Géométrie des Données et Apprentissage Machine

Youssef MESRI - MINES Paris - PSL

November 19, 2025

Contents

Chapter 1

Géométrie des Données et Apprentissage sur Variétés

1.1 Introduction

La géométrie des données est une discipline fondamentale pour comprendre et exploiter la structure cachée des données modernes, souvent de très haute dimension. Ce livre propose une introduction détaillée aux concepts clés, illustrés par des exemples, des démonstrations et des exercices corrigés.

Les données modernes, issues de domaines variés (images, sons, textes, signaux), sont généralement représentées dans des espaces vectoriels de grande dimension ($\mathbb{R}^d$ avec $d \gg 1$). Pourtant, ces données possèdent souvent une structure intrinsèque de faible dimension, appelée **variété**.

Exemple : Les images de chiffres manuscrits (base MNIST) sont des vecteurs de dimension 784, mais l'ensemble des images possibles se répartit sur une variété de dimension beaucoup plus faible (environ 10).

Ce chapitre est organisé en plusieurs sections, chacune abordant un aspect fondamental de la géométrie des données et de l'apprentissage sur variétés :

- Malédiction de la dimension et concentration des distances
- Variétés et distances géodésiques
- Réduction de dimension : PCA, Isomap, Diffusion Maps, t-SNE
- Applications pratiques et exercices

1.2 Malédiction de la dimension et concentration des distances

1.2.1 Explosion dimensionnelle

Lorsque la dimension d augmente, le volume de l'espace croît exponentiellement. Dans un cube unité $[0,1]^d$, la majeure partie du volume se concentre près des bords.

Les distances entre points deviennent moins discriminantes :

$$d_{\min} = \min_{i \neq j} \|x_i - x_j\|,$$
$$d_{\max} = \max_{i \neq j} \|x_i - x_j\|.$$

On observe que :

$$\lim_{d \to \infty} \frac{d_{\max} - d_{\min}}{d_{\min}} \to 0.$$

Conséquence : toutes les distances deviennent presque égales !

1.2.2 Concentration des distances

La distance au carré entre deux points aléatoires $x, y \in [0,1]^d$ s'écrit :

$$S_d = \|x - y\|_2^2 = \sum_{i=1}^{d} (x_i - y_i)^2.$$

L'espérance et la variance de S_d croissent avec d, mais la dispersion relative décroît comme $1/\sqrt{d}$.

Interprétation : En grande dimension, toutes les distances se ressemblent (concentration autour de la moyenne). Les méthodes classiques basées sur la distance (kNN, clustering) deviennent inefficaces.

1.2.3 Démonstration mathématique

Soient $U, V \sim \mathcal{U}[0,1]$ indépendants. Posons $X = (U - V)^2$.

Espérance : $\mathbb{E}[X] = \frac{1}{6}$

Moment d'ordre 4 : $\mathbb{E}[(U - V)^4] = \frac{1}{15}$

Variance : $\mathrm{Var}(X) = \frac{7}{180}$

Pour d dimensions, $S_d = \sum_{i=1}^{d} X_i$ avec X_i i.i.d.

Espérance : $\mathbb{E}[S_d] = d\mu$

Variance : $\mathrm{Var}(S_d) = d\sigma^2$

Coefficient de variation : $\frac{\sqrt{d\sigma^2}}{d\mu} = \frac{C}{\sqrt{d}}$

1.2.4 Exercice

Montrer que la différence de deux variables uniformes sur $[0,1]$ **suit une loi triangulaire sur** $[-1,1]$**.**

Corrigé : La densité de $Z = U - V$ est donnée par :

$$f_Z(z) = \begin{cases} 1 + z, & -1 \leq z < 0, \\ 1 - z, & 0 \leq z \leq 1, \\ 0, & \text{sinon.} \end{cases}$$

1.2.5 Conséquences pratiques

Les méthodes classiques basées sur des distances euclidiennes deviennent inefficaces en grande dimension. Il est nécessaire d'utiliser des méthodes exploitant la structure intrinsèque des données (géométrie des données).

1.3 Variétés et distances géodésiques

1.3.1 Qu'est-ce qu'une variété ?

Une **variété** est un espace qui, localement, ressemble à un espace euclidien de dimension plus faible. Les données réelles, bien que plongées dans un espace de grande dimension, sont souvent contraintes par des relations internes qui les font se répartir sur une variété de dimension intrinsèque $k \ll d$.

Exemple : Les images de chiffres manuscrits, bien que décrites par 784 pixels, sont générées par un nombre limité de facteurs (forme, épaisseur, inclinaison, etc.), ce qui réduit la dimension effective de l'ensemble des images possibles.

1.3.2 Distance euclidienne vs distance géodésique

La **distance euclidienne** mesure le plus court chemin dans l'espace ambiant $\mathbb{R}^d$. Sur une variété courbe, cette distance peut être trompeuse : deux points proches sur la variété peuvent être éloignés euclidiennement si la variété est repliée.

La **distance géodésique** est la longueur du plus court chemin sur la variété elle-même. Elle capture la vraie proximité intrinsèque des données.

Illustration : Sur un "Swiss Roll" (rouleau suisse), la distance euclidienne entre deux points sur des tours différents est grande, alors que la distance géodésique (le long du rouleau) est plus courte.

1.3.3 Conséquences pour l'apprentissage

Les méthodes d'apprentissage doivent tenir compte de la structure de la variété pour être efficaces. Les algorithmes de réduction de dimension (Isomap, Diffusion Maps, t-SNE) cherchent à préserver les distances géodésiques ou la structure locale de la variété.

1.3.4 Exercice

Exercice : Sur un Swiss Roll généré en 3D, comparez la distance euclidienne et la distance géodésique entre deux points situés sur des tours différents.

Corrigé : La distance euclidienne ignore la structure du rouleau et peut relier deux points à travers l'espace, alors que la distance géodésique suit la surface du rouleau, respectant la structure intrinsèque des données.

1.4 Réduction de dimension : PCA, Isomap, Diffusion Maps, t-SNE

1.4.1 Pourquoi réduire la dimension ?

La réduction de dimension vise à représenter des données de grande dimension dans un espace de plus faible dimension, tout en préservant leur structure essentielle. Cela facilite la visualisation, l'analyse et l'apprentissage automatique.

1.4.2 Analyse en Composantes Principales (PCA)

Définition et principe

L'Analyse en Composantes Principales (PCA) est une méthode linéaire de réduction de dimension qui consiste à projeter les données sur les axes de plus grande variance. Elle permet de simplifier la structure des données tout en conservant l'information essentielle.

Objectif : Trouver les directions principales (axes) qui capturent le plus de variance dans les données.

Algorithme de la PCA

1. Centrer les données : soustraire la moyenne de chaque variable.
2. Calculer la matrice de covariance Σ :

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T$$

3. Calculer les valeurs propres et vecteurs propres de Σ.
4. Sélectionner les k plus grandes valeurs propres et leurs vecteurs associés.
5. Projeter les données sur les k axes principaux.

Remarque : La décomposition en valeurs singulières (SVD) de la matrice des données X est une alternative efficace pour calculer la PCA.

Exemple numérique

- Données simulées $X \in \mathbb{R}^{100\times 5}$.
- Calcul de la matrice de covariance, valeurs propres et projection sur les deux premières composantes principales.
- Visualisation : les données se concentrent le long des directions principales.

Applications

- Visualisation de données haute dimension (ex : MNIST en 2D ou 3D).
- Prétraitement pour l'apprentissage automatique.
- Compression et débruitage.

Exercices

- Appliquer la PCA sur un jeu de données simulé et visualiser les deux premières composantes.
- Calculer la variance expliquée par chaque composante et choisir le nombre optimal de dimensions.
- Utiliser la SVD pour réaliser la PCA sur des données réelles.

1.4.3 Isomap

Définition et principe

Isomap (Isometric Mapping) est une méthode non linéaire de réduction de dimension qui vise à préserver les distances géodésiques entre les points d'une variété plongée dans un espace de haute dimension. Elle étend la méthode MDS (Multidimensional Scaling) aux variétés non linéaires.

Objectif : Reconstituer la structure intrinsèque des données en utilisant les distances le long de la variété (géodésiques) plutôt que les distances euclidiennes.

Algorithme Isomap

1. Construire le graphe de voisinage (k-NN ou ϵ-voisinage).
2. Calculer les plus courts chemins entre tous les points (algorithme de Dijkstra ou Floyd-Warshall) pour approximer les distances géodésiques.
3. Appliquer la MDS (décomposition spectrale de la matrice de distances) pour obtenir une représentation en dimension réduite.

Exemple : Swiss Roll

- Générer un jeu de données 3D enroulé (Swiss Roll).
- PCA ne parvient pas à "dérouler" la structure.
- Isomap reconstitue correctement la structure 2D sous-jacente.

Applications

- Visualisation de variétés non linéaires.
- Analyse de la structure intrinsèque des données.

Exercices

- Implémenter Isomap sur un Swiss Roll et visualiser la représentation 2D obtenue.
- Comparer les résultats d'Isomap et de PCA sur des données simulées.
- Étudier l'influence du paramètre k sur la qualité de la reconstruction.

1.4.4 Diffusion Maps

Définition et principe

Diffusion Maps est une méthode non linéaire de réduction de dimension qui exploite la notion de diffusion sur un graphe de voisinage pour révéler la structure globale et multi-échelle des données. Elle repose sur l'analyse spectrale du laplacien du graphe construit à partir des données.

Objectif : Capturer la connectivité et la structure intrinsèque des données, même en présence de bruit ou de distributions complexes.

Algorithme Diffusion Maps

1. Construire le graphe de voisinage (pondéré par une fonction de similarité, ex : $W_{ij} = \exp(-\|x_i - x_j\|^2/\varepsilon)$).
2. Calculer la matrice de transition $P = D^{-1}W$, où D est la matrice de degré.
3. Calculer les valeurs propres $\lambda_1, \lambda_2, ...$ et vecteurs propres ψ_i de P.
4. Représenter chaque point x par le vecteur $(\lambda_1^t \psi_1(x), ..., \lambda_k^t \psi_k(x))$ pour une certaine échelle t.

Exemple : Séparation de groupes

- Appliquer Diffusion Maps sur des données connectées par des chemins complexes.
- Permet de séparer des groupes de données difficilement distinguables par des méthodes classiques.

Applications

- Analyse de la connectivité globale des données.
- Visualisation et clustering robuste au bruit.
- Étude de phénomènes dynamiques (ex : diffusion sur réseaux).

1.4.5 Exercices

- Implémenter Diffusion Maps sur un Swiss Roll bruité et comparer avec Isomap.
- Tester différents paramètres (ε, t) et observer leur influence sur la représentation.
- Visualiser les vecteurs propres et interpréter leur signification géométrique.

1.4.6 t-SNE

Définition et principe

t-SNE (t-distributed Stochastic Neighbor Embedding) est une méthode non linéaire de réduction de dimension et de visualisation, conçue pour préserver la structure locale des données. Elle est particulièrement efficace pour représenter des clusters et des groupes dans des données complexes.

Objectif : Représenter fidèlement les voisins proches et révéler la structure locale des données en 2D ou 3D.

Algorithme t-SNE

1. Calculer les similarités locales entre points dans l'espace original (probabilités p_{ij}).
2. Placer les points dans l'espace réduit et définir des similarités q_{ij} (loi t de Student).
3. Minimiser la divergence de Kullback-Leibler entre p_{ij} et q_{ij} par descente de gradient.
4. Ajuster les paramètres (perplexité, learning rate) pour optimiser la séparation des clusters.

Exemple : Visualisation MNIST

- Appliquer t-SNE sur les images MNIST pour obtenir une carte 2D où chaque chiffre forme un groupe distinct.
- Comparer la séparation des classes avec PCA et Isomap.

Applications

- Visualisation intuitive de données complexes.
- Analyse exploratoire et détection de clusters.
- Prétraitement pour le clustering ou la classification.

Exercices

- Appliquer t-SNE sur différents jeux de données et analyser la structure obtenue.
- Étudier l'influence de la perplexité et du learning rate sur la visualisation.
- Comparer t-SNE avec PCA et Diffusion Maps sur des données simulées et réelles.

1.4.7 Exercices

- Appliquer la PCA, Isomap et t-SNE sur le jeu de données MNIST. Comparer les visualisations obtenues.
- Sur un Swiss Roll, comparer les résultats de PCA, Isomap et Diffusion Maps.

1.5 Applications pratiques et exercices

1.5.1 TP et exercices avancés

TP1 : Visualisation et réduction de dimension sur MNIST

- Appliquer PCA, Isomap, Diffusion Maps et t-SNE sur le jeu de données MNIST.
- Visualiser les résultats en 2D et 3D, analyser la séparation des classes.
- Corrigé : PCA sépare globalement, t-SNE isole les clusters, Isomap et Diffusion Maps révèlent la structure intrinsèque.

TP2 : Swiss Roll et reconstruction de variétés

- Générer un Swiss Roll synthétique en 3D.
- Appliquer PCA, Isomap et Diffusion Maps, comparer la capacité à "dérouler" la structure.
- Corrigé : PCA échoue, Isomap et Diffusion Maps réussissent à retrouver la structure 2D.

TP3 : Étude de la concentration des distances

- Simuler des points dans $[0,1]^d$ pour différentes valeurs de d.
- Calculer la distance moyenne et l'écart-type, observer la concentration.
- Corrigé : l'écart relatif décroît avec d, illustrant la malédiction de la dimension.

TP4 : Paramètres et robustesse des méthodes

- Étudier l'influence des paramètres (k pour Isomap, ε et t pour Diffusion Maps, perplexité pour t-SNE).
- Tester sur des données bruitées ou incomplètes.
- Corrigé : certains paramètres rendent les méthodes plus robustes ou sensibles au bruit.

1.5.2 Exercices avancés

- Implémenter Isomap et Diffusion Maps sur des jeux de données réels (ex : images, signaux).
- Comparer les avantages et limites de chaque méthode selon la nature des données.
- Discuter les applications possibles en IA, physique, biologie.

Chapter 2

Graphes et diffusion sur données

2.1 Introduction et construction de graphes

La représentation des relations locales entre points d'un nuage de données est essentielle pour de nombreuses méthodes modernes d'apprentissage et d'analyse. Les graphes permettent de modéliser ces relations via différentes constructions :

- **k-NN graph** : chaque point est relié à ses k plus proches voisins, ce qui garantit une connectivité locale contrôlée.
- **ε-graph** : relie tous les points dont la distance est inférieure à ε, le nombre de voisins varie selon la densité locale.

Les graphes peuvent être pondérés pour refléter la similarité entre points :

$$w_{ij} = \exp\Big(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\Big)$$

La pondération permet de capturer la force de connexion locale et prépare le graphe pour les méthodes spectrales.

Exercice : Construisez un graphe k-NN et un ε-graph sur un jeu de données 2D (ex : spirale, cercles imbriqués). Visualisez les différences de connectivité et discutez l'impact du choix de k et ε.

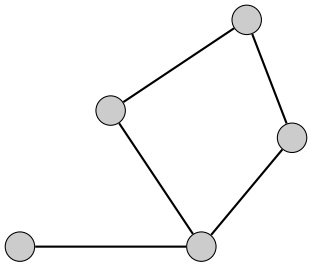


Figure 2.1: Exemple de graphe k-NN sur des points 2D

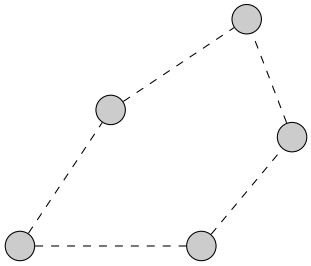


Figure 2.2: Exemple de graphe ε sur des points 2D

2.2 Laplacien de graphe : définitions et interprétations

Soit $G = (V, E, W)$ un graphe pondéré avec matrice de poids W et matrice de degrés D :

$$D_{ii} = \sum_j w_{ij}$$

Le Laplacien non normalisé est $L = D - W$. Les variantes normalisées sont :

$$L_{\text{sym}} = D^{-1/2} L D^{-1/2}, \quad L_{\text{rw}} = D^{-1} L$$

Le Laplacien est semi-défini positif, et son vecteur propre associé à $\lambda_0 = 0$ est constant. Il joue un rôle central dans la diffusion et la réduction de dimension.

2.2.1 Interprétation et propriétés

Le Laplacien est l'analogue discret du Laplacien continu sur une variété. Il mesure la variation d'une fonction f sur le graphe :

$$f^\top L f = \frac{1}{2} \sum_{i,j} w_{ij} (f_i - f_j)^2$$

Un petit $f^\top L f$ indique que f varie peu entre voisins fortement connectés. Les vecteurs propres associés aux plus petites valeurs propres capturent la structure globale du graphe (Eigenmaps), utiles pour la réduction de dimension et le clustering spectral.

Exercice : Pour un graphe simple (ex : chaîne, cycle), calculez le Laplacien et ses valeurs propres. Interprétez les vecteurs propres en termes de structure du graphe.

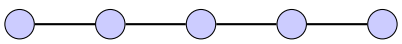


Figure 2.3: Graphe chaîne : illustration du Laplacien

2.3 Diffusion sur graphe et noyau de chaleur

La diffusion sur graphe s'appuie sur la matrice de transition $P = D^{-1}W$ (random walk). La propagation d'une fonction f se fait par :

$$f(t+1) = Pf(t)$$

Le noyau de chaleur discret est donné par :

$$H_t = e^{-tL} \approx \sum_{k=0}^{n-1} e^{-t\lambda_k} \phi_k \phi_k^\top$$

Il mesure la diffusion de la chaleur entre points après un temps t, révélant les structures multi-échelles et la proximité entre points.

Exemple : Initialisez la chaleur sur un noeud d'un graphe et simulez la diffusion pour différents temps t. Observez la propagation et discutez l'effet de la topologie (bottleneck, clusters).

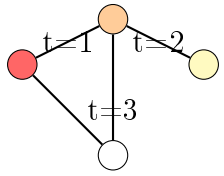


Figure 2.4: Diffusion multi-échelle sur un graphe

2.4 Apprentissage semi-supervisé sur graphe

L'objectif est de propager des labels connus sur un sous-ensemble de points vers l'ensemble du graphe. On résout l'équation de Poisson discrète :

$$Lf = 0 \quad \text{sur les points non étiquetés}$$

avec conditions de Dirichlet sur les points étiquetés. La solution pratique consiste à résoudre le système linéaire restreint :

$$L_{uu} f_u = -L_{u\ell} Y_\ell$$

Les labels se diffusent le long des arêtes pondérées, permettant la classification semi-supervisée.

TP : Classification semi-supervisée sur graphes Utilisez un sous-ensemble étiqueté du jeu MNIST pour propager les labels sur le graphe k-NN. Comparez la performance selon le nombre de labels connus et la structure du graphe.

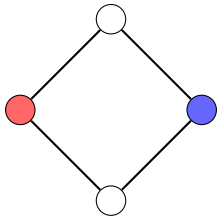


Figure 2.5: Propagation de labels sur un graphe

2.5 Courbure de graphe et topologie intuitive

La courbure discrète mesure comment les voisins d'un noeud sont connectés entre eux. Les points de "bottleneck" relient deux clusters et ralentissent la diffusion. Les valeurs propres du Laplacien capturent la topologie : modes lents pour les grandes structures, modes rapides pour les détails locaux. Cette analyse guide la détection de communautés et la compréhension des structures locales et globales.

Exercice : Construisez un graphe avec deux clusters reliés par un bottleneck. Simulez la diffusion et observez la lenteur de la propagation entre clusters.

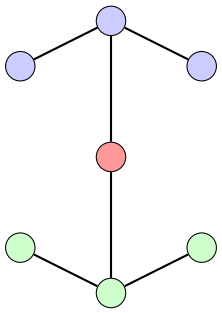


Figure 2.6: Bottleneck : deux clusters reliés par un point central

2.6 Applications pratiques

2.6.1 Classification de digits (MNIST)

On construit un graphe k-NN ou ε-graph à partir des images, pondère les arêtes par similarité, puis applique Laplacian Eigenmaps ou diffusion pour réduire la dimension. Les labels connus sont propagés par Poisson discrete.

2.6.2 Segmentation d'image

Les pixels sont les noeuds du graphe, les arêtes reflètent le voisinage spatial et la similarité de couleur. La diffusion ou Poisson discrete permet de propager des labels initiaux pour obtenir une segmentation cohérente.

2.6.3 Applications modernes

NLP : graphes de mots ou documents, arêtes pondérées par similarité sémantique, diffusion pour classification ou extraction de communautés. **Bioinformatique** : réseaux de gènes ou protéines, propagation de labels, clustering, analyse multi-échelle.

2.7 Résumé du chapitre

- Construction de graphes : k-NN, ε-graph, pondérés
- Laplacien : définitions, interprétations, Eigenmaps
- Diffusion et noyau de chaleur : distances multi-échelles
- Poisson discrete : propagation de labels
- Courbure et topologie : bottlenecks, structures locales et globales
- Exemples : classification, segmentation
- Applications : NLP, bioinformatique

Pour aller plus loin : - Explorez les graphes orientés, les graphes dynamiques et les applications en réseaux sociaux. - Testez la robustesse des méthodes spectrales face au bruit et aux données manquantes. - Comparez diffusion, random walk et autres méthodes de propagation sur différents types de graphes.

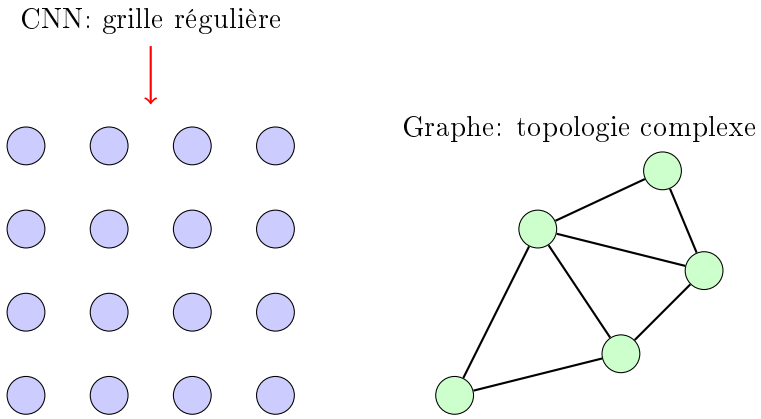
Chapter 3

Motivation et limites des méthodes classiques

3.1 Limites des modèles classiques

Les modèles classiques d'apprentissage (MLP, CNN) sont conçus pour des données euclidiennes (vecteurs, images 2D) et supposent une structure régulière. Ils présentent plusieurs limites :

- Non différentiables ou difficilement applicables sur graphes/variétés
- Difficulté à capturer les relations locales et globales non régulières
- Scalabilité limitée pour des structures complexes



Exemple :

- Les CNN ne peuvent pas propager l'information entre voisins arbitraires sur un graphe
- Les MLP traitent des vecteurs fixes, sans tenir compte de la topologie

3.2 Principe du Message Passing

Pour traiter des données structurées (graphes, variétés), on utilise le principe du **message passing** : chaque nœud met à jour sa représentation en agrégeant les informations de ses voisins.

$$h_i^{(l+1)} = \text{UPDATE}\Big(h_i^{(l)}, \text{AGGREGATE}(\{h_j^{(l)} : j \in \mathcal{N}(i)\})\Big)$$

- **Message** : $m_{ij} = \phi(h_i^{(l)}, h_j^{(l)}, e_{ij})$
- **Agrégation** : somme/moyenne/max des messages
- **Mise à jour** : $h_i^{(l+1)} = \psi(h_i^{(l)}, m_i)$

extbfÀ retenir :

- Le message passing généralise la convolution aux structures non régulières (graphes, maillages).
- Il permet de capturer l'information locale et globale dans des réseaux complexes.

extbfQuestion de réflexion :

- Pourquoi la convolution classique ne fonctionne-t-elle pas sur un graphe ?
- Quelles sont les limites du message passing pour des graphes très grands ou très hétérogènes ?

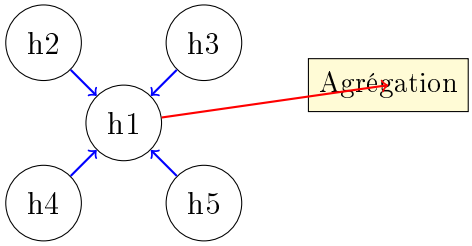
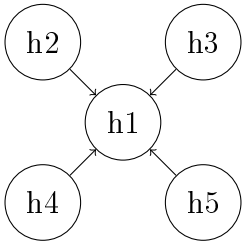


Illustration : propagation de messages



3.3 Exemple concret

Supposons un graphe de personnes, chaque nœud ayant un attribut (âge). À chaque étape, chaque personne met à jour son âge en prenant la moyenne pondérée de son âge et de ceux de ses voisins :

$$h_i^{(l+1)} = \tfrac{1}{2} h_i^{(l)} + \tfrac{1}{2} \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} h_j^{(l)}$$

Après plusieurs itérations, chaque nœud possède une représentation influencée par ses voisins proches et éloignés.

extbfApplication pratique :

- Propagation d'information dans les réseaux sociaux (ex : diffusion d'une opinion ou d'une rumeur).
- Recommandation personnalisée : chaque utilisateur agrège les préférences de ses voisins dans le graphe d'interactions.

3.4 Graph Convolutional Network (GCN)

Le GCN (Kipf & Welling 2017) généralise la convolution aux graphes :

$$H^{(l+1)} = \sigma\Big(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} H^{(l)} W^{(l)}\Big)$$

extbfÀ retenir :

- Le GCN applique une normalisation symétrique pour stabiliser l'apprentissage.
- Il permet de traiter des graphes non réguliers et d'apprendre des représentations structurées.

où $\tilde{A} = A + I$ (adjacence avec boucles), $\tilde{D}$ (degrés), $H^{(l)}$ (représentations), $W^{(l)}$ (poids).

Exemple PyTorch Geometric :

```
import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv

class GCN(torch.nn.Module):
def __init__(self, in_dim, hid_dim, out_dim):
super().__init__()
self.conv1 = GCNConv(in_dim, hid_dim)
self.conv2 = GCNConv(hid_dim, out_dim)

def forward(self, x, edge_index):
x = F.relu(self.conv1(x, edge_index))
x = self.conv2(x, edge_index)
return F.log_softmax(x, dim=1)
```

3.5 Graph Attention Network (GAT)

Le GAT introduit un mécanisme d'attention pour pondérer l'importance des voisins :

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}(a^\top [Wh_i \| Wh_j])\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(\text{LeakyReLU}(a^\top [Wh_i \| Wh_k])\right)}$$

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} W h_j\right)$$

extbfÀ retenir :

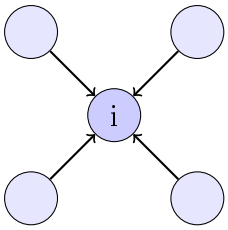
- Le GAT apprend à pondérer chaque voisin selon son importance pour la tâche.
- Il est particulièrement utile pour des graphes hétérogènes ou des relations complexes.

3.6 Comparaison GCN vs GAT

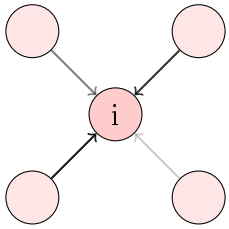
- **GCN** : normalisation fixe, voisins traités uniformément, simplicité
- **GAT** : pondération apprise, plus flexible, coût de calcul plus élevé

extbfQuestion de réflexion :

- Dans quels cas préférer GAT à GCN ?
- Quels sont les avantages et inconvénients de l'attention sur les graphes ?



GCN: voisins traités uniformément



GAT: attention pondérée

3.7 Deux approches : Spectral vs Spatial

- **Spectral** : décomposition spectrale du Laplacien, analyse fréquentielle
- **Spatial** : agrégation locale des voisins, message passing

3.8 Applications de l'IA géométrique

- Physique computationnelle : simulation moléculaire, dynamique des particules
- Biologie structurale : protéines, réseaux de gènes
- IA générative : diffusion sur variétés et graphes

extbfÀ retenir :

- Les GNN sont utilisés dans des domaines variés où la structure des données est essentielle.
- Ils permettent de modéliser des interactions complexes et des phénomènes dynamiques.

3.9 Outils Python pour l'IA géométrique

- **PyTorch Geometric (PyG)** : bibliothèque modulaire basée sur PyTorch
- **DGL (Deep Graph Library)** : efficace, multi-backend
- **Spektral** : intégré à TensorFlow/Keras

Exemple DGL :

```
import dgl
import torch.nn as nn
import torch.nn.functional as F
from dgl.nn import GraphConv

class GCN(nn.Module):
def __init__(self, in_dim, hid_dim, out_dim):
super().__init__()
self.conv1 = GraphConv(in_dim, hid_dim)
self.conv2 = GraphConv(hid_dim, out_dim)

def forward(self, g, features):
x = F.relu(self.conv1(g, features))
x = self.conv2(g, x)
return F.log_softmax(x, dim=1)
```

Exemple Spektral :

```
import tensorflow as tf
from spektral.layers import GCNConv

class GCN(tf.keras.Model):
def __init__(self, out_dim):
super().__init__()
self.conv1 = GCNConv(16, activation="relu")
self.conv2 = GCNConv(out_dim, activation="softmax")

def call(self, inputs):
x, a = inputs
x = self.conv1([x, a])
return self.conv2([x, a])
```

3.10 Conclusion et perspectives

Le deep learning géométrique permet de traiter des données sur graphes, maillages et variétés, en exploitant les relations locales et topologiques complexes. Les techniques clés incluent le message passing, les GNN (GCN, GAT), l'apprentissage spectral et spatial, et l'exploitation de la structure pour l'IA générative et scientifique. Les outils Python modernes (PyTorch Geometric, DGL, Spektral) facilitent l'implémentation de ces modèles. Les applications couvrent la physique, la biologie et l'IA générative. Ouverture : optimisation, IA générative avancée, intégration avec HPC.

Chapter 4

Topologie des données, variétés et transport optimal

4.1 Plan du module

- Introduction à la topologie des données
- Variétés riemanniennes
- Transport optimal
- Applications à l'IA générative
- TP : Mapper (KeplerMapper), mini-projet transport optimal (POT library)

4.2 Topologie des données

Pour étudier la forme globale des données (trous, cycles, composantes), on utilise des techniques comme :

1. **Mapper** : visualisation simplifiée d'un nuage de points
2. **Persistent Homology** : détection de features topologiques robustes

extbfDiagramme de persistance : Barres représentant la durée de vie des composantes.
extttPoints → Graphe simplicial → Filtration → Barcodes

- Chaque barre = un trou ou composante connectée
- Longue barre → feature robuste
- Courte barre → bruit

4.3 Variétés riemanniennes

Une variété lisse $(\mathcal{M}, g)$ avec métrique g définit longueur et angles.

La distance géodésique $d_{\mathcal{M}}(x, y)$ est la longueur minimale d'un chemin sur $\mathcal{M}$.

extbfExemples : sphère, tore, espace de rotations SO(3).

extbfApplications : données sur sphère, pose 3D, embedding non-linéaire.

4.4 Transport optimal

Comparer deux distributions μ et ν avec la distance de Wasserstein :

$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Pi(\mu,\nu)} \int \|x - y\|^p d\gamma(x, y) \right)^{1/p}$$

- Comparaison distributions réelles vs générées
- Génération de données réalistes avec structures géométriques

4.5 Applications à l'IA générative

- Score-based diffusion : génération par gradient de log-densité
- Transport optimal : mesurer distances entre distributions générées et réelles
- Topologie persistante : régulariser génération pour préserver cycles/structures
- Graphes et variétés : génération de molécules, maillages 3D, images structurées

4.6 TP : Mapper et transport optimal

extbfTP Mapper (KeplerMapper) : Visualiser la structure globale d'un nuage de points avec Mapper. Utiliser la bibliothèque KeplerMapper pour explorer les cycles et composantes.

extbfMini-projet transport optimal (POT library) : Implémenter la comparaison de distributions et le calcul de la distance de Wasserstein avec la bibliothèque POT. Appliquer à des données réelles ou simulées.

4.7 Synthèse

- Topologie : Mapper, Persistent Homology $\rightarrow$ analyser la forme globale des données
- Géométrie avancée : variétés riemanniennes et distances géodésiques
- Transport optimal : distance de Wasserstein pour comparer distributions
- Applications IA générative : score-based diffusion, OT, régularisation topologique