A Regularized Wasserstein Framework for Graph Kernels

Grégoire Béchade

17/01/2025

1 Abstract

2 Introduction

The authors of this paper aim to develop a Wasserstein distance between graphs, in order to perform graph classification.

3 Theoretical Background

The objective of the paper is to be able to define a optimal transport plan between two graphs. But, optimal transport enables to have access to a mapping between probability distrivutions. The first issue is to transform a graph into a probability distribution.

Notations :
Let $G = (V, E)$ a non-oriented graph with V the set of vertices and E the set of edges. We note n the number of vertices and m the number of edges. We consider $\xi_f : V \to \mathbb{R}^m$ a feature embedding function (TODO ADD REF), and $\xi_s : V \to \mathbb{R}^k$ a structure embedding function TODO ADD REF.
We can therefore associate to each vertex of the graph a single point in $\mathbb{R}^m \times \mathbb{R}^k$, and therefore a probability distribution on $\mathbb{R}^m \times \mathbb{R}^k$: $p = \sum_{i=0}^{n} \mu_i \delta(\xi_f(v_i), \xi_s(v_i))$, with μ_i the weight of the vertex v_i (set to $\frac{1}{n}$ if none is provided).

We are now equipped with a function that takes as an input a graph and outputs a probability distribution. We can easily define a distance between those two graphs, by computing the Wasserstein distance between the two associated probability distributions and solve the Kantorovich problem. However, the authors introduce two regularization terms and a clever cost matrix in the Wasserstein distance to take into account feature and structure similarity between graphs.

Problem formulation :
Let G_1 and G_2 two graphs, and $\mu 1$ and ν their associated probability distributions. The authors define the RW discrepancy between of these two graphs as :

$$RW(\mu, \nu) = \min_{\gamma \in \pi(\mu,\nu)} \left\langle \gamma, C^V \right\rangle_F + \beta_1 LW(\mu, \nu) + \beta_1 GW(\mu, \nu)$$

We will explain the different terms of this equation in the following sections.

3.1 C^V : a cost function to reflect feature similarity

We consider a graph $G = (V, E)$ and a graph signal $(x_i)_{i \in \{1,\dots,|V|\}} \in \mathbb{R}^m$, which is just a function on the vertices of the graph. This signal can for instance be the value of a physical

quantity in a graph of sensors, or the number of followers of a user in a social network. TODO ADD REF TO MLTS. We can therefore associate to a graph a signal matrix $X \in \mathbb{R}^{n \times m}$ (with n the number of nodes and m the space in which those features leave).

The local variation matrix of G is defined as : $\Delta(X) = |X - \frac{1}{\lambda_{max}(L)} L^j X|$, with L the graph Laplacian and $\lambda_{max}(L)$ its largest eigenvalue. The intuition behind this matrix is to measure the variation of the signal between two nodes of the graph. Considering a node v_i, its feature $x_i \in \mathbb{R}^m$ and the corresponding local variation $\Delta(x_i) \in \mathbb{R}^m$, the authors define the feature embedding function as $\xi_f(v_i) = [x_i, \Delta(x_i)] \in \mathbb{R}^{2m}$.

Finally, the feature similarity matrix, that will later be used as a cost matrix in the Wasserstein distance, is defined as matrix in $\mathbb{R}^{n_1 \times n_2}$: $C^V(i,j) = d_f(a_i, a_j)$, with d_f a distance function, which will be the L_2 distance in the paper.

This cost function means that it costs many to match nodes with different features, in terms of values but also of variations. The intuition behind this is that we want to take into account the behaviour of the graph signal when assessing the difference between two graphs.

3.2 LW : a regularization term to preserve neighbourhood similarity

The first regularization term is the Local Barycentric Wasserstein Distance (LW). This term enables to neighbourghood similarity, by solving once again a Wasserstein distance problem, but with a different cost matrix. The cost matrix of this distance is $C^N(i,j) = (d_s(e_i, e_j))_{i,j} \in \mathbb{R}^{n_1 \times n_2}$, with e_i and e_j the node embeddings of the nodes i and j from the two graphs, and d_s a distance function, which will be set as the hammer distance. However, the hammer distance is not defined for continuous variables, and the authors did not explai how they managed to compute it. A extension of hammer distance to continuous variables was found in Labib, Uznanski et Wolleb-Graf 2019, and might be the adaptation used ? This cost matrix reflects the idea that this term preserves the similarity between connected nodes in the transport plan : it costs lots to associate a vertex from G_1 to a vertex of G_2 if their embeddings are far away.
This cost matrix enables to associate nodes only of their embeddings are close, which seems intuitive : two similar nodes whould be associated together with a small cost. However, nothing enables the user to be sure that two connected nodes in G_1 will end up close in G_2, after the transportation. That is why the authors introduced a regularization term to enforce this property.

Formally, let us define $\hat{e}_i^{\mu} = \frac{\sum_{j=1}^{n_2} \gamma(i,j) e_j^{\nu}}{\sum_{j=1}^{n_2} \gamma(i,j)}$. This vector is the barycenter of the embeddings of the nodes of G_2 that are connected to the node i of G_1. Then the source regularization term is defined as :
This source regularization term is minimized when two connected nodes of G_1 are sent by the transport plans to two groups of nodes whose barycenters are close.
The target regularization term can be defined in the same way, as : $\Omega_{\nu}(\gamma) = \frac{1}{n_2^2} \sum_{i,j} b_{i,j} \left\| \hat{e}_i^{\nu} - \hat{e}_j^{\nu} \right\|^2$, with $b_{i,j}$ the adjacency matrix of G_2.

We can now define the regularization term of the LW distance between to graphs as :
$\Theta_{\omega}(\gamma) = \lambda_{\mu} \Omega_{\mu}(\gamma) + \lambda_{\nu} \Omega_{\nu}(\gamma) + \frac{\rho}{2} \|\gamma\|_F^2$

Finally, we have :

$$LW(\mu, \nu) = \min_{\gamma \in \pi(\mu,\nu)} \langle \gamma, C^N \rangle_F + \Theta_{\omega}(\gamma)$$

This distance preserves a proximity between nodes connectes by the transport plan and between nodes connected in the graphs.

A result from the paper is that this distance is strongly convex and smooth with respect to γ, which provides guarantees for the convergence of the algorithm.

3.3 GW : a regularization term to preserve pairwise similarity

A last term is introduced to preserve the pairwise similarity between matrixes. As done above, a cost matrix C^P is defined as : $C^P(i,j) = d_s(e_i, e_j)$, with d_s a distance function and e_i, e_j the node embeddings of the nodes i and j of the considered graph. A C^P matrix is defined for each pair of graphs, and the pairwise similarity between G_1 and G_2 is defined as the following 4-dimensional tensor (I noted taht this approach was also found in TITOUAN et al. 2019) :

$L_2(C_1^P(i,j), C_2^P(k,l)) = \left|C_1^P(i,j) - C_2^P(k,l)\right|^2$.

The Gromov-Wasserstein distance is then defined as :

$GW(\mu,\nu) = \min_{\gamma \in \pi(\mu,\nu)} \left\langle \gamma, L_2(C_1^P, C_2^P) \otimes \gamma \right\rangle_F - \lambda_g \Theta_g(\gamma)$, with $\Theta_g(\gamma)$ the Kulback Liebler divergence between γ and a prior distribution named γ' in the paper.

The authors note that the Gromov Wasserstein distance is not convex, but the introduction of the regularization term enables to have better convergence properties.

Finally, we have explained all the terms of the RW discrepancy, which computes an optimal transport plan between two "clever" probability distributions representing the graphs, while preserving neighbourghood similarity (see 3.2), pairwise similarity (see 3.3) and feature similarity (see 3.1). However, solving the problem is very hard, and the authors propose an algorithm to solve it, which is detailed in the next section.

3.4 An algorithm to solve the problem

In the paper, an algorithm named Sinkhorn Conditionnal Gradient is introduced to find the transport plan between the two graphs G_1 and G_2. The idea of the algorithm is the following : at each step, the gradient of the loss function is computed ($\nabla H(\gamma)$), and a direction of descent is determined as the transport plan that minimizes the OT transport between μ, ν, with $\nabla H(\gamma)$ as the cost function, with Sinkhorn algorithm. The step size is then determined through a grid-search, and the algorithm stops when the convergence criterion is met, or the maximum number of iterations is reached.

Corrolary 1 of the paper states that it takes $O(\frac{1}{\epsilon^2})$ iterations to find a solution whose sub-optimality gap is below ϵ.

3.5 Some concepts

Some key points of the method are not explained in the paper and lead to some research. They are detailed in the following sections.

3.5.1 Kernel classification

Kernel method :

The main objective of the paper is to be able to determine a clever distance between graphs in order to perform efficient classification of those graphs through kernel methods. The idea of kernel methods is that we like to have linearly separable sata when it comes to classification. very simple algorithms such as SVM can then be used to classify data. however, it is very common to have non linear separable data. The idea of kernel methods is to define a function ϕ

that maps the data into a higher dimesional space, in which they are then linearly separable. The SVM algorithms then finds an hyperplan that separates the two classes by maximizing the margin between the two classes. An hyperplan is defined by its normal vector w and its bias w_0. The points on the hyperplan are the one following this equation : $w^T\phi(x) + w_0 = 0$. The points on a side of the hyperplan are the ones for which $w^T\phi(x) + w_0 > 0$, and on the other side the inverse inequality. A separator hyperplan is the the one that separates the two classes : $\forall k, sgn(y_k) \times (w^T\phi(x_k) + w_0) \geq 0$. The optimal hyperplan is the determined by maximizing the margin between the two classes :

$$w_{opt} = \min_{w, \forall k y_k(w^T w_k + w_0 \geq 1)} \frac{1}{2} \|w\|^2$$

The dual form of this optimization problem is :

$$\max_{\alpha_k \geq 0 \sum_{k=1}^{p} \alpha_k y_k = 0} \sum_{k=1}^{p} \alpha_k - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \tag{1}$$

$$= \max_{\alpha \geq 0, \alpha^T y = 0} \alpha^T e - \frac{1}{2} Tr(K(Y\alpha)(Y\alpha)^T) \tag{2}$$

(with the notations from Lanckriet et al. 2004).

Kernel trick :

However, this method can be very computationally expensive, as the dimension of the space in which the data is mapped can be very high. The kernel trick introduces a kernel function $K(x, x') = \phi(x)^T\phi(x')$ that enables to compute the dot product in the higher dimensional space without computing the mapping ϕ. Indeed, with the previous notations, the separator hyperplan can be written as : $\{x \in \mathcal{X}, s.t. w^T\phi(x) + w_0 = 0\}$, which can be re-written as $\{x \in \mathcal{X}, s.t. K(w, x) + w_0 = 0\}$. This trick enables to prevent to compute the mapping ϕ. According to Mercer's theorem (Mercer 1909), a function K is a valid kernel if and only if the matrix K is positive semi-definite, which is not the case in the article studied. To overcome this issue, the authors decide to treat their undefinite kernel as the noisy observation of a true kernel. The idea is to solve the problem for a semi definite positive kernel (K) that is very close to the one actually used (K_0). With the same notations as Lanckriet et al. 2004, the problem becomes :

$$\min_{K \succeq 0} \max_{\alpha^T y = 0, 0 \leq \alpha \leq C} \alpha^T e - \frac{1}{2} Tr(K(Y\alpha)(Y\alpha)^T) + \rho \|K - K_0\|_F^2$$

The idea is from Luss et d'Aspremont 2007, who introduces a penalty term instead of looking for a very close semi definite positive kernel.

Example with the Gaussian kernel :

A common kernel is the gaussian kernel. Let x and x' two points in $\mathbb{R}^d$. The gaussian kernel is defined as : $K(x, x') = exp(-\frac{\|x - x'\|^2}{2\sigma^2})$.
$K(x, x') = e^{-\frac{\|x\|^2}{2\sigma^2}} \times e^{-\frac{\|x'\|^2}{2\sigma^2}} \times e^{\frac{x^T x'}{\sigma^2}}$.
However, $e^{\frac{x^T x'}{\sigma^2}} = \sum_{k=0}^{\infty} \frac{1}{k!} \left(\frac{x^T x'}{\sigma^2}\right)^k$, which can be expressed as a scalar product.

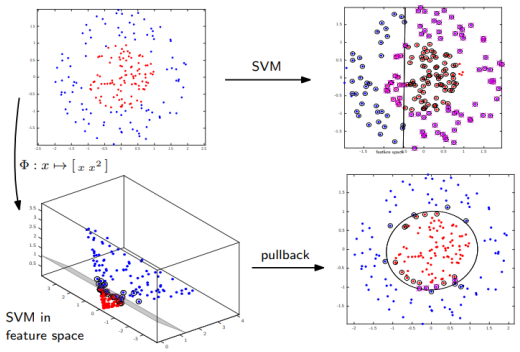


FIGURE 1 – Illustration of the kernel trick, from OUDOT 2022

3.5.2 Heat kernel random walk

As discussed above, a key step of the method is to define an embedding function that takes as an input a node and outputs a vector in $\mathbb{R}^m$. Two embedding functions are used : one for the features on the node (ξ_f) and one for the structure of the node (ξ_s). The idea of the heat kernel random walk is to learn for each graphe an embedding function. The method is explained in ABU-EL-HAIJA et al. 2018. The idea is to randomly select a node of the graph (v_0), and then to randomly select a neighbour of this node(v_1), and continue until a certain number of steps is reached. This leads to a chain of linked nodes $v_0, v_1, ..., v_k$. The embedding of a node v_i is then moved closer to the embedding of the following nodes $v_{i+1}, \ldots, v_{i+C}$, with C a context parameter.

4 Numerical Experiments

As the code was available here, I decided to test it on a simple classification task to assess the performance of the method and the influence of some parameters.

4.1 Data

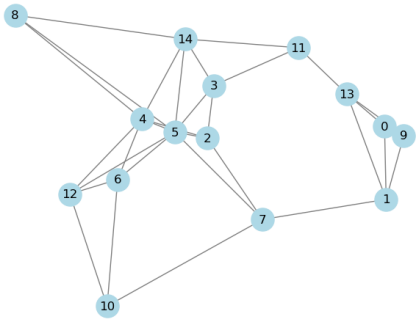
I decided to perform a very simple classification task on nearest neighbourghs graphs. The graphs were generated with the following methodology :

— Sample uniformly n=*15* points in $[-1, 1]^2$.
— Connect each point to its k nearest neighbours.
— Set the coordinates of the point as the node value.

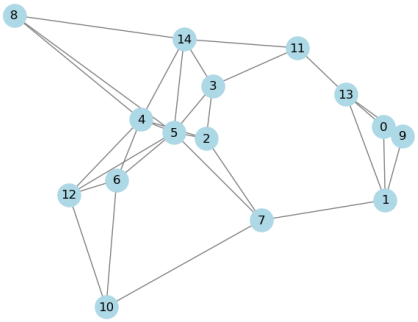
The two classes of graphs correspond to values of k equal to *2* and *3*.
I have then generated a training set of 400 graphs, containing 200 graphs of each class, and a test set of 100 graphs, containing 50 graphs of each class.

Some insights on the data :

The 200 2-nn graphs contain TODO ADD NUMBER edges in average. The 200 3-nn graphs contain TODO ADD NUMBER edges in average. See Fig. 3 for more details.
56.5 % of the 2nn graphs are connected, and 90.0 % of the 3nn graphs are connected.

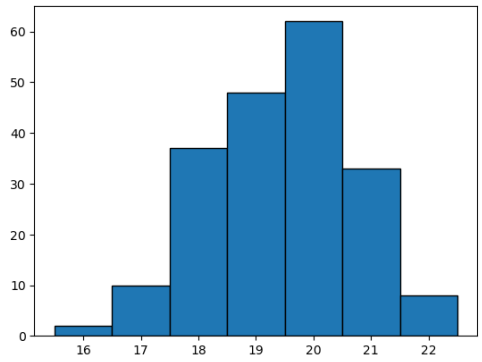


(a) Example of a graph with k = 2 TODO CHANGE IMAGE

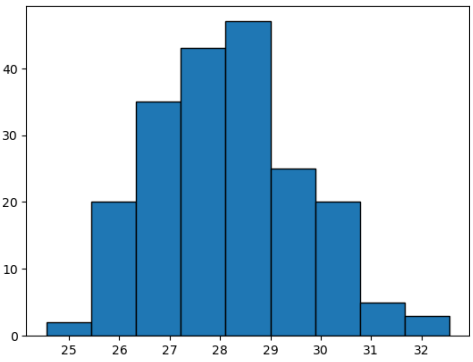


(b) Example of a graph with k = 3

FIGURE 2 – Example of two graphs of each class



(a) Distribution of the number of edges in the graphs with k = 2



(b) Distribution of the number of edges in the graphs with k = 3

FIGURE 3 – Number of edges in the train set

4.2 Methodology

The SVC classifier implemented in the paper was used to classify the graphs. It was trained on the training set and tested on the test set. The performances was evaluated through accuracy and F1 score. I also decided ti study the impact of the differnt terms by setting some regularization terms to 0 and by making some vary along different values. The computational time and the performance were then compared.

Two baselines model were also tested : the default random forest classifier from sklearn, and a SVM classifier with a gaussian kernel.

4.3 Results

5 Conclusion

critique : code ne marche pas, très long pour des résultats pas si fou petites erreurs : " two strongly convex terms " ben boyons... on me prend pour un jambon.

6 Connection with the course

connection avec cours 4 : régularisation intelligente. Par contre le problème n'est pas strictement convexe

lien avec sinkhorn, différences avec l'algo vu en cours.

Références

Abu-El-Haija, Sami et al. (2018). "Watch your step : Learning node embeddings via graph attention". In : *Advances in neural information processing systems* 31.

Labib, Karim, Przemyslaw Uznanski et Daniel Wolleb-Graf (2019). "Hamming distance completeness". In : *30th Annual Symposium on Combinatorial Pattern Matching (CPM 2019)*. T. 128. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, p. 14.

Lanckriet, Gert RG et al. (2004). "Learning the kernel matrix with semidefinite programming". In : *Journal of Machine learning research* 5.Jan, p. 27-72.

Luss, Ronny et Alexandre d'Aspremont (2007). "Support vector machine classification with indefinite kernels". In : *Advances in neural information processing systems* 20.

Mercer, James (1909). "Xvi. functions of positive and negative type, and their connection the theory of integral equations". In : *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character* 209.441-458, p. 415-446.

Oudot, Steve (2022). *Algorithms for Data Analysis in C++*.

Titouan, Vayer et al. (2019). "Optimal transport for structured data with application on graphs". In : *International Conference on Machine Learning*. PMLR, p. 6275-6284.