Graph Analytics

Redha Moulla

Ecole Centrale de Casablanca

Automne 2023

Plan

- Introduction aux graphes
- Apprentissage sur les graphes
- Node embeddings
- Graph Neural Networks

Applications des graphes

Les graphes sont des structures de données essentielles en machine learning, offrant une manière flexible de représenter des relations et des interdépendances complexes.

- **Représentation des données** : Modélisation de relations complexes dans divers domaines tels que les réseaux sociaux, les systèmes biologiques, et les réseaux de communication.
- **Analyse des réseaux sociaux** : Étude des interactions et des communautés dans les réseaux sociaux.
- **Systèmes de recommandation** : Utilisation de graphes pour améliorer la précision des systèmes de recommandation.
- **Détection de fraude** : Analyse de graphes pour identifier des schémas suspects dans les transactions financières.
- **Optimisation des réseaux** : Graphes dans la gestion des réseaux de transport, des réseaux informatiques, etc.

Introduction aux Graphes

Definition

Un graphe est une structure de données composée d'un ensemble de *nœuds* (ou *sommet*) et d'un ensemble d'*arêtes* (ou *liens*) qui connectent ces nœuds. Les graphes sont utilisés pour modéliser des relations et des interconnexions dans divers domaines.

- **Représentations des graphes** : Les graphes peuvent être représentés de plusieurs manières, notamment par des *matrices d'adjacence* ou des *listes d'adjacence*.
- **Graphes orientés et non orientés** : Les graphes peuvent être *orientés* (où les arêtes ont une direction) ou *non orientés* (où les arêtes n'ont pas de direction).
- **Graphes pondérés** : Les graphes peuvent aussi être *pondérés* où chaque arête se voit assigner un poids.

Représentation des graphes

Definition

Il y a plusieurs manières de réprésenter les relations et les structures des graphes sous forme mathématique ou visuelle, pour faciliter leur analyse et leur traitement algorithmique.

Représentations des graphes

- **Matrice d'adjacence** : Un tableau 2D indiquant la présence ou l'absence d'arêtes entre les paires de nœuds.
- **Liste d'adjacence** : Une collection de listes associées à chaque nœud détaillant ses voisins directs.
- **Représentation graphique** : Une visualisation où les nœuds sont des points et les arêtes sont des lignes les reliant.

Graphes non orientés

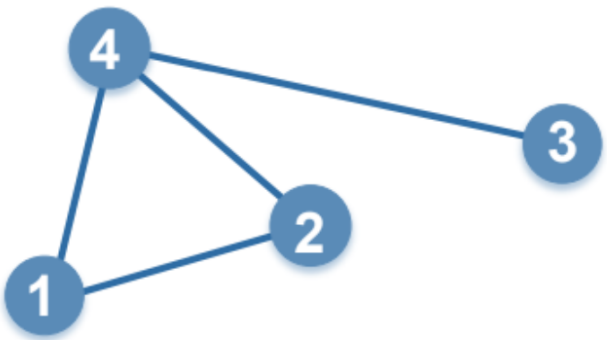
Definition

Un **graphe non orienté** est un graphe où les arêtes ne possèdent pas de direction. Les arêtes indiquent simplement une relation bilatérale entre les nœuds.

Matrice d'adjacence A

La matrice d'adjacence d'un graphe non orienté est symétrique et définie par :

$$A_{ij} = \begin{cases} 1 & \text{si le nœud } i \text{ est connecté avec le nœud } j, \\ 0 & \text{sinon.} \end{cases}$$



$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Graphes orientés

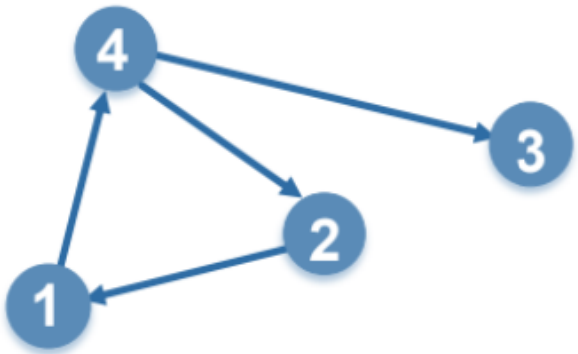
Definition

Un **graphe orienté** est un graphe où les arêtes ont une direction, indiquant une relation unidirectionnelle entre les nœuds.

Matrice d'adjacence A

La matrice d'adjacence d'un graphe orienté reflète la direction des arêtes et est définie par :

$$A_{ij} = \begin{cases} 1 & \text{si une arête va du nœud } i \text{ au nœud } j, \\ 0 & \text{sinon.} \end{cases}$$



$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Graphes pondérés

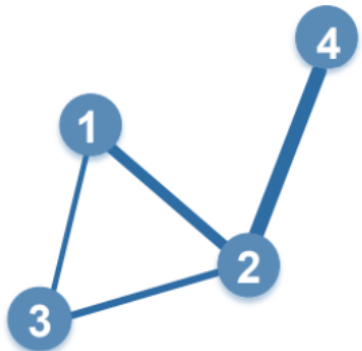
Definition

Un **graphe pondéré** est un graphe où chaque arête a une valeur numérique associée, appelée poids, qui représente généralement le coût, la longueur ou la capacité de cette connexion.

Matrice d'adjacence pondérée

La matrice d'adjacence d'un graphe pondéré indique le poids de chaque arête :

$$W_{ij} = \begin{cases} \text{poids de l'arête} & \text{si une arête relie } i \text{ à } j, \\ 0 & \text{sinon.} \end{cases}$$



$$W = \begin{bmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{bmatrix}$$

Machine learning sur graphes

L'apprentissage sur graphes peut être classé en trois catégories principales : **supervisé**, **non supervisé**, et **semi-supervisé**.

- **Apprentissage supervisé** : Dans l'apprentissage supervisé, on dispose d'un graphe $G = (V, E)$ avec des étiquettes ou des labels pour certains ou tous les nœuds. L'objectif est d'apprendre une fonction de prédiction $f : V \to Y$ où Y est l'ensemble des labels.
- **Apprentissage non supervisé** : Dans l'apprentissage non supervisé, le graphe $G = (V, E)$ ne possède pas d'étiquettes. L'objectif est de découvrir la structure sous-jacente du graphe, souvent à travers des tâches telles que la détection de communauté, l'embedding de graphes, ou la génération de graphes.
- **Apprentissage semi-supervisé** : Cette approche combine des éléments des deux précédentes, utilisant à la fois des données étiquetées et non étiquetées. Elle est particulièrement utile dans les situations où seuls quelques nœuds du graphe sont étiquetés. L'objectif est d'exploiter les données non étiquetées pour améliorer la précision et la généralisation du modèle appris.

Extraction de caractéristiques dans les graphes

L'extraction de caractéristiques dans les graphes implique la transformation des informations structurées d'un graphe en un format exploitable par les algorithmes de machine learning.

Méthodes d'extraction

- **Caractéristiques des nœuds** : Propriétés individuelles des nœuds, telles que le degré, la centralité, ou des attributs spécifiques au domaine.
- **Caractéristiques des arêtes** : Informations sur les relations entre les nœuds, comme le poids des arêtes ou le type de connexion.
- **Caractéristiques globales du graphe** : Mesures globales, telles que le nombre total de nœuds/arêtes, la densité, ou la distribution des degrés.

Techniques plus avancées

- **Embeddings de nœuds** : Utilisation de techniques comme Node2Vec ou Graph Neural Networks pour générer des représentations vectorielles des nœuds.
- **Analyse spectrale** : Utilisation du spectre du Laplacien du graphe pour extraire des informations structurelles.

Le Degré d'un noeud

Definition

Le **degré d'un noeud** dans un graphe est le nombre d'arêtes connectées à ce noeud. Dans un graphe non orienté, c'est le nombre total d'arêtes incidentes. Dans un graphe orienté, on distingue le degré entrant et sortant.

Degré dans les graphes non orientés

Dans un graphe non orienté, le degré d'un noeud v, noté $d(v)$, est le nombre d'arêtes qui le relient à d'autres nœuds.

$$d(v) = |\{e \in E : v \in e\}|$$

où E est l'ensemble des arêtes du graphe.

Degré dans les graphes orientés

Dans un graphe orienté :

- **Degré entrant** $(d_{in}(v))$: Nombre d'arêtes entrant dans le noeud v.
- **Degré sortant** $(d_{out}(v))$: Nombre d'arêtes sortant du noeud v.

Centralité de proximité

Définition

La **centralité de proximité** d'un nœud reflète sa proximité moyenne par rapport à tous les autres nœuds dans le graphe. Elle est inversement proportionnelle à la somme des distances les plus courtes de ce nœud à tous les autres nœuds.

$$C_p(v) = \frac{1}{\sum_{u \in V} d(v,u)}$$

où $d(v,u)$ est la distance la plus courte de v à u.

Interprétation :

- Un nœud avec une centralité de proximité élevée peut rapidement interagir avec tous les autres nœuds du réseau.
- Utile pour identifier les nœuds qui peuvent efficacement propager des informations.

Centralité d'intermédiarité

Définition

La **centralité d'intermédiarité** d'un nœud est la fréquence à laquelle il apparaît sur les plus courts chemins entre les paires de nœuds dans le graphe.

$$C_b(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

où σ_{st} est le nombre total de plus courts chemins de s à t et $\sigma_{st}(v)$ est le nombre de ces chemins passant par v.

Interprétation

- Un nœud avec une centralité d'intermédiarité élevée agit comme un point de passage ou un pont dans le réseau.
- Indique les nœuds qui jouent un rôle clé dans la facilitation de la communication entre différents groupes ou communautés dans le graphe.

Centralité de vecteur propre

Définition

La **centralité de vecteur propre** mesure l'influence d'un nœud en tenant compte de l'importance de ses voisins. Elle est basée sur l'idée qu'être connecté à des nœuds qui sont eux-mêmes importants augmente la propre importance du nœud.

$$C_v(v) = \frac{1}{\lambda} \sum_{t \in M(v)} A_{vt} C_v(t)$$

où $M(v)$ est l'ensemble des voisins de v, A_{vt} est l'élément de la matrice d'adjacence entre v et t, et λ est une constante telle que la solution soit stable.
Interprétation :

- Un nœud avec une centralité de vecteur propre élevée est non seulement connecté à de nombreux nœuds, mais est aussi connecté à des nœuds qui sont eux-mêmes centraux.
- Utile dans les réseaux sociaux pour identifier les influenceurs ou dans les réseaux de citations pour détecter des articles de recherche influents.

Centralité des arêtes

La centralité des arêtes dans un graphe mesure l'importance ou l'influence des arêtes plutôt que des nœuds. Cette mesure est cruciale pour identifier les connexions clés au sein d'un réseau.

Importance des arêtes

- Permet de comprendre quelles connexions au sein du graphe sont les plus influentes ou critiques.
- Utile pour l'analyse des réseaux de transport, les réseaux sociaux, et les réseaux biologiques.

Arêtes : la distance

Definition

La **distance** dans un graphe est la longueur du plus court chemin entre deux nœuds. Elle est souvent utilisée pour mesurer la proximité ou l'éloignement entre les nœuds dans le graphe.

Calcul de la Distance :

- La distance entre deux nœuds u et v est le nombre minimal d'arêtes qu'il faut parcourir pour aller de u à v.
- Peut être calculée à l'aide d'algorithmes tels que Dijkstra ou Bellman-Ford pour les graphes pondérés, et BFS (Breadth-First Search) pour les graphes non pondérés.

Remarque : La notion de distance peut être étendue pour inclure les poids des arêtes, les directions, et d'autres contraintes spécifiques au domaine.

Arêtes : caractéristiques de voisinage local

Nombre de Nœuds en Commun :

- Le nombre de voisins communs entre deux nœuds.

Coefficient de Jaccard :

- Mesure la similarité entre les ensembles de voisins de deux nœuds.
- Calculé comme le rapport entre le nombre de voisins communs et le nombre total de voisins uniques des deux nœuds.

$$Jaccard(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$$

où $N(u)$ et $N(v)$ sont les ensembles de voisins des nœuds u et v.

Indice d'Adamic-Adar :

- Il pondère les voisins communs en fonction de leur degré. Plus le degré d'un voisin commun est faible, plus sa contribution à l'indice est importante.

$$AdamicAdar(u, v) = \sum_{w \in N(u) \cap N(v)} \frac{1}{\log |N(w)|}$$

où w est un voisin commun de u et v.

Voisinage global : indice de Katz

Definition

L'**indice de Katz** est une mesure de centralité dans un graphe qui tient compte non seulement des voisins immédiats d'un nœud, mais aussi de tous les autres nœuds du réseau, avec une pondération décroissante pour les nœuds plus éloignés.

$$Katz(v) = \sum_{k=1}^{\infty} \sum_{u \in V} \alpha^k (A^k)_{uv}$$

où α est un facteur d'atténuation, A est la matrice d'adjacence du graphe, et $(A^k)_{uv}$ représente le nombre de chemins de longueur k de u à v.

Applications :

- Identifie les nœuds influents dans les réseaux sociaux et, plus généralement, dans les réseaux complexes.
- Utilisé dans les systèmes de recommandation et l'analyse de liens.

Node embedding

Node embedding

Definition

Le **node embedding** est une méthode de représentation des nœuds d'un graphe dans un espace vectoriel de faible dimension, tout en préservant la structure et les propriétés du graphe.

Utilité :

- Facilite l'application de techniques de machine learning sur les graphes en fournissant une représentation numérique des nœuds.
- Permet de capturer à la fois les informations topologiques et les attributs des nœuds.

Techniques communes :

- **DeepWalk**, **Node2Vec** : Utilisent des séquences de marche aléatoire pour apprendre les embeddings.
- **Graph Neural Networks (GNNs)** : Apprennent des embeddings en intégrant les caractéristiques des voisins d'un nœud.

Node embedding

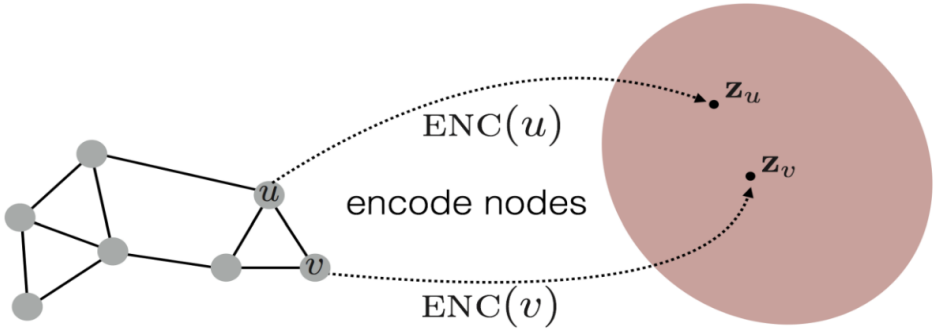


Figure: Transformation des noeuds en vecteurs dans un espace de dimension plus réduite.

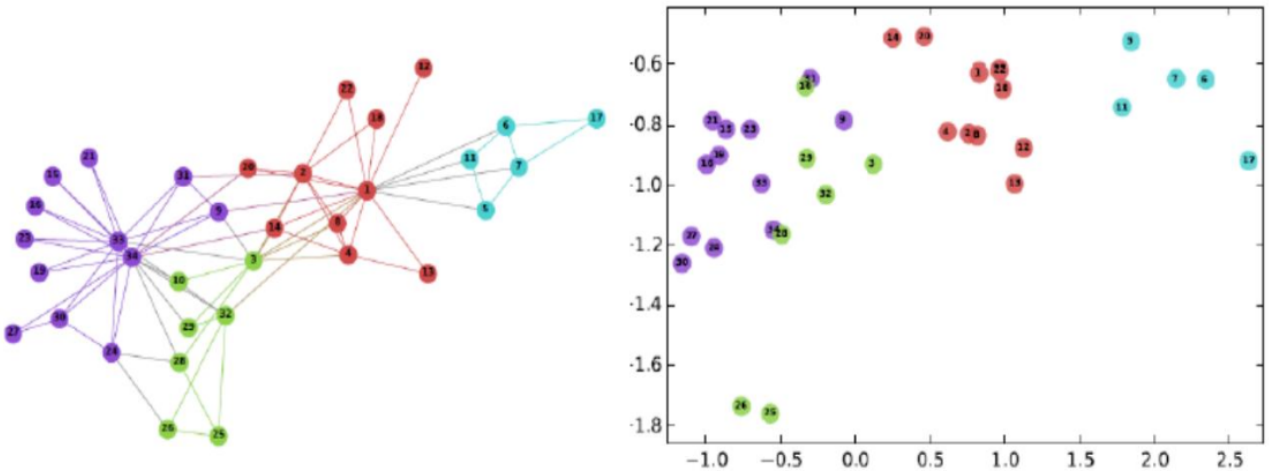


Figure: Représentation des embeddings dans l'espace cible.

Node2Vec - Formalisme mathématique

Node2Vec est un algorithme qui apprend des représentations vectorielles de nœuds dans un graphe en utilisant des marches aléatoires biaisées.

Définition de la marche aléatoire biaisée : Une marche aléatoire biaisée est définie par deux paramètres, p et q, qui déterminent la probabilité de transition d'un nœud à l'autre :

- p (paramètre de retour) : Contrôle la probabilité de revisiter un nœud précédent.
- q (paramètre d'exploration) : Gère l'équilibre entre l'exploration des voisins immédiats et des régions éloignées.

Probabilité de transition : La probabilité de transition d'un nœud v à un nœud x est donnée par :

$$\pi_{vx} = \begin{cases} \frac{1}{p} & \text{si } x \text{ est le nœud précédent} \\ 1 & \text{si } x \text{ est un voisin direct de } v \\ \frac{1}{q} & \text{si } x \text{ est éloigné de } v \end{cases} \tag{1}$$

Ces probabilités sont normalisées pour chaque nœud.

Génération de marches aléatoires

Node2Vec génère des marches aléatoires biaisées pour explorer efficacement le graphe :

- **Initialisation** : Chaque marche aléatoire commence à un nœud choisi aléatoirement.
- **Sélection des voisins** : À chaque étape, le prochain nœud est choisi parmi les voisins du nœud actuel, en utilisant la probabilité de transition basée sur p et q.

Formulation mathématique : La probabilité de transition d'un nœud v vers un nœud voisin x est donnée par :

$$\pi_{vx} = \frac{\alpha_{pq}(v, x)}{Z} \tag{2}$$

où $\alpha_{pq}(v, x)$ est un poids basé sur p et q, et Z est un facteur de normalisation.

- $\alpha_{pq}(v, x)$ définit la probabilité de se déplacer vers le nœud x en prenant en compte la structure locale et globale du graphe.

Node2Vec

Après avoir généré des séquences de nœuds via des marches aléatoires, Node2Vec utilise un modèle de type Skip-Gram pour apprendre les embeddings.
Modèle Skip-Gram : Le modèle vise à maximiser la probabilité conditionnelle des nœuds voisins dans les séquences générées.

$$\max \sum_{v \in V} \sum_{u \in N(v)} \log \Pr(u|v) \tag{3}$$

où

$$\Pr(u|v) = \frac{\exp(u \cdot v)}{\sum_{w \in V} \exp(w \cdot v)} \tag{4}$$

- V est l'ensemble des nœuds dans le graphe.
- $N(v)$ est l'ensemble des nœuds voisins de v dans les marches aléatoires.
- $f(v)$ est l'embedding du nœud v.

Optimisation : Node2Vec utilise des techniques comme le Negative Sampling pour optimiser l'apprentissage des embeddings.

Node2Vec

Node2Vec utilise le modèle Skip-Gram pour apprendre les embeddings de nœuds :

- **Objectif d'apprentissage** : Maximiser la similarité des embeddings des nœuds qui apparaissent fréquemment ensemble dans les marches aléatoires.
- **Negative sampling** : Améliore l'efficacité de l'apprentissage en échantillonnant des nœuds "négatifs" pour chaque nœud cible, accélérant ainsi la convergence.

Fonction objective :

$$\max \sum_{v \in V} \left(\sum_{u \in N(v)} \log \sigma(u \cdot v) + \sum_{k \in K} \mathbb{E}_{k \sim P(n)} [\log \sigma(-k \cdot v)] \right) \tag{5}$$

où σ est la fonction sigmoid, $N(v)$ est l'ensemble des nœuds voisins de v, et K est un ensemble de nœuds négatifs échantillonnés selon la distribution $P(n)$.

Introduction à DeepWalk

Definition

DeepWalk est une technique d'embedding de graphes qui utilise des marches aléatoires uniformes pour apprendre des représentations vectorielles des nœuds dans un espace de faible dimension.

Caractéristiques principales :

- **Marches aléatoires uniformes** : Explore le graphe en effectuant des marches aléatoires, traitant chaque transition de nœud avec une probabilité égale.
- **Modèle de langage** : Utilise le modèle Skip-Gram de Word2Vec pour apprendre des embeddings à partir des séquences de nœuds générées par les marches aléatoires.
- **Capturer la structure du voisinage** : Les embeddings appris reflètent les structures de voisinage des nœuds dans le graphe.

Génération de marches aléatoires dans DeepWalk

DeepWalk génère des marches aléatoires pour chaque nœud du graphe, servant de base à l'apprentissage des embeddings.

Processus de génération :

1. Chaque marche aléatoire commence à un nœud sélectionné aléatoirement.
2. La marche se poursuit en choisissant aléatoirement un voisin à chaque étape, pour une longueur prédéfinie.

Importance des marches aléatoires :

- Les séquences de nœuds générées imitent les "phrases" dans le langage naturel, capturant les contextes locaux des nœuds dans le graphe.
- Permettent d'extraire des informations sur les structures locales et globales du graphe sans nécessiter de connaissances spécifiques sur sa topologie.

Apprentissage des embeddings avec DeepWalk

DeepWalk utilise le modèle Skip-Gram pour transformer les marches aléatoires en embeddings de nœuds.

Méthodologie :

- Chaque nœud dans une marche aléatoire est traité comme un "mot" et la séquence complète comme une "phrase".
- Le modèle Skip-Gram apprend à prédire les nœuds voisins dans une séquence, en maximisant la probabilité des nœuds voisins étant donné un nœud cible.

Fonction objective :

$$\max \sum_{v \in V} \left(\sum_{u \in N(v)} \log \Pr(u|f(v)) \right) \tag{6}$$

où V est l'ensemble des nœuds, $N(v)$ est l'ensemble des nœuds dans le voisinage de la marche aléatoire de v, et $f(v)$ est l'embedding du nœud v.

Optimisation : Utilise des techniques comme le Negative Sampling pour améliorer l'efficacité de l'apprentissage.

Apprentissage sur graphes

Apprentissage supervisé sur graphes

L'apprentissage supervisé sur graphes se concentre sur l'apprentissage de modèles prédictifs pour des graphes où des étiquettes sont disponibles pour certains éléments (nœuds, arêtes ou sous-graphes).

Tâches classiques :

- **Classification de nœuds** : Prédire l'étiquette de nœuds individuels basée sur leurs caractéristiques et leurs relations dans le graphe.
- **Classification d'arêtes** : Prédire l'existence ou la nature d'une relation entre deux nœuds.
- **Classification de graphes** : Prédire une étiquette pour un graphe entier, par exemple, dans la reconnaissance de motifs dans les réseaux chimiques ou biologiques.
- **Prédiction de liens** : Estimer la probabilité d'existence d'une arête non observée dans le graphe.

Apprentissage non supervisé sur graphes

L'apprentissage non supervisé sur graphes se concentre sur l'extraction de modèles et de structures utiles à partir de graphes qui ne disposent pas d'étiquettes pour leurs éléments (nœuds, arêtes ou sous-graphes).

Tâches classiques :

- **Clustering de nœuds** : Regrouper les nœuds en fonction de leurs similarités en termes de caractéristiques et de connectivité.
- **Détection de communautés** : Identifier des groupes de nœuds étroitement liés qui forment des communautés ou des modules dans le graphe.
- **Réduction de dimensionnalité** : Réduire la dimensionnalité des données du graphe pour une visualisation ou une analyse plus facile, souvent via des techniques d'embedding.
- **Génération de graphes** : Créer de nouveaux graphes qui imitent la distribution statistique de graphes réels ou synthétiques.

Laplacien d'un graphe

Definition

Le Laplacien L d'un graphe non orienté $G = (V, E)$ est défini par $L = D - A$, où A est la matrice d'adjacence et D est la matrice diagonale des degrés. Pour $i, j \in \{1, \ldots, |V|\}$, $A_{ij} = 1$ si $(i, j) \in E$ et 0 sinon, et $D_{ii} = \deg(i)$.

Propriétés du Laplacien :

- Matrice symétrique et semi-définie positive.
- Les valeurs propres sont réelles et non négatives.
- Le nombre de valeurs propres nulles indique le nombre de composantes connexes.
- $x^T L x = \frac{1}{2} \sum_{i,j} A_{ij}(x_i - x_j)^2$ pour tout $x \in \mathbb{R}^{|V|}$.

Clustering spectral

Definition

Le **clustering spectral** est une technique de partitionnement de graphes qui utilise les propriétés spectrales (valeurs et vecteurs propres) du Laplacien de graphe pour identifier des clusters de nœuds.

Principes clés :

- Basé sur le principe que les vecteurs propres du Laplacien peuvent fournir des insights importants sur la structure en clusters du graphe.
- Cherche à minimiser la coupe de graphe entre les différents clusters tout en maximisant la cohésion à l'intérieur de chaque cluster.

Fondements Mathématiques du Clustering Spectral

La formule de base du clustering spectral est centrée sur la minimisation de la coupe de graphe entre différents clusters.
Formulation du problème :

$$\min_{\substack{S_1,\ldots,S_k \\ \text{partition de } V}} \sum_{i=1}^{k} \frac{W(S_i, \bar{S}_i)}{|S_i|} \tag{7}$$

où $W(S_i, \bar{S}_i)$ représente le poids total des arêtes entre les sommets dans S_i (un cluster) et ceux dans $\bar{S}_i$ (le reste du graphe).
Explication de la formule :

- $S_1, \ldots, S_k$ sont les clusters à identifier dans le graphe.
- $W(S_i, \bar{S}_i)$ mesure la somme des poids des arêtes reliant les nœuds à l'intérieur de S_i avec ceux à l'extérieur.
- $|S_i|$ est le nombre de sommets dans le cluster S_i.
- L'objectif est de trouver une partition des sommets minimisant cette somme, ce qui revient à minimiser les connexions inter-clusters tout en conservant des clusters équilibrés en termes de taille.

Algorithme de Clustering Spectral

L'algorithme de clustering spectral se décline selon les étapes suivantes :

1. Calculer le Laplacien normalisé $L_{\text{norm}} = D^{-1/2} L D^{-1/2}$.
2. Trouver les k plus petits vecteurs propres de L_{norm} pour former la matrice $U \in \mathbb{R}^{n \times k}$.
3. Normaliser les lignes de U pour créer U_{norm}.
4. Appliquer un algorithme de clustering, tel que k-means, sur U_{norm}.

Implications

- Permet de révéler des clusters naturels dans des graphes même en l'absence de connexions clairement définies.
- Particulièrement utile pour les données structurées en graphes complexes.

Apprentissage semi-supervisé sur graphes

Definition

L'apprentissage semi-supervisé sur graphes combine les approches supervisées et non supervisées pour l'analyse de graphes, en exploitant à la fois les données étiquetées et non étiquetées.

Caractéristiques principales

- **Utilisation de données partiellement étiquetées** : Seule une fraction des nœuds ou des arêtes du graphe a des étiquettes.
- **Propagation d'informations** : Les informations des éléments étiquetés sont utilisées pour inférer les étiquettes des éléments non étiquetés.
- **Apprentissage basé sur la structure du graphe** : La structure du graphe est exploitée pour améliorer la prédiction des étiquettes.

Applications :

- Classification de nœuds dans les réseaux sociaux.
- Analyse de réseaux biologiques et chimiques.
- Systèmes de recommandation où les interactions partielles sont connues.

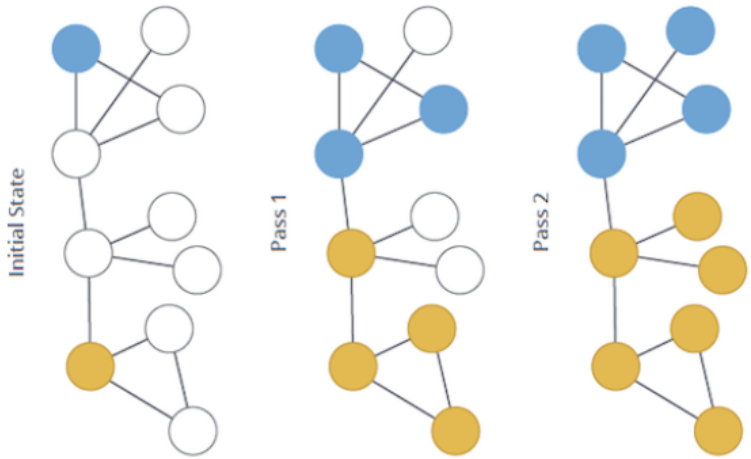
Label propagation

Definition

Le **label propagation** est une méthode d'apprentissage semi-supervisé sur graphes où les étiquettes (labels) sont diffusées à travers le réseau à partir d'un ensemble initial de nœuds étiquetés.

Principe :

- Les étiquettes sont propagées aux nœuds voisins en fonction de leur connectivité et des étiquettes voisines.
- L'algorithme itère jusqu'à ce que le système atteigne un état stable ou que le nombre maximal d'itérations soit atteint.



Algorithme de Label propagation

Étapes de l'algorithme :

1. Initialiser les nœuds avec des étiquettes connues et attribuer des étiquettes temporaires ou aléatoires aux autres nœuds.
2. À chaque itération, mettre à jour l'étiquette de chaque nœud non étiqueté pour correspondre à l'étiquette la plus fréquente parmi ses voisins.
3. Répéter le processus jusqu'à convergence ou pour un nombre prédéfini d'itérations.

Critères de convergence :

- L'algorithme converge lorsque les étiquettes de tous les nœuds restent inchangées entre deux itérations successives.
- Peut aussi être arrêté après un nombre fixe d'itérations pour éviter les calculs excessifs.

Applications de label propagation

Utilisations pratiques :

- **Classification de nœuds** : Assigner des étiquettes aux nœuds dans les réseaux sociaux, biologiques, ou d'information.
- **Détection de communautés** : Identifier les groupes naturels ou les communautés dans les graphes en se basant sur la similarité des étiquettes.
- **Analyse de réseaux** : Comprendre la structure et les dynamiques des réseaux complexes par la diffusion d'informations.

Avantages :

- Simple à implémenter et à exécuter, avec une faible complexité computationnelle.
- Efficace pour les grands graphes où peu de nœuds sont initialement étiquetés.

Réseaux de neurones graphiques (GNN)

Introduction aux Graph Convolutional Networks (GCN)

Definition

Les **Graph Convolutional Networks (GCN)** sont une classe de réseaux de neurones utilisés pour l'apprentissage sur graphes, combinant la structure des graphes avec des techniques de convolution.

Principes de Base :

- Intègrent à la fois les caractéristiques des nœuds et la structure topologique du graphe.
- Adaptent les méthodes de convolution, typiques des CNNs, au domaine des graphes.

Applications :

- Classification de nœuds et de graphes.
- Analyse de réseaux sociaux, bioinformatique, chimie computationnelle.

Architecture des GCN

Structure des couches :

- Les GCN sont généralement composés de plusieurs couches de convolution, suivies de couches de pooling et de couches entièrement connectées.
- Chaque couche convolutive capte des informations de voisinage à un degré plus élevé.

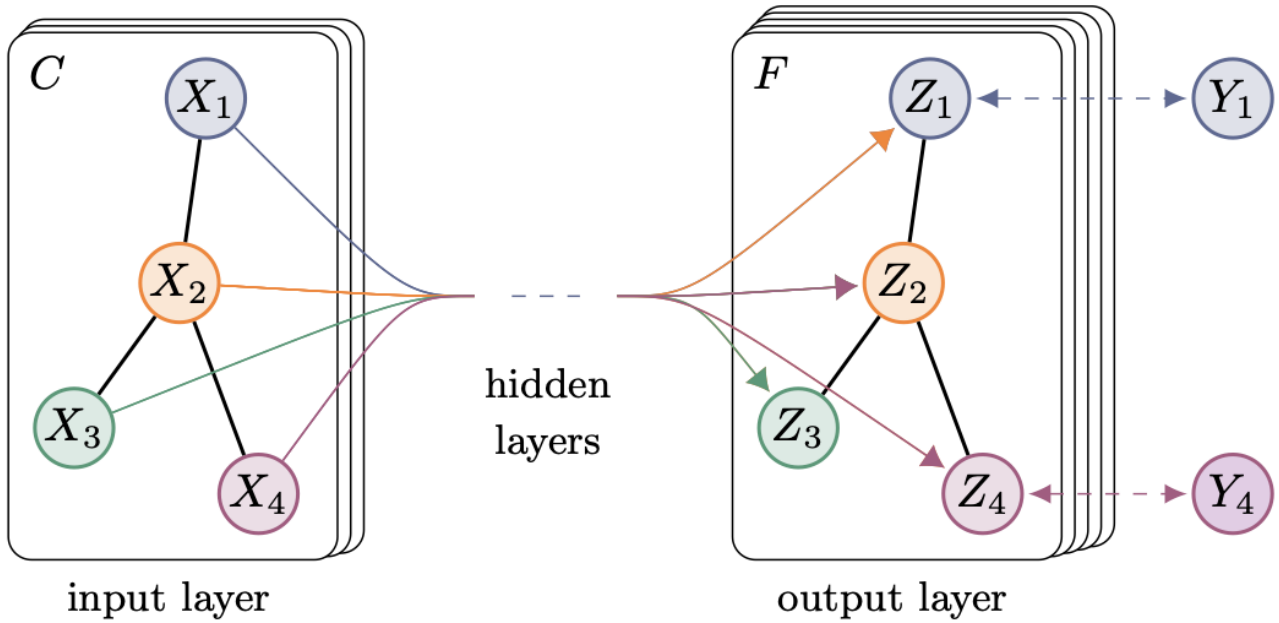


Figure: Architecture d'un Graph Convolutional Network

Fonctionnement des GCN

Mécanisme de convolution :

- Chaque couche d'un GCN applique une opération de convolution qui agit sur les nœuds et leurs voisins immédiats.
- Les caractéristiques des nœuds sont mises à jour en combinant leurs propres caractéristiques avec celles de leurs voisins.

Formule de convolution :

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

où $\tilde{A}$ est la matrice d'adjacence avec des boucles ajoutées, $\tilde{D}$ est le degré de $\tilde{A}$, $H^{(l)}$ est la représentation des nœuds à la couche l, et σ est une fonction d'activation.

Apprentissage des GCN

Processus d'apprentissage :

- Les GCNs sont entraînés en utilisant la rétropropagation, similaire aux CNNs traditionnels.
- Les poids des couches convolutives sont ajustés pour minimiser une fonction de perte définie, souvent une perte de classification croisée.

Optimisation :

- Des techniques comme la descente de gradient stochastique ou Adam sont utilisées pour l'optimisation.
- Les régularisations, telles que le dropout, peuvent être appliquées pour éviter le surajustement.

Défis et limitations des GCN

Défis:

- **Échelle** : La taille et la complexité des graphes peuvent rendre l'entraînement des GCNs coûteux en termes de calculs, surtout pour les grands réseaux.
- **Hétérogénéité** : Les graphes avec divers types de nœuds et d'arêtes peuvent nécessiter des adaptations spécifiques des modèles GCN.

Limitations :

- **Sur-smoothing** : Avec un trop grand nombre de couches, les caractéristiques des nœuds peuvent devenir indifférenciables, réduisant ainsi l'efficacité du modèle.
- **Indépendance spatiale** : Les GCNs se basent sur l'hypothèse que les nœuds proches sont similaires, ce qui n'est pas toujours le cas.
- **Manque de flexibilité** : Les architectures GCN standard peuvent ne pas être optimales pour tous les types de structures de graphes ou de tâches d'apprentissage.

Introduction aux Graph Attention Networks (GAT)

Definition

Les **Graph Attention Networks (GAT)** sont une forme de réseaux de neurones pour graphes qui utilisent des mécanismes d'attention pour pondérer les contributions des voisins d'un nœud.

Principes :

- Intègrent les informations des voisins d'un nœud en mettant l'accent sur les plus pertinents, grâce à l'attention.
- Permettent de gérer les graphes hétérogènes et de capter des relations complexes entre les nœuds.

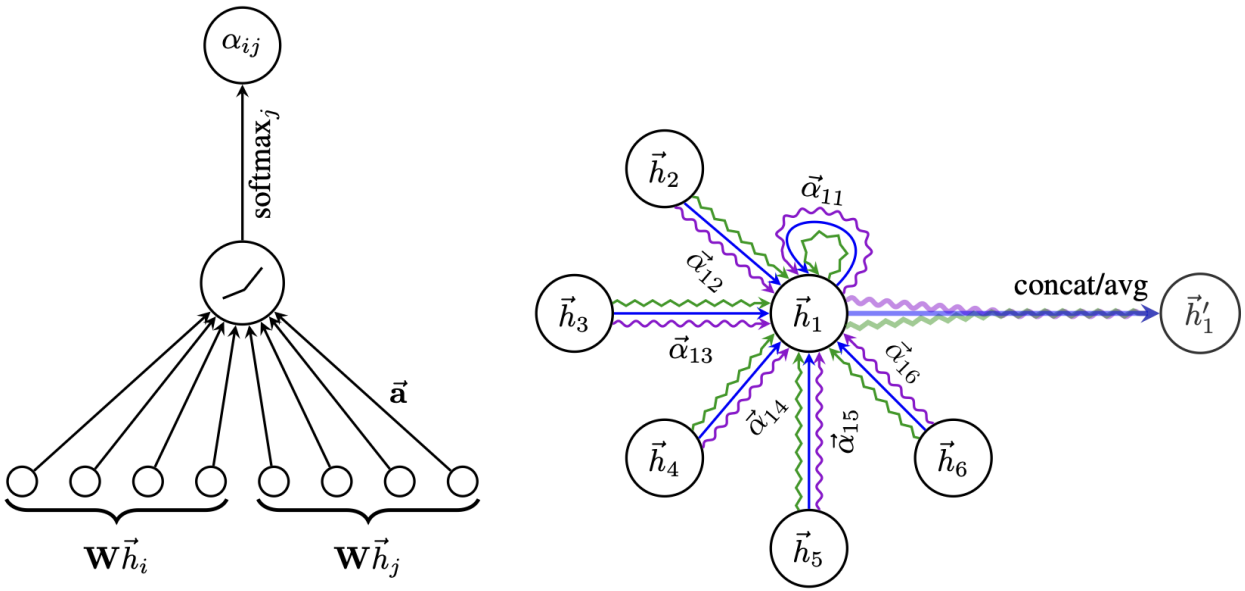
Applications :

- Classification de nœuds, analyse de réseaux sociaux, bioinformatique.
- Tâches nécessitant une compréhension fine des relations entre entités.

Architecture des GAT

Structure des couches :

- Les GAT sont composés de plusieurs couches d'attention, chacune mettant à jour les caractéristiques des nœuds en se basant sur leurs voisins.
- L'attention multi-tête peut être utilisée pour capturer divers aspects de la relation entre les nœuds.



Mécanisme d'Attention et mise à jour des états dans les GAT

Fonctionnement de l'Attention :

- Chaque nœud reçoit des informations de ses voisins avec un poids d'attention, calculé dynamiquement.
- L'attention permet de se concentrer sur les informations les plus pertinentes venant des voisins.

Calcul de l'Attention :

$$\alpha_{ij} = \frac{\exp(\sigma(a^T[Wh_i \| Wh_j]))}{\sum_{k \in N(i)} \exp(\sigma(a^T[Wh_i \| Wh_k]))}$$

où α_{ij} est le coefficient d'attention entre les nœuds i et j, h sont les caractéristiques des nœuds, W est une matrice de poids et a est un vecteur d'attention appris.

Mise à Jour des États des Nœuds :

- Les états des nœuds sont mis à jour en combinant les informations pondérées par l'attention de leurs voisins.
- $h'_i = \sigma(\sum_{j \in N(i)} \alpha_{ij} W h_j)$ où h'_i est le nouvel état du nœud i et σ est une fonction d'activation non linéaire.

Mécanisme de Multi-Head Attention dans les GAT

Mécanisme de Multi-Head Attention :

Dans la pratique, plusieurs plusieurs mécanismes d'attention (têtes) sont appliqués à chaque nœud, capturant ainsi divers aspects des informations.

Combinaison des résultats des têtes :

- **Concaténation** (couches cachées) :

$$h_i' = \Big\|_{k=1}^{K} \sigma\Big(\sum_{j \in N(i)} \alpha_{ij}^k W^k h_j \Big)$$

- **Moyenne** (couche de sortie) :

$$h_i' = \sigma\Big(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in N(i)} \alpha_{ij}^k W^k h_j\Big)$$

Avantages :

- Augmente la capacité du modèle à extraire et combiner diverses caractéristiques des nœuds et de leurs relations.
- Améliore la représentation globale des nœuds en intégrant plusieurs perspectives.

Apprentissage des GAT

Processus d'apprentissage :

- Les GAT sont entraînés en utilisant la rétropropagation, avec une attention ajustée pour optimiser la tâche spécifique.
- Les coefficients d'attention sont appris pour maximiser la performance sur la tâche de prédiction.

Optimisation :

- Des techniques d'optimisation comme la descente de gradient stochastique ou Adam sont employées.
- Le dropout et d'autres stratégies de régularisation peuvent être utilisés pour améliorer la généralisation.

Défis des GAT

Défis :

- Gestion de grands graphes : La complexité computationnelle peut augmenter avec la taille du graphe.
- **Interprétabilité** : Comprendre comment les mécanismes d'attention prennent leurs décisions reste un défi.
- **Hétérogénéité des Graphes** : Adapter les GAT pour fonctionner efficacement sur des graphes avec des types de nœuds et d'arêtes variés

Impact : Les GAT offrent une approche puissante et flexible pour l'apprentissage sur graphes, avec un potentiel significatif pour des avancées dans divers domaines.

Merci de votre attention

redha_moulla@yahoo.fr