Comment estimer la distance d'édition de graphe avec le problème du transport optimal ?

Praden Pierre

April 12, 2024

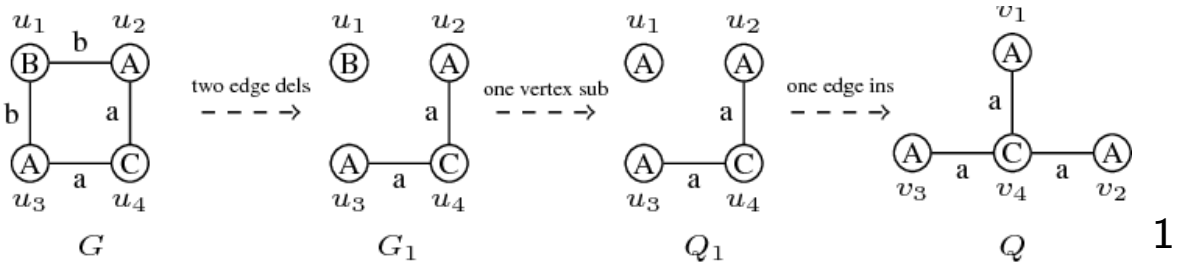
Sommaire

Qu'est ce que la GED ?

Graph Edit Distance(GED) $\rightarrow$ Distance entre 2 graphes
Chemin d'édition le plus court d'un graphe g_1 à un graphe g_2

Pour cela, plusieurs opérations sont possibles :

- Addition/suppression de noeud
- Addition/suppression d'arête
- Edition de noeud
- Edition d'arête



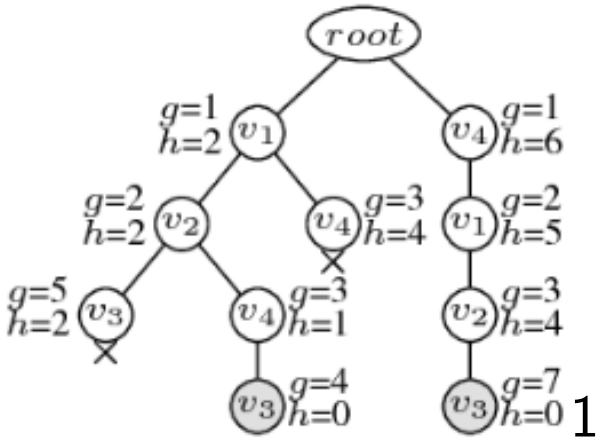
GED(G, Q) = 4

[1] An efficient algorithm for graph edit distance computation, Knowledge-Based Systems (2019)

Calcul exacte de la GED

Meilleur algorithme : A*.
Rappel de A*:

- Un arbre de possibilités Γ
- Une fonction de calcul de la distance déja parcouru g
- Une heuristique de la distance minimale à parcourir h
- On développe le noeud n qui minimise g(n) + h(n) jusqu'à obtenir une solution.



1

Determiner la ged, un probleme d'assignation

On peut représenter la solution trouvé precedemment par :

$$\Pi^* = \begin{pmatrix} & v1 & v2 & v3 & v4 \\ u1 & 1 & 0 & 0 & 0 \\ u2 & 0 & 1 & 0 & 0 \\ u3 & 0 & 0 & 1 & 0 \\ u4 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Et retrouver la GED correspondante avec :

$$\underbrace{\sum_{i,j}^{n} Cn_{ij}\Pi^*_{ij}}_{\text{Cout d'édition des noeuds}} + \underbrace{\sum_{i,j,k,l}^{n} \Pi^*_{ij} Ce_{ijkl}\Pi^*_{kl}}_{\text{Cout d'édition des arêtes}}$$

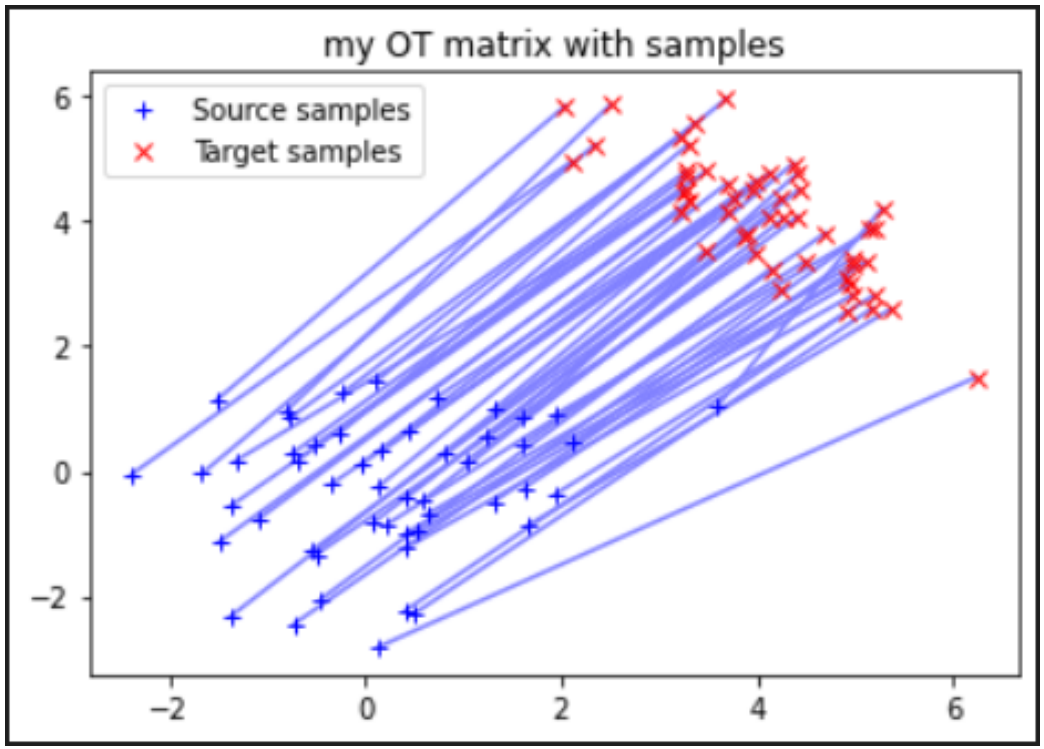
C_n étant la matrice de cout d'associer le noeud v_i au noeud u_j et
C_e le tenseur qui représente le coût d'associer les arêtes $v_i \rightarrow v_j$ et $u_k \rightarrow u_l$

Qu'est ce que le problème de transport?

Problème de transport optimal : On souhaite déplacer un ensemble de point X vers un autre ensemble de point Y et on cherche la fonction qui minimise la distance totale.

$$\min_{\Pi} \sum_{i,j} C_{i,j} \Pi_{i,j}$$

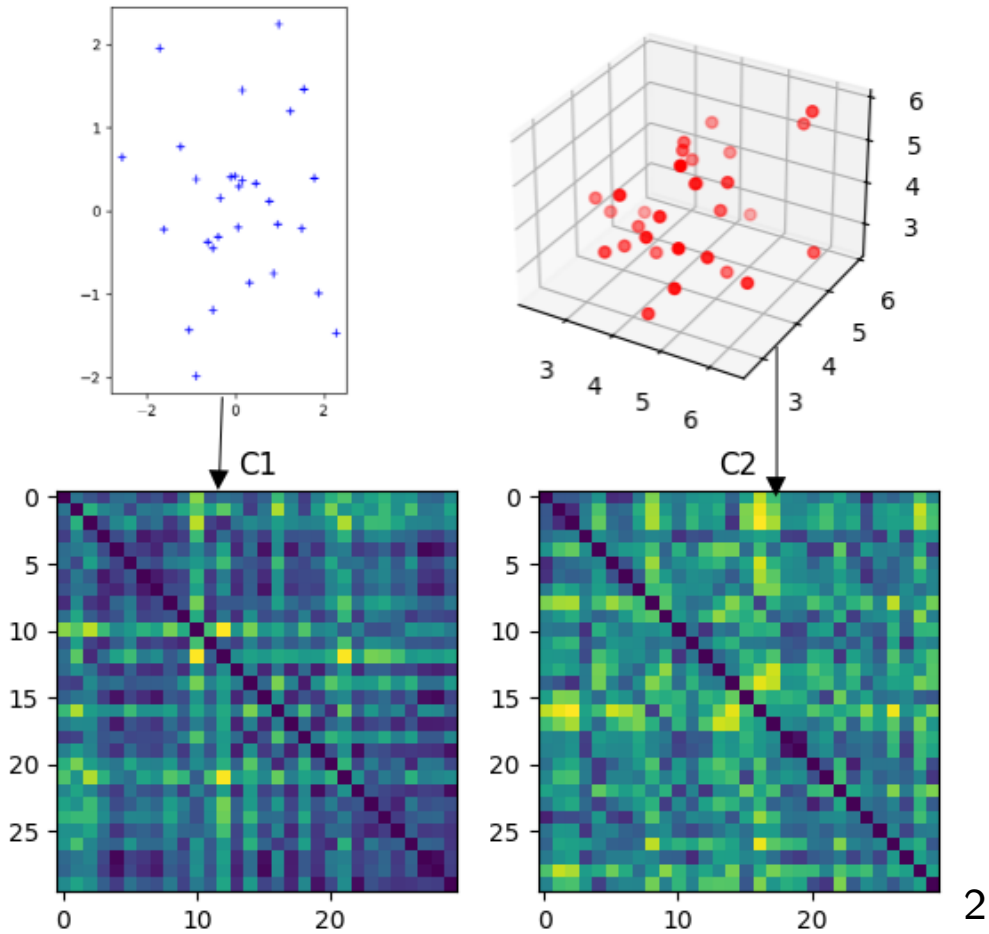
avec $\Pi_{i,j} = 1$ si X[i] est associé à Y[j] sinon $\Pi_{i,j} = 0$



Distance de Gromov-Wasserstein

On minimise cette fois-ci $\min_{\Pi} \sum_{i,j,k,l}^{n} \Pi^*_{ij} Ce_{ijkl} \Pi^*_{kl}$

Avec Ce_{ijkl} la différence des distances de $u_i \to u_k$ et $v_j \to v_l$



[2]

[2] https://pythonot.github.io/auto_examples/gromov/plot_gromov.html

Pourquoi utiliser le transport?

Matrice bistochastique qui peut être optimisé par descente de gradient.

$$\Pi = \begin{pmatrix} & v1 & v2 & v3 & v4 \\ u1 & 0.8 & 0.2 & 0 & 0 \\ u2 & 0.2 & 0.8 & 0 & 0 \\ u3 & 0 & 0 & 0.5 & 0.5 \\ u4 & 0 & 0 & 0.5 & 0.5 \end{pmatrix}$$

Note : Restriction à des graphes de même taille
$\rightarrow$ Répartition des GED différentes (Surtout pour IMDB)

Le but: Trouver une représentation du graphe telle que le problème de transport donne la même matrice de permutation que A*

Les représentations

3 représentations :

- Degrés de chaque noeud : G $\rightarrow$ [2 2 2 2], Q $\rightarrow$ [1 1 1 3]
- Vecteur des degrés d'ordre 1 à n:

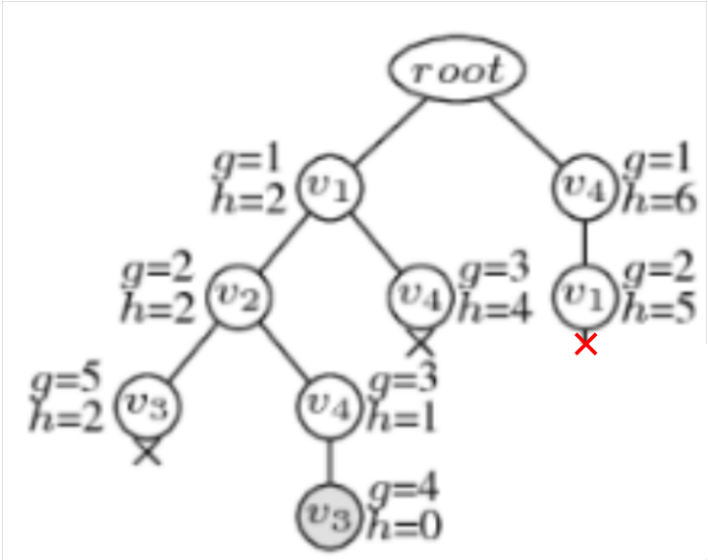
$$\text{G} \rightarrow \begin{pmatrix} degré & 1 & 2 \\ u1 & 2 & 3 \\ u2 & 2 & 3 \\ u3 & 2 & 3 \\ u4 & 2 & 3 \end{pmatrix} \quad \text{Q} \rightarrow \begin{pmatrix} degré & 1 & 2 \\ u1 & 1 & 3 \\ u2 & 1 & 3 \\ u3 & 1 & 3 \\ u4 & 3 & 3 \end{pmatrix}$$

- Matrice de distance de dijkstra (Gromov-Wasserstein)

Estimation de la GED sur IMDB

Calcul exact $\rightarrow$ complexité $O(n!)$

- a* avec beamsearch

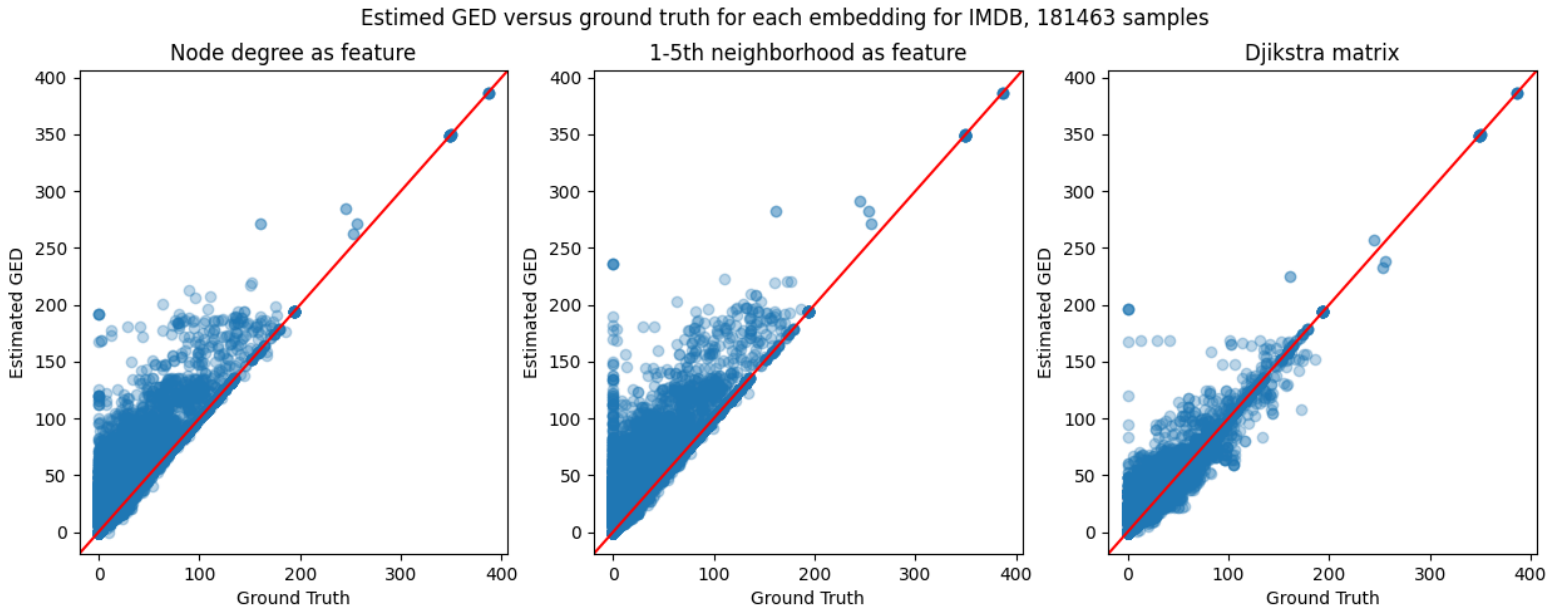


1

- Volgenant and Jonker [3]

[3] SimGNN: A Neural Network Approach to Fast Graph Similarity Computation (2020)

Résultats sur IMDB



Mean Squared Error

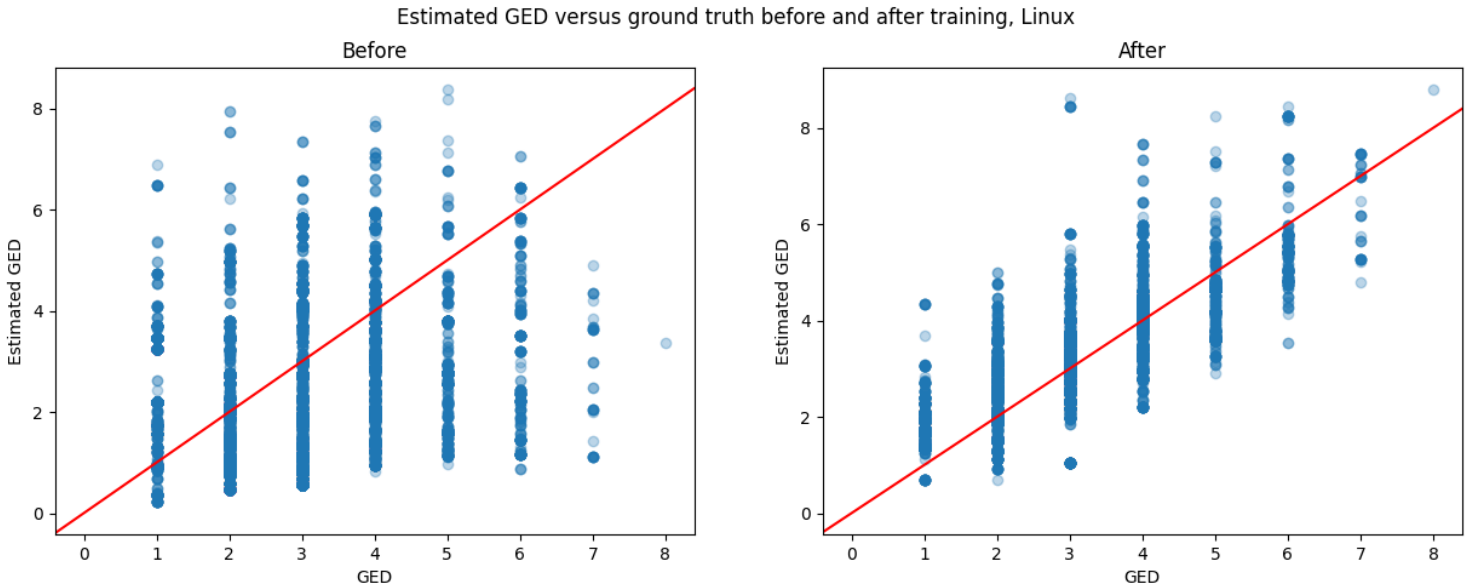
MSE	Degré	Degré successif	dijkstra
Linux	64.791491	64.92949	46.77884
AIDS	110.643564	119.437623	119.066117
IMDB	72.371183	81.769154	16.704606

Mean Absolute Error

MAE	Degré	Degré successif	dijkstra
Linux	7.275048	7.048691	5.572168
AIDS	9.989424	10.37421	10.31246
IMDB	2.397525	2.569196	0.876972

MLP sur les degrés succesifs

On pondère les poids du vecteur de caractéristique pour obtenir des meilleurs prédictions.



Mean Squared Error

MSE	Linux	AIDS	IMDB
MLP	0.389862	5.083177	37.316564

Améliorations possibles

- Etendre les réprésentations précedantes à des graphes de tailles différents
  $\rightarrow$Transport partiel [4]
- Prendre en compte l'intégralité du graphe pour le calcul du vecteur de caractéristique de chaque noeud
  $\rightarrow$GCN, Transformer

[4] Partial Optimal Transport with Applications on Positive-Unlabeled Learning (2020)

Références

1 An efficient algorithm for graph edit distance computation, Knowledge-Based Systems (2019)

2 https://pythonot.github.io/auto examples/gromov/plot_gromov.html

3 SimGNN: A Neural Network Approach to Fast Graph Similarity Computation (2020)

4 Partial Optimal Transport with Applications on Positive-Unlabeled Learning (2020)