vecteurs indicateurs de ces composantes connexes sont des vecteurs propres de \mathbf{L} pour la valeur propre 0
- Matrices laplaciennes normalisées :
 - $\mathbf{L}_{rw} = \mathbf{D}^{-1} \mathbf{L} (= \mathbf{I}_N - \mathbf{D}^{-1} \mathbf{W})$
 - $\mathbf{L}_{sym} = \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{-1/2} (= \mathbf{I}_N - \mathbf{D}^{-1/2} \mathbf{W} \mathbf{D}^{-1/2})$

Plongement spectral : un algorithme

Données : matrice de similarités \mathbf{S} , $N \times N$

Résultat : observations réduites $\mathcal{F} = \{\mathbf{y}_i, 1 \leq i \leq N\}$

- 1 Construire le graphe de similarités G à partir de \mathbf{S} ($w_{ij} = s_{ij}$)
- 2 Calculer la matrice laplacienne \mathbf{L}
- 3 Calculer les k vecteurs propres $\mathbf{u}_1, \dots, \mathbf{u}_k$ associés aux k plus petites valeurs propres de $\mathbf{L}\mathbf{u} = \lambda\mathbf{D}\mathbf{u}$
- 4 Soit \mathbf{U}_k la matrice $N \times k$ dont les colonnes sont les vecteurs $\mathbf{u}_1, \dots, \mathbf{u}_k$
- 5 $\mathbf{y}_i \in \mathbb{R}^k, 1 \leq i \leq N$ sont les N lignes de la matrice \mathbf{U}_k

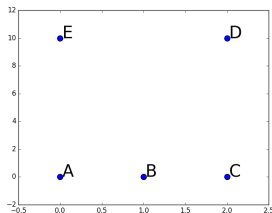
Justification issue de la coupe d'un graphe

- Graphe non orienté à arêtes pondérées $G = (V, E) \rightarrow$ le découper en k sous-ensembles disjoints V_1, \dots, V_k de sommets
- Coupe minimum : minimiser $\sum_{s=1}^k \omega(V_s, V - V_s)$, où $\omega(V_s, V_t) = \sum_{v_i \in V_s, v_j \in V_t} w_{ij}$
 \rightarrow souvent, la solution sépare le graphe en quelques sommets plus un grand sous-graphe...
- Coupe normalisée (*normalized cut*, $Ncut$) : minimiser $\sum_{s=1}^k \frac{\omega(V_s, V - V_s)}{\text{vol}(V_s)}$, où $\text{vol}(V_s) = \sum_{v_i \in V_s} d_i$
 \rightarrow permet d'obtenir des sous-ensembles plus équilibrés
 \rightarrow mais problème NP-difficile !
 - Reformuler le problème en utilisant un vecteur indicateur pour chaque sous-ensemble et la « relaxation » continue du problème discret \Rightarrow
 - 1 Minimisation sous contraintes d'égalité
 - 2 Recherche des plus petites valeurs propres de $\mathbf{L}\mathbf{U} = \lambda\mathbf{D}\mathbf{U}$ et des vecteurs propres associés (\rightarrow colonnes de \mathbf{U}_k)
 - 3 Retour au discret en appliquant K -means aux lignes de \mathbf{U}_k

Exemple simple : données bidimensionnelles

■ Exemple

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 2 & 0 \\ 2 & 10 \\ 0 & 10 \end{bmatrix}$$



- Matrice des similarités obtenues à partir du noyau gaussien de variance 1 appliqué à la distance euclidienne (valeurs arrondies à 6 décimales)

$$\begin{bmatrix} 1.000000 & 0.367879 & 0.018316 & 0.000000 & 0.000000 \\ 0.367879 & 1.000000 & 0.367879 & 0.000000 & 0.000000 \\ 0.018316 & 0.367879 & 1.000000 & 0.000000 & 0.000000 \\ 0.000000 & 0.000000 & 0.000000 & 1.000000 & 0.018316 \\ 0.000000 & 0.000000 & 0.000000 & 0.018316 & 1.000000 \end{bmatrix}$$

Exemple simple : matrice laplacienne

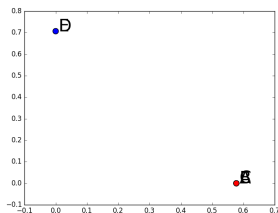
- Matrice laplacienne normalisée L_{rw} (valeurs arrondies à 6 décimales)

$$\begin{bmatrix} 1.000000 & -0.952574 & -0.047426 & 0.000000 & 0.000000 \\ -0.500000 & 1.000000 & -0.500000 & 0.000000 & 0.000000 \\ -0.047426 & -0.952574 & 1.000000 & 0.000000 & 0.000000 \\ 0.000000 & 0.000000 & 0.000000 & 1.000000 & -1.000000 \\ 0.000000 & 0.000000 & 0.000000 & -1.000000 & 1.000000 \end{bmatrix}$$

- 2 composantes connexes, valeur propre 0.0 de L_{rw} de multiplicité 2
 - Valeurs propres de L_{rw} en ordre croissant : **0.0**, **0.0**, 1.0474, 1.9525, 2.0000

- Matrice U :

$$\begin{bmatrix} 0.577350 & 0.000000 \\ 0.577350 & 0.000000 \\ 0.577350 & 0.000000 \\ 0.000000 & 0.707107 \\ 0.000000 & 0.707107 \end{bmatrix}$$



- Réduction 2D \searrow 1D : représentation des observations par une des colonnes de U

Éléments de topologie : le simplexe

- Soit $\mathcal{E} \in \mathbb{R}^k$ un ensemble de points dans un espace vectoriel de dimension k
- Un **simplexe** de dimension k est défini comme étant l'enveloppe convexe de $k + 1$ points $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{k+1}\} \subset \mathcal{E}$
 - Le simplexe est un objet multi-dimensionnel à $k + 1$ sommets et $k + 1$ côtés (ou faces)
- Exemples :
 - $k = 0$: un point
 - $k = 1$: un segment (deux points)
 - $k = 2$: un triangle (trois points)
 - $k = 3$: un tétraèdre (quatre points)etc.



0-simplex



1-simplex



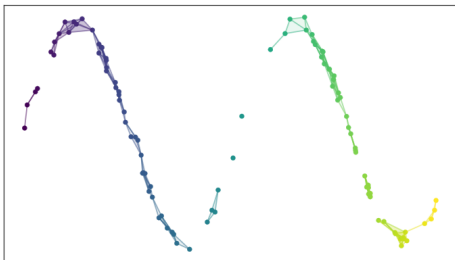
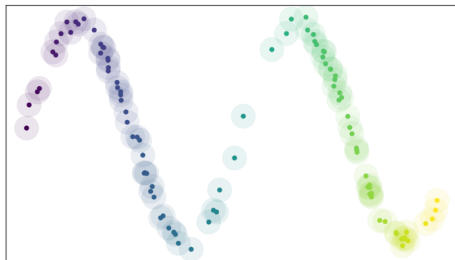
2-simplex



3-simplex

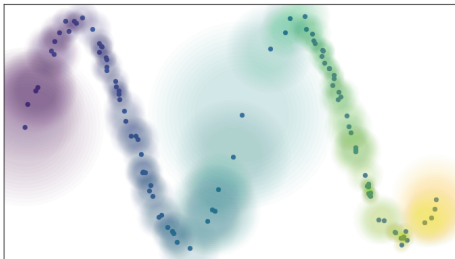
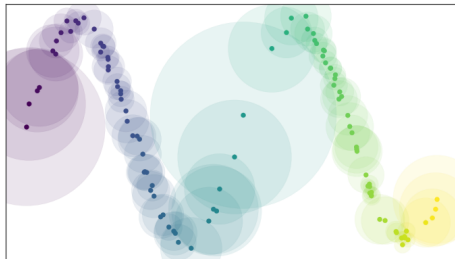
Éléments de topologie : le complexe simplicial

- Un **complexe simplicial** \mathcal{K} de \mathcal{E} est défini comme un ensemble de simplexes reliés par au moins une face, c'est-à-dire :
 - $\mathcal{K} = \{S_1, S_2, \dots, S_m\}$
 - $\forall 1 \leq i \leq m, S_i$ est un k -simplexe de \mathcal{E}
 - $\forall 1 \leq i \leq m, \exists 1 \leq j \leq m, j \neq i$ tel que S_i et S_j partagent une face en commun.
- Pour estimer la topologie d'un nuage de points $\mathcal{E} = \{x_i, 1 \leq i \leq n\} \subset \mathbb{R}^k$, il est intéressant d'examiner le complexe simplicial qui « couvre » tous les points :
 - Chaque point x_i est relié aux voisins qui sont dans une boule ouverte de rayon r
 - Si deux boules ouvertes s'intersectent (si x_i et x_j sont voisins), on place un 1-simplexe reliant x_i et x_j
 - Si k boules ouvertes s'intersectent (si $x_{i_1}, x_{i_2}, \dots, x_{i_k}$ sont voisins), on place un k -simplexe reliant tous ces points



Estimation de la topologie d'un nuage de points

- Le choix du rayon r de la boule ouverte est difficile :
 - boules nettes (ou strictes) \Rightarrow fragilité par rapport au choix du rayon
 - r fixe \Rightarrow boules uniformes, or données non uniformément réparties
 - r variable \rightarrow comment savoir quelle valeur prendre pour un x_i donné ?
- Solution partielle : complexe simplicial « flou » (boules floues) \leftarrow graphe pondéré
 - Chaque arête dispose d'un poids $w_{ij} \leftarrow$ peut s'obtenir à partir de la similarité
 - Mais les similarités peuvent-elles être uniformes sur les données ?
 - t-SNE : **oui**, la perplexité (valeur unique) définit la forme de la fonction de similarité
 - UMAP : **non**, métrique locale normalisée par la distance au $k^{\text{ième}}$ voisin \rightarrow boules de taille variable



Références I



A. C. Belkina, C. O. Ciccolella, R. Anno, R. Halpert, J. Spidlen, and J. E. Snyder-Cappione.

Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets.

Nature Communications, 10(1) :5415, Nov. 2019.



G. C. Linderman and S. Steinerberger.

Clustering with t-sne, provably.

SIAM Journal on Mathematics of Data Science, 1(2) :313–332, Jan 2019.



L. v. d. Maaten.

Accelerating t-SNE using Tree-Based Algorithms.

Journal of Machine Learning Research, 15(93) :3221–3245, 2014.



L. v. d. Maaten and G. Hinton.

Visualizing Data using t-SNE.

Journal of Machine Learning Research, 9(86) :2579–2605, 2008.



L. McInnes, J. Healy, N. Saul, and L. Großberger.

Umap : Uniform manifold approximation and projection.

Journal of Open Source Software, 3(29) :861, 2018.

Références II



S. T. Roweis and L. K. Saul.

Nonlinear Dimensionality Reduction by Locally Linear Embedding.

Science, 290(5500) :2323–2326, Dec. 2000.



S. Stasis, R. Stables, and J. Hockman.

Semantically controlled adaptive equalisation in reduced dimensionality parameter space.

Applied Sciences, 6(44) :116, Apr 2016.



U. von Luxburg.

A tutorial on spectral clustering.

CoRR, abs/0711.0189, 2007.



M. Wattenberg, F. Viégas, and I. Johnson.

How to Use t-SNE Effectively.

Distill, 1(10), Oct. 2016.