Géométrie des Données et Apprentissage sur Variétés

Module 1 – Introduction enrichie avec exercices

Mastère Spécialisé HPC-AI

November 19, 2025

Plan

Module 3 : plan

- Limites des modèles classiques : pas différentiables / scalables
- Principe du Message passing
- GCN, GAT : mécanismes principaux
- Applications IA géométrique :
  - Physique (simulation moléculaire)
  - Biologie (protéines)
  - IA générative (diffusion sur variétés)
- Outils Python : PyTorch Geometric, DGL, Spektral
- TP : Mini-GNN avec PyTorch Geometric ou Deep Graph Library.

Motivation : pourquoi le deep learning géométrique ?

- Les données modernes sont souvent non-euclidiennes : graphes, maillages, variétés
- Les modèles classiques (MLP, CNN) supposent une structure régulière :
  - vecteurs fixes ou images 2D
  - pas adaptés à la topologie complexe
- Objectif : apprendre des représentations sur des structures géométriques

Limites des modèles classiques

- Pas différentiables sur graphes/variétés
- Difficulté à capturer relations locales/globales non régulières
- Scalabilité limitée à des structures complexes

```
Vecteurs fixes / images 2D
          |
Pas de propagation entre voisins arbitraires
          |
Graphe ou variété → besoin de modèles géométriques
```

Comparatif : classique vs géométrique

- MLP/CNN :
  - Entrée : vecteurs fixes / pixels
  - Convolutions régulières
  - Limité aux grilles Euclidiennes
- Deep Learning Géométrique :
  - Entrée : graphes, points sur variété
  - Message passing / convolutions sur graphes
  - Exploite topologie et voisinage local

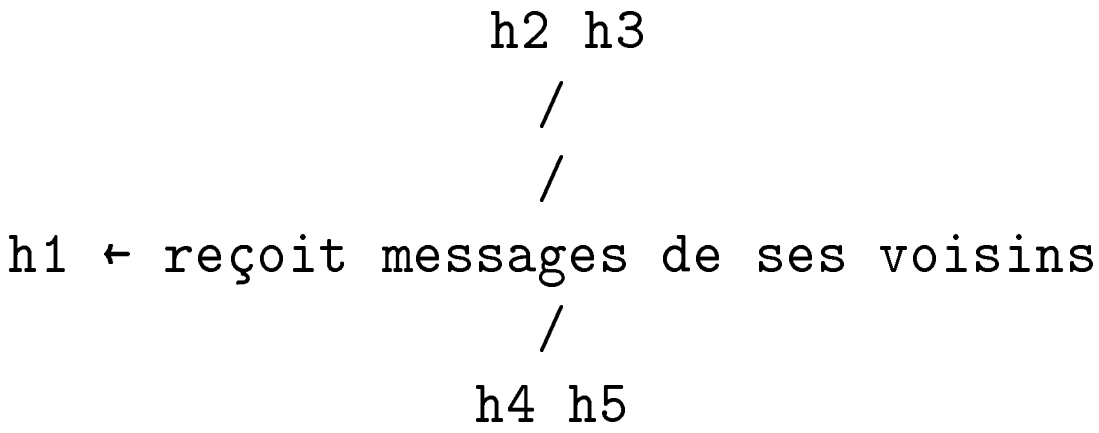
Message Passing : idée générale

- Chaque nœud met à jour sa représentation en agrégeant les informations de ses voisins.
- Formule générale :

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

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

Illustration : propagation de messages



- $h_1^{(l+1)}$ dépend de $h_1^{(l)}$ et de l'agrégation des messages venant de h_2, h_3, h_4, h_5.
- Propagation couche par couche : information diffuse sur tout le graphe.

Exemple simple

- Noeuds = personnes, attributs = âge initial
- Agrégation = moyenne des âges des voisins
- Mise à jour = moyenne pondérée de son âge et de ses voisins

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

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

Graph Convolutional Network (GCN)

- Formule de base (Kipf Welling 2017) :

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

où :

  - $\tilde{A} = A + I$: matrice d'adjacence avec boucles
  - $\tilde{D}$: matrice diagonale des degrés
  - $H^{(l)}$: représentations des nœuds à la couche l
  - $W^{(l)}$: poids entraînables

- Normalisation symétrique évite l'explosion/vanishing.
- Interprétation : moyenne pondérée des voisins + transformation linéaire.

Exemple intuitif : GCN

- Imagine un graphe de documents liés par des citations.
- Chaque nœud = vecteur TF-IDF d'un document.
- Un GCN agrège les vecteurs des voisins $\rightarrow$ information contextuelle.
- Après plusieurs couches : un document est représenté par son contenu **et** celui des documents proches.

Graph Attention Network (GAT)

- Chaque voisin n'a pas la même importance !
- Coefficients d'attention :

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

- Mise à jour :

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

- Idée clé : pondération adaptative des voisins via mécanisme d'attention.

Comparaison GCN vs GAT

GCN

- Normalisation fixe (degré)
- Voisins traités de façon uniforme
- Simplicité, efficacité

GAT

- Pondération apprise entre voisins
- Plus flexible et expressif
- Coût de calcul plus élevé

Deux approches de la convolution sur graphe

- **Spectral** : basé sur la décomposition spectrale du Laplacien du graphe.
- **Spatial** : basé sur l'agrégation locale des voisins (message passing).
- Les deux approches sont équivalentes dans certains cas mais avec des compromis différents.

Approche spectrale

- Laplacien normalisé : $L = I - D^{-1/2} A D^{-1/2}$
- Décomposition en valeurs propres : $L = U \Lambda U^\top$
- Convolution spectrale définie comme :

$$g_\theta * x = U g_\theta(\Lambda) U^\top x$$

où $g_\theta(\Lambda)$ agit comme un filtre en fréquence.

- Avantage : interprétation en termes de fréquences du graphe.
- Limites : coût élevé, dépend de la structure exacte du graphe.

Approche spatiale

- Directement définir une agrégation des voisins :

$$h_i^{(l+1)} = \sigma\Big(\sum_{j \in \mathcal{N}(i)} \frac{1}{c_{ij}} W h_j^{(l)} \Big)$$

  où c_{ij} est un facteur de normalisation (ex: degré).
- Interprétation : propagation de l'information comme dans le message passing.
- Avantage : plus scalable, indépendant de la décomposition spectrale.
- Limite : pas de contrôle explicite des fréquences.

Comparaison : spectral vs spatial

Spectral

- Basé sur le Laplacien
- Analyse fréquentielle possible
- Limité aux graphes fixes

Spatial

- Agrégation des voisins
- Plus intuitif et local
- Fonctionne pour graphes dynamiques

Applications de l'IA géométrique

- Les GNN et méthodes géométriques sont appliquées à des domaines variés :
  1. Physique computationnelle (simulation moléculaire, dynamique des particules)
  2. Biologie structurale (protéines, réseaux de gènes)
  3. IA générative (diffusion sur variétés et graphes)
- Atout : respect des symétries, invariances et topologie des données.

Physique : simulation moléculaire

- Molécules représentées comme graphes (atomes = nœuds, liaisons = arêtes).
- GNN utilisés pour approximer :
  - Énergies de configuration
  - Forces interatomiques
- Exemple : **SchNet**, **PhysNet**, modèles équivariants SE(3).
- Applications : accélération de la dynamique moléculaire, découverte de matériaux.

Biologie : protéines et génomique

- Protéines = graphes 3D d'acides aminés.
- GNN $\rightarrow$ prédiction de structure, fonction, interactions.
- Exemples :
  - **AlphaFold** : géométrie + réseaux de neurones
  - GNN pour réseaux de régulation génétique
- Impact : compréhension des maladies, conception de médicaments.

IA générative sur variétés et graphes

- Génération de graphes structurés :
  - Molécules valides
  - Réseaux complexes
- Méthodes récentes :
  - Diffusion sur graphes
  - Score-based generative models
- Applications : design de médicaments, génération de maillages 3D, chimie computationnelle.

Synthèse des applications

- **Physique** : simulation moléculaire, dynamique des particules
- **Biologie** : prédiction de structure de protéines, réseaux biologiques
- **IA générative** : création de graphes complexes, molécules, géométries

Message clé : les GNN exploitent la structure géométrique et topologique $\rightarrow$ meilleures généralisations et applications pratiques.

Outils Python pour l'IA géométrique

- Frameworks populaires :
  1. **PyTorch Geometric (PyG)** : bibliothèque modulaire basée sur PyTorch
  2. **DGL (Deep Graph Library)** : efficace, multi-backend
  3. **Spektral** : intégré à TensorFlow/Keras
- Fournissent des implémentations de GCN, GAT, GraphSAGE, DiffPool, etc.
- Support de grands graphes et du GPU.

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)
```

Exemple : DGL (Deep Graph Library)

```
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 (TensorFlow/Keras)

```
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])
```

Synthèse des outils Python

PyTorch Geometric

- Basé sur PyTorch
- Grande communauté
- Très flexible

DGL

- Hautes performances
- Multi-backend (PyTorch, MXNet, TensorFlow)
- Optimisé pour graphes massifs

Spektral

- Simple, basé sur Keras
- Idéal pour prototypage rapide
- Moins optimisé grands graphes

Module 3 : Conclusion et perspectives

- Le deep learning géométrique permet de traiter :
  - Données sur graphes, maillages et variétés
  - Relations locales et topologiques complexes
- Techniques clés :
  - Message passing et GNN (GCN, GAT)
  - Apprentissage spectral vs spatial
  - Exploitation de la structure pour IA générative et scientifique
- Applications concrètes :
  - Physique : simulation moléculaire
  - Biologie : protéines, réseaux génétiques
  - IA générative : diffusion sur graphes/variétés
- Outils Python modernes pour implémenter facilement ces modèles : **PyTorch Geometric, DGL, Spektral**
- **Ouverture** : vers l'optimisation, l'IA générative avancée et l'intégration avec HPC.