A Regularized Wasserstein Framework for Graph Kernels

Grégoire Béchade

17/01/2025

1 Abstract

This report aims to present the work of the authors of the paper "A Regularized Wasserstein Framework for Graph Kernels". They introduce a distance between graphs based on an optimal transport problem with a cost matrix that reflects the similarity between the features of the nodes of the graphs. Two regularization terms are added to preserve the similarity of structures of graphs that are "close" for this distance : a term to preserve similarity between the nodes that are connected in the optimal transport plan (LW), and the Gromov-Wasserstein discrepancy, that ensures a global similarity between graphs. The authors introduce an algorithm to solve this non convex problem, along with some theoretical results on its convergence. The method was tested on a very simple classification task, and the results show that it is very time-consuming and does not outperform by far very simple models (random forest, SVM with gaussian kernel). The impact of the two regularization terms was also tested, and many errors of convergence were raised, leading to believe that a careful fine tuning of the parameters has to be done to be able to use the method. One of the parameters, the Gromov-Wasserstein regularization term, was found to have very little impact on the performances of the classifier.

2 Introduction

The authors of this paper aim to develop a Wasserstein distance between graphs, in order to perform graph classification. Several studies have been done to define distance between graphs to measure their dissimilarity. Those approaches rely on measuring the difference between the node embeddings (Nikolentzos, Meladianos et Vazirgiannis 2017), the node features, or the graph structure (Titouan et al. 2019). In this work I will first present the method introduced by the authors to define a distance between two graphs that takes into account both node embeddings and structure similarity of the graphs, along with mathematical tools necessary to understand the method. I will then discuss the algorithm used to solve the problem, and the results of some numerical experiments I did to test the method.

3 Theoretical Background

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

Notations :

Let $G = (V, E)$ a undirected graph with V the set of vertices and E the set of edges. We denote n the number of vertices and m the number of edges. We consider $\xi_f : V \to \mathbb{R}^m$ a feature embedding function (see 3.1), and $\xi_s : V \to \mathbb{R}^k$ a structure embedding function.
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 μ and ν their associated probability distributions. The authors define the RW discrepancy between those two graphs as :

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

This is an optimal transport problem, with two regularization terms : LW and GW. 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 $(x_i)_{i \in \{1,\ldots,|V|\}} \in \mathbb{R}^m$, 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.

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 these features live).

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. This notion was studied in the MVA course Machine Learning For Time Series, from Laurent Oudre. 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 evaluate the neighbourhood 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 explain how they managed to compute it. An 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 would 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 :

$\Omega_\mu(\gamma) = \frac{1}{n_1^2} \sum_{i,j} a_{i,j} \left\| \hat{e}_i^\mu - \hat{e}_j^\mu \right\|^2$, with $a_{i,j}$ the adjacency matrix of G_1.

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 :

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

This distance preserves a proximity between nodes connectes in 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 matrices. 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 that 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 Kullback-Leibler 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 neighbourhood 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 Conditional 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.

Corollary 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.

This algorithm has a complexity of $O(t(n^3 + kn^2))$ with t the number of iterations, $n = max(n_1, n_2)$ the number of nodes in the largest graph, and k the dimension of the embeddings of the nodes. However, in my experiences, I have noticed that the computational time was multiplied by a factor 9 when the number of nodes was multiplied by 3, so the dominant term might be the squared one. This complexity is therefore similar to the one of the Sinkhorn algorithm, which has a complexity of $O(T \times n^2)$. However, with different datasets, the complexity might be in n^3 and the computational time might be very high.

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 data 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-linearly separable data. The idea of kernel methods is to define a function ϕ that maps the data into a higher dimensional space, in which they are then linearly separable. The SVM algorithms then finds an hyperplane that separates the two classes by maximizing the margin between the two classes. An hyperplane is defined by its normal vector w and its bias w_0. The points on the hyperplane are the one following this equation : $w^T\phi(x) + w_0 = 0$. The points on a side of the hyperplane are the ones for which $w^T\phi(x) + w_0 > 0$, and on the other side the inverse inequality. A separator hyperplane is the one that separates the two classes : $\forall k, sgn(y_k) \times (w^T\phi(x_k) + w_0) \geq 0$. The optimal hyperplane 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 computing the dot product in the higher dimensional space without computing the mapping ϕ. Indeed, with the previous notations, the separator hyperplane 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 indefinite kernel as the noisy observation of a true kernel. The idea is to solve the problem for a positive semi-definite 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 :

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

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.

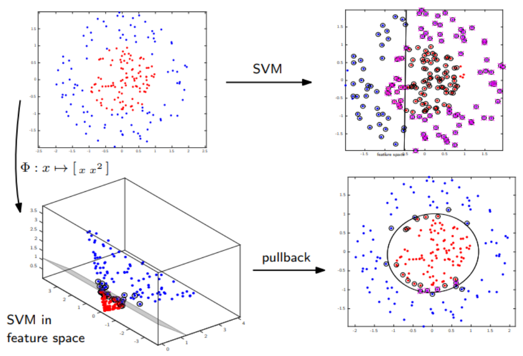


FIGURE 1 – Illustration of the kernel trick, from OUDOT 2022

Example with the Gaussian kernel :

A common kernel is the gaussian kernel. Let x and x' be 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.

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 graph 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.

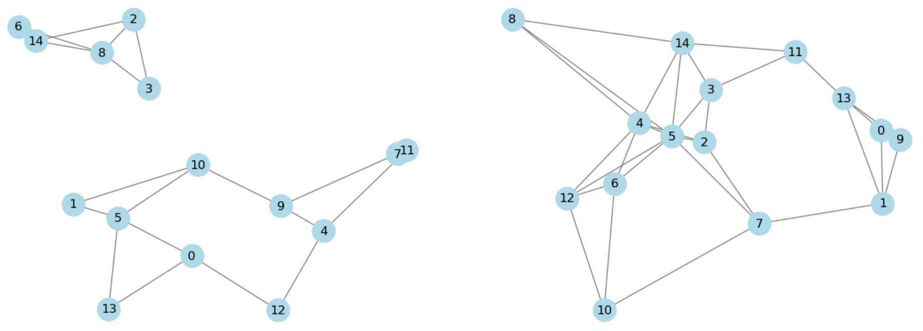
4.1 Data

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.



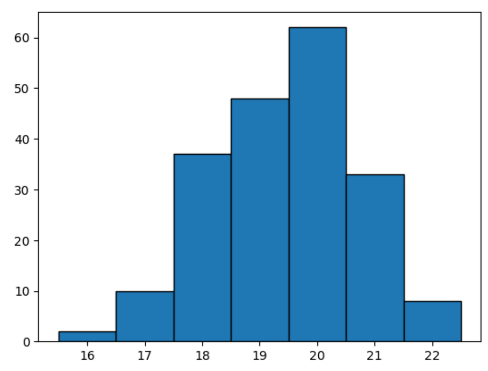
(a) Example of a graph with k = 2 (b) Example of a graph with k = 3

Figure 2 – Example of two graphs of each class

Some insights on the data :

The 200 2-nn graphs contain 19.45 edges in average. The 200 3-nn graphs contain 28.55 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) Distribution of the number of edges in the graphs with k = 2

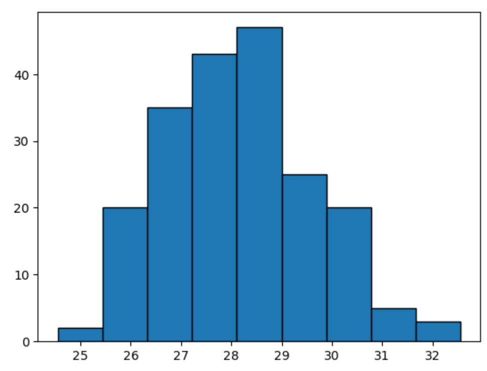

(b) Distribution of the number of edges in the graphs with k = 3

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 were evaluated through accuracy and F1 score. I also decided to study the impact of the different terms by setting some regularization terms to 0 and by making some vary along different values. I have assessed the performances of the classifier with β_2, ρ, λ_μ, λ_ν and λ_g set to 0, and with β_1 and β_2 scaled among different values (see Results). As the algorithm computes an optimal transport with a special regularization term, I have also tested the performances of the classifier with the Sinkhorn algorithm, which is a simpler algorithm. The computational time and the performances in terms of F1-score and accuracy were then compared. Two baseline model were also tested : the default random forest classifier from sklearn, and a SVM classifier with a gaussian kernel.

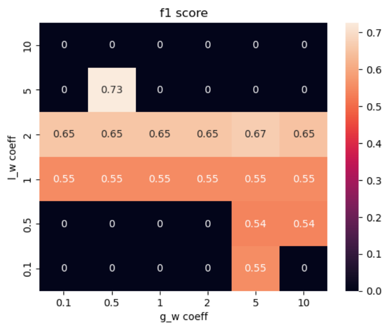
4.3 Results

The results of the first experiment, which consist into assessing the performances of the classifier with different parameters set to 0 are presented in the following table.

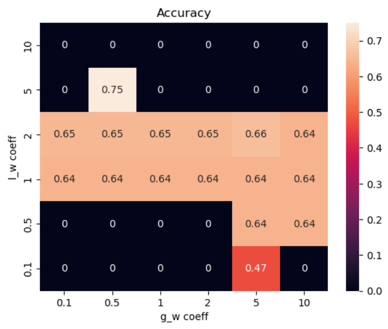
	Time (min)	F1-score	Accuracy
Default	47	0.65	0.73
$\beta_2/\lambda_g = 0$	25	0.78	0.76
$\rho = 0$	62	0.63	0.71
$\lambda_\mu = 0$	58	0.65	0.71
$\lambda_\nu = 0$	32	0.65	0.72
$\lambda_g = 0$	38	0.61	0.69
Sinkhorn algorithm	11	0	0.5
Random Forest	2.10^{-2}	0.63	0.67
SVC Gaussian kernel	0	0.60	0.66

TABLE 1 – Performances of the classifier with different variations

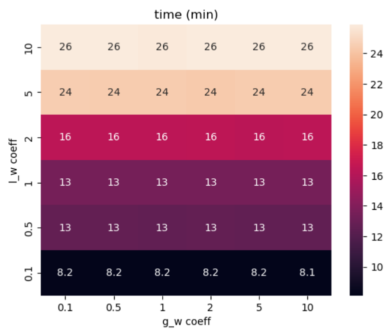
We can note from 1 that the Gromov-Regularization term diminishes the performances of the algorithm. Indeed, the F1-score and the accuracy are higher when β_2 is set to 0. It makes sense when considering that this term is a non convex one and should therefore prevent good convergence. Another observation is that the Sinkhorn algorithm has very bad results (non convergence errors were raised during the training), which is surprising as theoretical results state that it should converge. Moreover, it is surprising to observe that the different parameters have little impact on the performance of the classifiers. Indeed, the authors have already tested the impact of the different parameters and had concluded that they improved significantly the results and sometimes helped to make the computations faster. Finally, one can observe that a mere random forest classifier has roughly the same performances as the normal classifier from the paper, for a computational time drastically lower.



(a) F1-score



(b) Accuracy

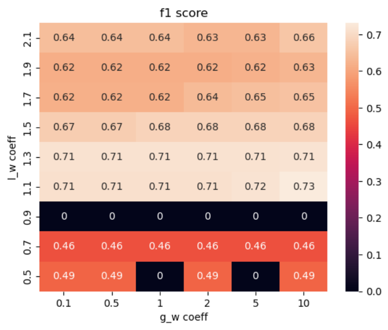


(c) Computational time

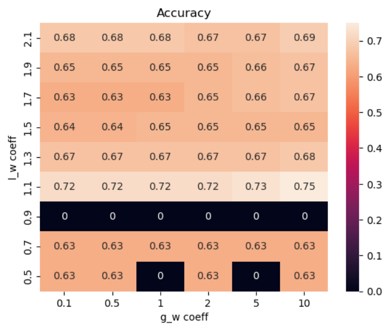
FIGURE 4 – Results of the classifier with variations of the two regularization terms

The figures 4a, 4b and 4c show the evolution of the F1-score, the accuracy and the computational time of the classifier with the variations of the GW and LW regularization terms.

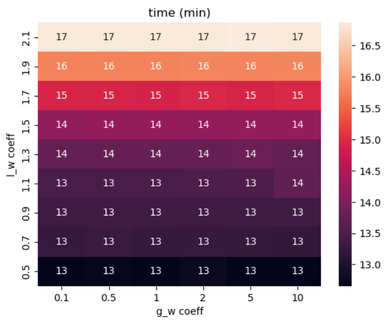
First, we can note that in many cases, the algorithm does not manage to converge. The good parameters have to be carefully found to have a good convergence. One can also notice that the GW regularization term has very little impact on the computational time and performance of the classifiers. A more precise grid-search was then performed in the zone in which the algorithm converges (see Fig. 5a, 5b and 5c). We can observe that we have roughly the same results as before. However, there seems to be an discontinuity around l_w_coeff = 1, as the algorithm does not converge in 0.9 and have the best performances next to the not converging point.
The optimal value for l_w lies around 1.1, and g_w does not seem to impact the performances of the classifier.



(a) F1-score



(b) Accuracy



(c) Computational time

FIGURE 5 – Results of the classifier with variations of the two regularization terms with a more precise grid search

Theoretical analysis of the non convergence :

The theorem 1 from WIJESINGHE 2021 gives a convergence criterion for the SCG : the minimal suboptimality gap has the following upper-bound :

$$\min_{0\leq i\leq k} \delta_i \leq \frac{max(2h_0,(L-\sigma)\times diam_{\|.\|}(\pi(\mu,\nu))^2)}{k+1}$$

However, the coefficients L and σ (lipschitz constants) were not explicitly given, so I did not manage to see if my non convergence was due to a value of the suboptimality gap above the stopping criterion.

A KNN classifier :

I have decided to test the performances of a KNN classifier on the same dataset, with the RJW distance (the distance between the two graphs). Cross validation was used to determine the best value of k. For each value of k, the F1-score and the accuracy were computed 200 different time, with a different train and validation set each time. The results are presented in the following figure (6), with the standard deviation. An optimal value of k was found to be $k = 14$. The algorithm was then tested on the test set, and the F1-score and the accuracy were computed, with a value of 0.70 and 0.68 respectively.

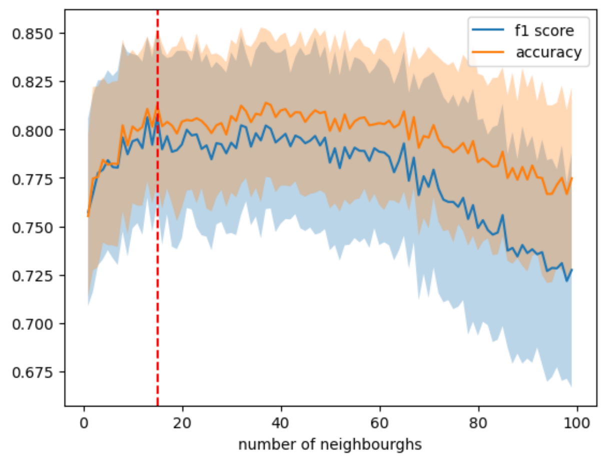


FIGURE 6 – Evolution of the F1-score and the accuracy of the KNN with k. The red bar is the optimal value of k

5 Conclusion

5.1 Comments on the paper

As a first conclusion, I would like to make some comments on the article, of some aspects that could be improved.

1. The code was available online but hard to understand and had some mistakes that might be due to old versions of Python, but also to errors (variables not declared for instance). I had to change it a bit to make be able to run it on my computer. Moreover, the code is not very user-friendly, as many variables can't be directly accessible to be modified. I had to change them manually in the relevant files to assess their impact on the classifier.
2. The paper has some inconsistency. For instance, they explain in the abstract that "two strongly convex regularization terms" are added to regularize the problem, but explain later that the Gromov-Wasserstein regularization is not convex.
3. On the method, it introduces an interesting measure to compare graphs, but the simple tests that were run show that it is very time consuming, faces lots of non convergence points and does not outperform very simple baselines.

5.2 Final conclusion

Finally, the author introduce a measure based on an optimal transport problem to evaluate the dissimilarity between two graphs. This method takes into account a cost matrix that reflects feature similarity, and adds two regularization terms to preserve pairwise and neighbourhood similarity. The optimization problem is NP-hard to solve, but the authors introduce an algorithm that can solve it in $O(N^3)$. Theoretical results are proved, with the guarantee to obtain a convergence toward a point at distance ϵ from a local minimum in $O(\frac{1}{\epsilon^2})$ iterations. However, numerical experiments on a simple task (SVC and KNN classification on a 2d graphs of 2 and 3 nn) show the defaults of this method. It is computationally time-consuming, for very little number of points and does not manage to outperform very simple models that are way faster. The impact of the different parameters were assessed and did not show significant differences in the performances of the classifier, except for β_2/λ_g, who seems to have a negative impact on the performances. A more precise grid-search was performed to assess the impact of both GW and LW regularizations, and showed that the algorithm does not converge in many cases. Moreover, the GW regularization does not seem to have significant impact on the results.

6 Connection with the course

First, a question that arises in the paper is to find the optimal transport plan between two graphs, which are embedded as points in $\mathbb{R}^m \times \mathbb{R}^k$. This problem is an optimal transport between two sums of a different amount of diracs in very high dimension. It therefore corresponds to the Kantorovich problem with discrete distributions, that was studied in the course 2. However, the authors have a differnet approach than the one studied in course to solve this problem. Instead of adding an entropic regularization that makes the problem strictly convex, they introduce two regularization terms that enable to preserve the structure of the graphs, but make the computations harder.

The whole problem of the paper is to solve an optimal transport between two probability distributions, with a penalty adapted to the nature of the data. It has a clear link with the course 4, that presents another algorithm to solve the optimal transport problem : the Sinkhorn algorithm. In the course, the penalty was the Shannon entropy of the transport plan, while the two penalties introduced in the paper are the local barycentric and the global connectivity penalties. The difference with the course is that the Shannon penalty makes the problem strictly convex, while despite the fact that the global barycentric penalty is strongly convex with respect to γ, the Gromov-Wasserstein distance is not convex, and therefore leads to a more complex optimization problem. Therefore, the authors managed to prove that the algorithm converges toward a stationary point, there is no guarantees on the convergence to the global minimum. On the contrary, the Sinkhorn algorithm consists in iterations of a contracting function, which has as a consequence that the algorithm converges.

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.

Nikolentzos, Giannis, Polykarpos Meladianos et Michalis Vazirgiannis (2017). “Matching node embeddings for graph similarity”. In : *Proceedings of the AAAI conference on Artificial Intelligence.* T. 31. 1.

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.

Wijesinghe, Author (2021). “Regularized regression”. In : *Journal of Example* 10.2, p. 123-145.