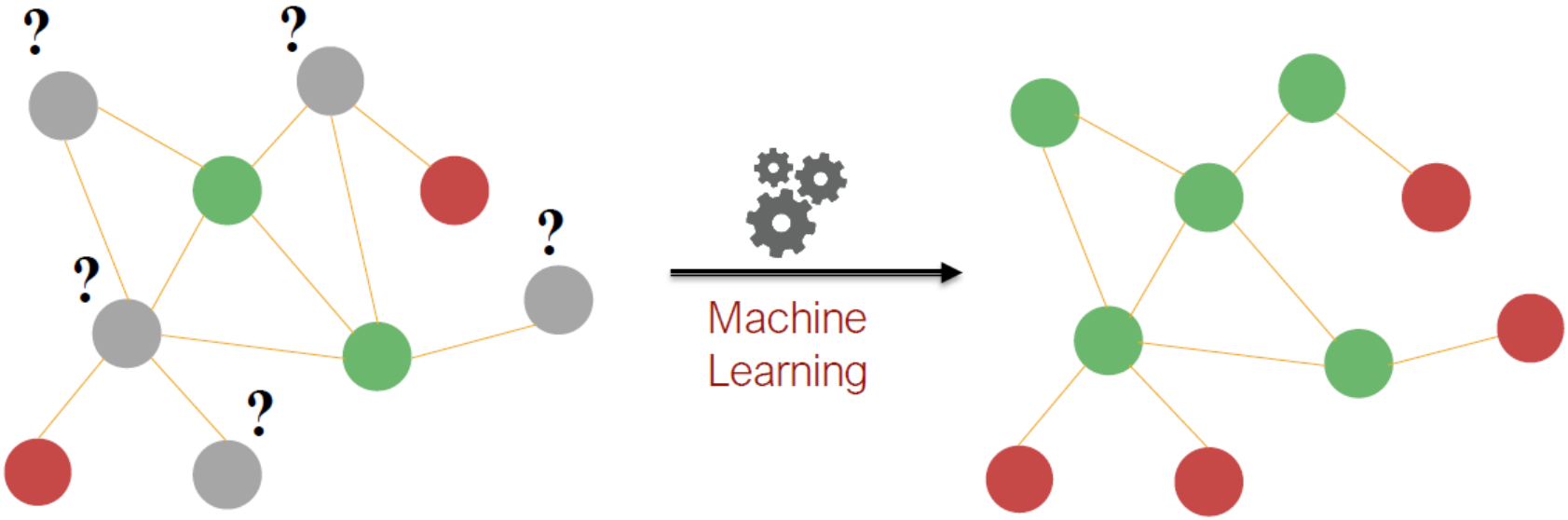
Apprentissage de représentation de graphes

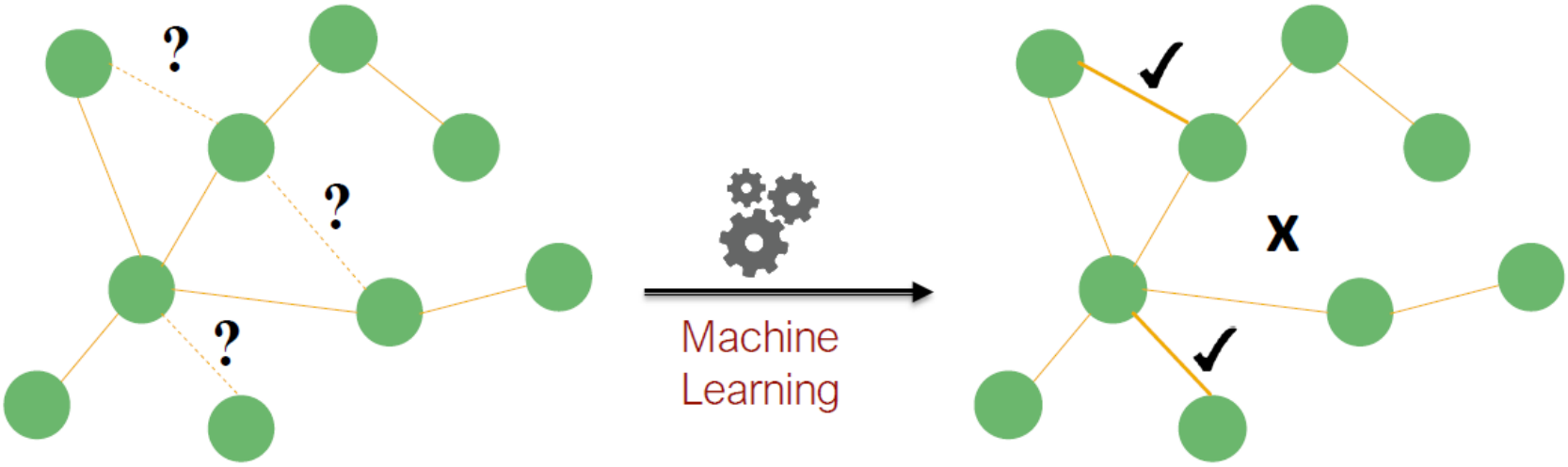
Daniel Aloise <daniel.aloise@polymtl.ca>
(basé sur les slides du professeur J. Leskovec, Stanford)

Example : classification collective



Source : Leskovec, 2021

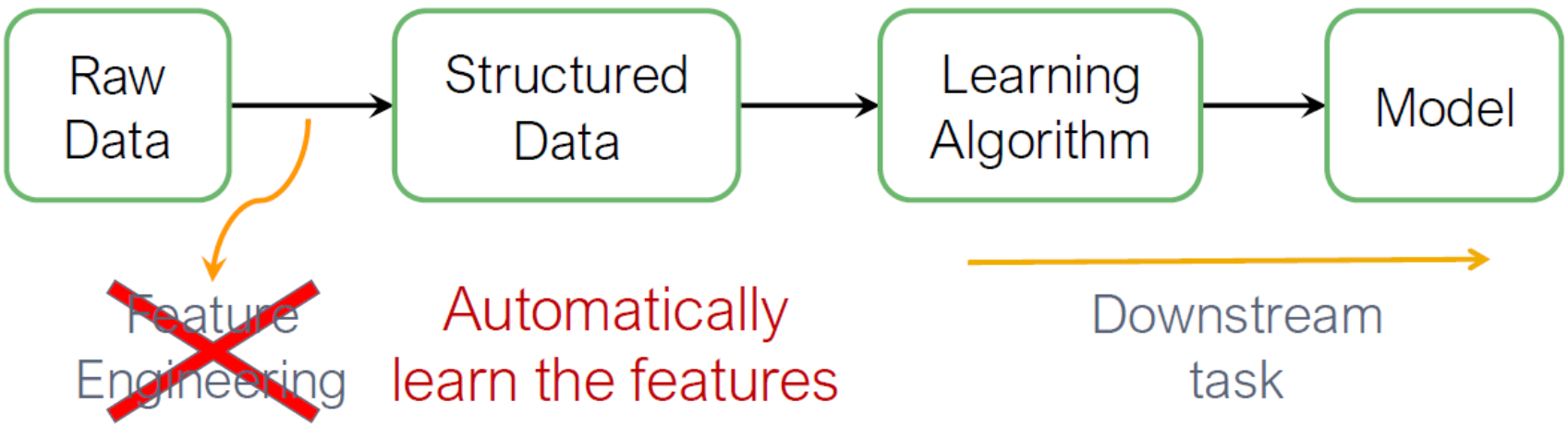
Example : prédiction des liens



Source : Leskovec, 2021

Génie d'attributs

- Le pipeline de la fouille des données nécessite que les données soient préparées d'abord
  - Extraction d'attributs : automatique vs. fait à la main.

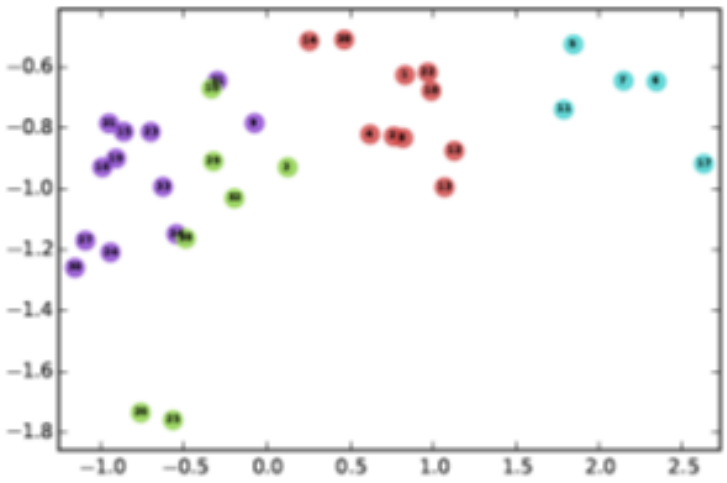


Source : Leskovec, 2021

Incorporation des sommets

embedding

représentation vectorielle dont les valeurs à l'intérieur représentent l'information du mot



Intuition : recherchez une incorporation des sommets dans un espace de dimension d de sorte que les sommets similaires dans le graphe soient proches les uns des autres.

Setup

V: ensemble de sommets
A: ensemble d'arrêtes

- Nous allons assumer un graphe $G = (V, A)$ où :
  - V est l'ensemble de sommets
  - A est une matrice d'adjacence (possiblement pondérée)

- Aucun attribut sur les sommets ou information supplémentaire n'est utilisée ! on n'a pas la topologie du graphe. ex: une personne on n'a pas son age, son poids, sa taille. On n'a pas considéré ces informations pour faire du embedding

Apprentissage de l'incorporation de sommets

Encodeur: Fonction qui prend un sommet U et qui va le transformer dans un vecteur Zu

1. Définition d'un encodeur (c'est-à-dire un *mapping* des sommets à l'espace d'incorporation)

Fonction de similarité: pouvoir comparer la similarité des sommets

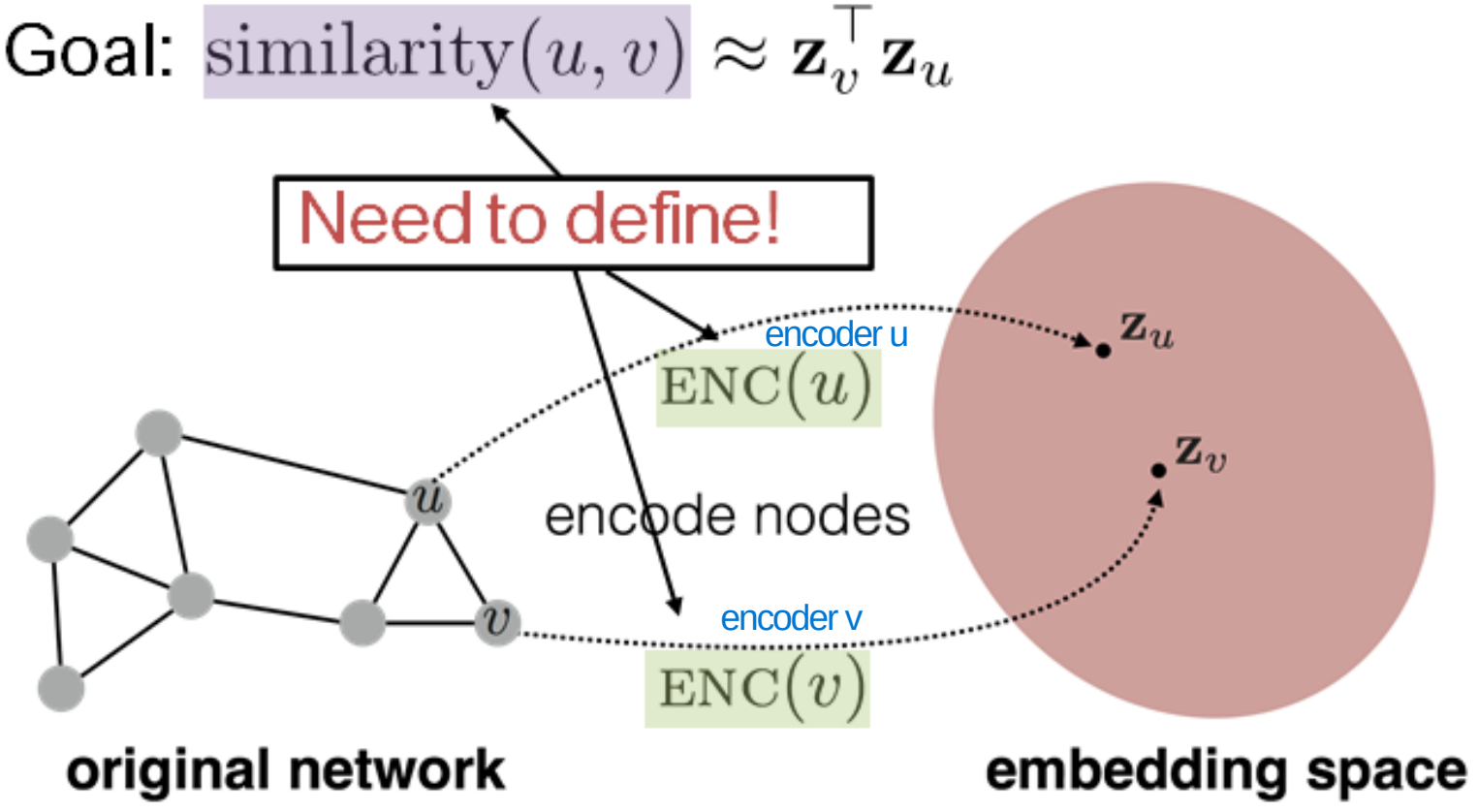
2. Définition d'une fonction de similarité entre les sommets (c'est-à-dire une mesure de similarité dans le graphe d'origine).

3. Optimisation des paramètres de l'encodeur pour que :

$$similarite(u, v) \approx z_v^T z_u$$

similarité des sommets u et v = produit scalaire des représentations est de 0.8 par exemple

Apprentissage de l'incorporation de sommets



Source : Leskovec, 2021

ex: si on veut que chaque sommet soit représenté par 10 valeurs numériques, on aura une matrice qui aura 10 lignes et qui a autant de colonnes qu'il y a de noeuds dans notre graphe. Pour chaque colonne, la 1ere colonne est la représentation du sommet sommet 1, la 2e colonne représente le 2e sommet etc...

Codage simple

Z: matrice qui contient les valeurs réelles

$$ENC(v) = Zv$$

$Z \in \mathbb{R}^{d \times |V|}$

d: nb de valeurs pour représenter les sommets

matrice où chaque colonne correspond à l'incorporation d'un sommet dans l'espace $\mathbb{R}^d$.

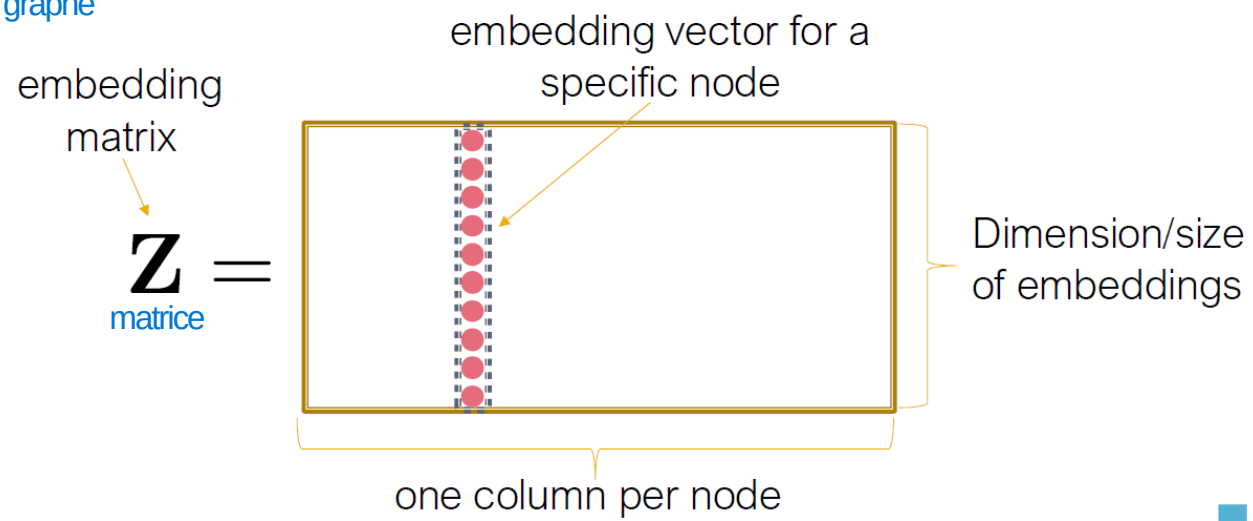
[ce qui nous apprenons]

$v \in \mathbb{I}^{|V|}$

vecteur qui fait la taille de notre vocabulaire, donc le nb de sommets dans notre graphe

vecteur où la seule position non-nulle correspond à l'indice du sommet v.

v vecteur



Codage simple

- Bref, chaque sommet se voit attribuer un vecteur d'incorporation unique.
  - plusieurs méthodes : node2vec, DeepWalk, LINE, etc.
- La distinction clé entre les méthodes de codage simple est la façon dont elles définissent la similarité entre les sommets dans le graphe d'origine.
- Deux sommets devraient-ils avoir des incorporations similaires s'ils :
  - sont connectés ?
  - ont des voisins communs ?
  - ont des « rôles structurels » similaires ? . . . si les sommets contrôlent des informations

Similarité basée sur la matrice d'adjacence

Problème: Approche locale, on considère uniquement les voisins directs. Si un noeud est lié, alors ils sont similaires. Si un noeud n'est pas lié ils sont différents, peu importe de la distance.

- La fonction de similarité est juste le poids de l'arête entre les sommets u et v dans le graphe d'origine.
- Intuition : les produits scalaires entre les incorporations de sommets se rapprochent de l'existence des arêtes.

$$\mathcal{L} = \sum_{(u,v)\in V\times V} \|\mathbf{z}_u^\top \mathbf{z}_v - \mathbf{A}_{u,v}\|^2$$

Loss (l'erreur)

perte (ce qui on veut minimiser)

Similarité dans l'espace d'incorporation

Matrice d'adjacence du graphe

les arretes (arc) dans notres graphe

si les arretes sont reliés, les vecteurs sont similaires, sinon ils ne sont pas similaires

Similarité basée sur la matrice d'adjacence

$$\mathcal{L} = \sum_{(u,v)\in V\times V} \|z_u^T z_v - A_{u,v}\|^2$$

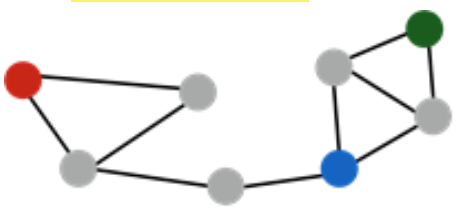
- L'objective est d'obtenir la matrice d'incorporation $Z \in \mathbb{R}^{d\times|V|}$ que minimize la perte $\mathcal{L}$.
  - Option 1 : Utiliser la méthode de descente de gradient stochastique.
  - Option 2 : Utiliser des méthodes de décomposition de matrices (e.g. SVD).

Problème: On considère uniquement les voisins directs!

Similarité basée sur la matrice d'adjacence

$$\mathcal{L} = \sum_{(u,v)\in V\times V} \|z_u^T z_v - A_{u,v}\|^2$$

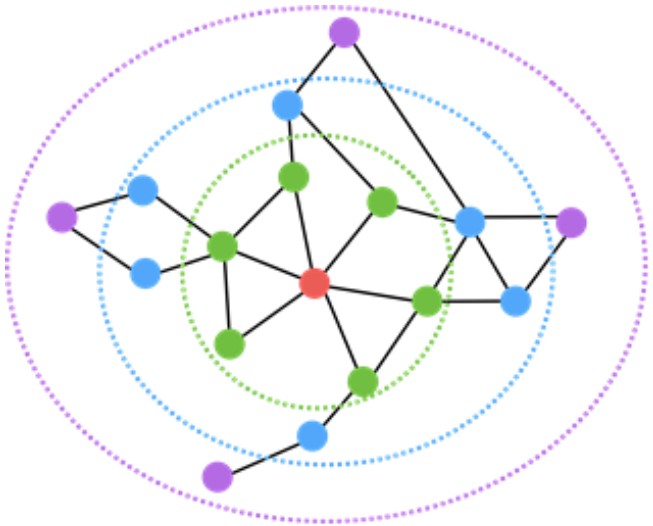
- Inconvénients
  - Temps $O(|V|^2)$ (considère tous les pairs de sommets)
    - Nous pouvons optimiser à $O(|E|)$ - réf. sur Moodle
  - L'optimisation considère uniquement des connections locales et directes.



e.g. le sommet bleu est plus similaire au sommet vert qu'au sommet rouge, bien qu'ils ne soient pas directement connectés.

Similarité *multi-hop*

- Le dernier inconvénient peut être contourné partiellement en utilisant de similarités multi-hops



- Rouge: sommet cible
- Vert: voisinage 1-hop
  - $\mathbf{A}$ (i.e., matrice d'adjacence)
- Bleu: voisinage 2-hop
  - $\mathbf{A^2}$
- Mauve: voisinage 3-hop
  - $\mathbf{A^3}$

sommets situés à une puissance 2, peut etre 3 ou 4

- Idée : $\min \mathcal{L} = \sum_{(u,v) \in V \times V} \|z_u^T z_v - A_{u,v}^k\|^2$, où k est le niveau de connectivité considéré.

Incorporation par marches aléatoires

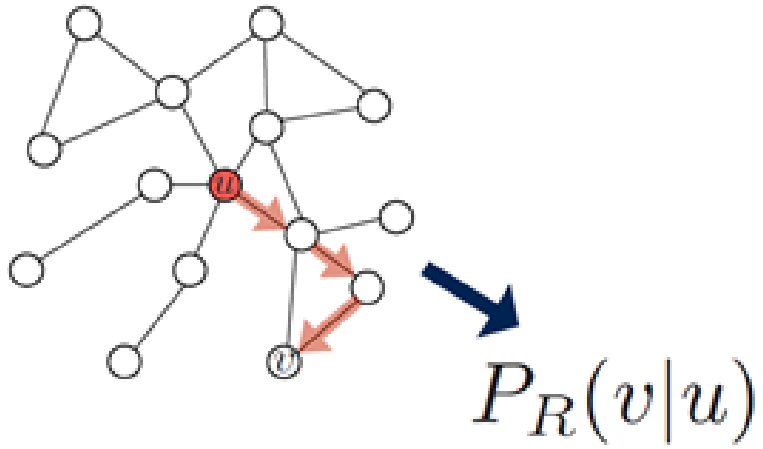
$$\mathbf{z}_u^\top \mathbf{z}_v \approx$$

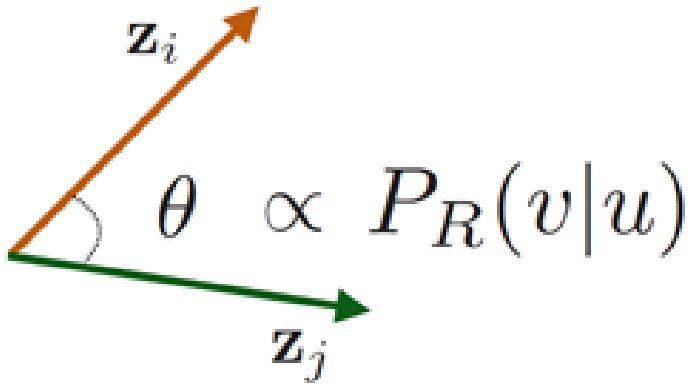
Probabilité que u et v co-occurrent dans une marche aléatoire sur le graphe.

aparaissent en mm temps

Incorporation par marches aléatoires

1. Estimez la probabilité de visiter le sommet v lors d'une marche aléatoire à partir du sommet u en utilisant une stratégie de marche aléatoire R.
2. Optimisez l'incorporation pour encoder ces statistiques de marche aléatoire.





angle des 2 embedding doit etre proportionnel à la proba de voir le sommet v sachant qu'on connait le sommet u

Pourquoi des marches aléatoires ?

AVANTAGES

Expressivité : définition stochastique flexible de la similarité entre les sommets qui intègre à la fois des informations de voisinage locales et d'ordre supérieur.

efficace

Efficience efficace car : pas besoin de prendre en compte toutes les paires de sommets lors de l'entraînement ; il suffit de considérer les paires qui coexistent lors de marches aléatoires.

Optimisation de marches aléatoires

1. Faite de courtes marches aléatoires à partir de chaque sommet du graphe en utilisant une stratégie R.
2. Pour chaque sommet u collectez $N_R(u)$, c.-à.d. l'ensemble* de sommets visités par des marches aléatoires débutant au sommet u.
3. Maximisez :

somme pr chacun des sommets de départ

$$\sum_{u \in V} \sum_{v \in N_R(u)} log(P(v|z_u)) \Rightarrow \text{objectif de maximum likelihood}$$

proba d'avoir v sachant Zu

Zu: embedding du sommet u, le sommet de départ

ressemblence maximale

où $P(v|z_u)$ est noté comme la probabilité (prédite) de visiter le sommet v sur des marches aléatoires à partir du sommet u.

$*N_R(u)$ peut contenir des sommets répétés

Optimisation de marches aléatoires

- De façon équivalente, on peut minimiser :

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -log(P(v|z_u))$$

- Ensuite, on paramètre $P(v|z_u)$ avec softmax.

$$P(v|z_u) = \frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)}$$

Proba d'obtenir le sommet v à partir du sommet u

Pourquoi softmax ?

Nous voulons que le sommet v soit le plus similaire au sommet u parmi tous les sommets du graphe.
Intuition : $\sum_i exp(x_i) \approx \max_i exp(x_i)$

Optimisation des marches aléatoires

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log\left(\frac{\exp(\mathbf{z}_u^\top \mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^\top \mathbf{z}_n)}\right)$$

somme pour tous les sommets u

Somme pour les sommets v visités à partir de u

Probabilité prédite pour que u et v co-occurrent ensemble dans une marche aléatoire

L'optimization de l'incorporation par marches aléatoires = trouver des incorporations z_u qui minimisent $\mathcal{L}$.

Optimisation des marches aléatoires

Pourtant faire cela naïvement coûte trop cher ! !

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log\left(\frac{\exp(\mathbf{z}_u^\top \mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^\top \mathbf{z}_n)}\right)$$

Les deux sommes imbriquées entraîne une complexité de $O(|V|^2)$

Le terme de normalisation du softmax est le coupable. . . peut-on l'approximer ?

Optimisation des marches aléatoires

- Solution : *échantillonage négatif*

$$\frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)}$$

$$\approx \log(\sigma(z_u^T z_v)) - \sum_{i=1}^{k} \log(\sigma(z_u^T z_{\overline{n}_i}))$$

- $\sigma(\cdot)$ est la fonction sigmoïde - rend chaque terme entre 0 et 1.
- L'ensemble $\overline{n}$ est sélectionné à partir des échantillons négatifs = sommets pas atteints par les marches aléatoires.
- Au lieu de normaliser par rapport à tous les sommets, on normalise juste par rapport à k échantillons négatifs $\Rightarrow$ plus rapide

Optimisation des marches aléatoires

- Solution : *échantillonage négatif*

$$\frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)}$$

$$\approx \log(\sigma(z_u^T z_v)) - \sum_{i=1}^{k} \log(\sigma(z_u^T z_{\bar{n}_i}))$$

- Les échantillons négatifs sont sélectionnés selon une probabilité proportionelle à leur dégrées.
- Deux considérations par rapport à k (# d'échantillons négatifs) :
  - Un k élevé donne des estimations plus robustes.
  - Un k élevé correspond aussi à un biais plus élevé sur les événements négatifs
  - En pratique, on utilise k entre 5 et 20 (très petite pour des graphes massives)

En résumé

1. Faite de courtes marches aléatoires à partir de chaque sommet du graphe en utilisant une stratégie R.
2. Pour chaque sommet u collectez $N_R(u)$, c.-à.d. l'ensemble de sommets visités par des marches aléatoires débutant au sommet u.
3. Maximisez :

$$\sum_{u \in V} \sum_{v \in N_R(u)} log(P(v|z_u)) \Rightarrow \text{objectif de maximum likelihood}$$

Pour cela, vous pouvez utiliser une descente de gradient stochastique.

Descente de gradient stochastique

Au lieu d'évaluer les gradients sur tous les sommets, évaluez-les pour chaque sommet individuellement.

- Initialisez z_u avec des valeurs aléatoires pour tous les sommets $u \in V$ chacune des colonnes initialisés aléatoirement
- Itérer jusqu'à la convergence : $\mathcal{L}^u = \sum_{v \in N_r(u)} -\log(P(v|z_u))$
  - Échantillonnez un sommet u, ensuite pour tout v calculez le dérivé $\frac{\partial \mathcal{L}^u}{\partial z_v}$
  - Pour tout v, calculez $z_v \leftarrow z_v - \eta \frac{\partial \mathcal{L}^u}{\partial z_v}$

Comment devrait-on marcher aléatoirement ?

- Jusqu'à présent, nous avons décrit comment optimiser des incorporations des graphes étant donné une stratégie de marche aléatoire R.
- Quelles stratégies pouvons-nous utiliser pour exécuter ces marches aléatoires ?
  - Idée la plus simple : **Juste faire des marches aléatoires non-biasées de taille fixe à partir de chaque sommet** (i.e., DeepWalk - Perozzi et al., 2013.)
  - node2vec marches aléatoires biasées qui arbitrent entre une recherche en profondeur (vision globale) et une recherche en largeur (vision locale) - Grover and Leskovec, 2016.
  - D'autres possibilités existent encore.

Comment pouvons-nous utiliser les incorporations ?

Zi: représentation vectorielle des points

Clustering : regroupement des sommets/points basés sur z_i

Classification des sommets : prédiction de l'étiquette du sommet i basés sur z_i

Prédiction de liens : prédiction de l'arête (i, j)

- Nous pouvons dans ce cas enchaîner, sommer, multiplier ou prendre la différence entre les incorporations.

Fouille des graphes : classification des graphes

- Nous pouvons sommer ou calculer la moyenne des incorporations de tous les sommets du graphe.

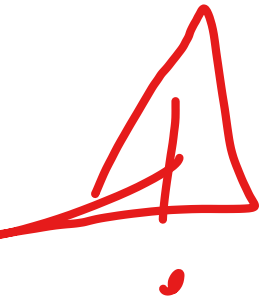
Conclusion

- Idée de base : Incorporer des sommets de sorte que les distances dans l'espace d'incorporation reflètent les similarités entre les sommets dans le graphe d'origine.

- Différentes notions de similarités :
  - basées sur l'adjacence (c.-à.-d similaires si connectés).
  - définition multi-hop.
  - basées sur des marches aléatoires.

Conclusion

- Alors, quelle méthode devrait-on utiliser ?
- Aucune méthode ne l'emporte dans tous les cas
  - node2vec marche mieux pour la classification tandis que l'approche multi-hop marche mieux pour la prédiction de liens.
- Les approches par marches aléatoires sont normalement plus rapides.
- Problèmes en commun :
  - Les méthodes de codage simple ne partagent aucun paramètre entre les sommets.
  - Ils n'exploitent pas les attributs des sommets.
  - Possible solution : Graph Neural Networks (GNNs).



peu importe la facon on représente un sommet, ca n'impacte pas la facon dont on représente les autres sommets, donc limitation

Si 1 sommet avait des attributs, on a des caractéristiques sur cet individu et on ne le prend pas en compte, donc limitation