Seminarreport: Optimal Transport

Andrea Cremer

August 30, 2020

1 Introduction

In this report I will present and compare the 3 papers of the session "Optimal Transport" of the Seminar "Principles of Data Mining and Learning Algorithms: Selected Papers from NeurIPS 2019". These are

- "GOT: An Optimal Transport framework for Graph comparison" by Maretic, Gheche, Chierchia and Frossard ([2])
- "Wasserstein Weisfeiler-Lehman Graph Kernels" by Togninalli, Ghisu, Llinares-López, Rieck and Borgwardt ([3])
- "A Graph Theoretic Additive Approximation of Optimal Transport" by Lahn, Mulchandani and Raghvendra ([1]).

For each paper I will give a short overview about the presented method and the experiments. Then I will discuss in a critical way the structure, style and experiment setup. At the end I will concentrate to compare them with each other and show advantages, disadvantages, similarities, how they deviate and how they can be connected.

All 3 papers are related to "Optimal Transport" and the application to compute the Wasserstein distance to measure similarity between graphs is the central topic that connects all 3 papers. Therefore I will first give a short overview about this application: As mentioned in [2] and [3] defining a useful distance measure, that is capable to store important information about graph structures with the goal to get a distance measure for the similarity between graphs is still a non-trivial and relevant topic. Both papers [2] and [3] use as key idea to convert the graphs towards a numerical value (vector, matrix) that represents the overall graphs structure and then apply the Wasserstein distance on these this numerical sets to obtain the promised distance measure on graphs. Here [2] uses the 2-Wasserstein distance and [3] the 1-Wasserstein distance. Generally the p-Wasserstein distance, defined as

$$\mathcal{W}_p(\mu,\nu) = \Big(\inf_{\gamma\in\Gamma(\mu,\nu)} \int_{M\times M} d(x,y)^p d\gamma(x,y)\Big)^{1/p},$$

with $\Gamma(\mu,\nu)$ describing the set of transportation plans from μ to ν. It measures for two given probability distribution μ and ν how many effort is necessary to transport one

probability distribution towards the other one, so the costs of the optimal transportation plan between both distributions.

2 Analysis of the Papers

2.1 GOT: An Optimal Transport framework for Graph comparison

The procedure:
In the paper "GOT: An Optimal Transport framework for Graph comparison" by Maretic, Gheche, Chierchia and Frossard ([2]) the authors explained their framework to measure the similarity between graphs based on the 2-Wasserstein distance.
The key idea is to computed a probability distribution for each graph that can store structural information and then to apply the Wasserstein distance on them. This leads to an optimization problem that needs to be solved by the framework. Here a stochastic gradient descent method is used.
For the probability distributions the normal distribution with zero mean and with covariance using the pseudoinverse L_i^+ of the respective graph laplacians L_i for graph $G_i,\ i \in \{1,2\}$ is used. Using the normal distribution lead to a simplification in the computation of the 2-Wasserstein distance that the authors exploited.
In the optimization problem, the enumeration of the nodes in each graph leads to different probability distributions and therefore to different distances, although the graphs are identical. To avoid this, all permutations of node enumerations need to be taken into account. This can be expressed using a permutationmatrix P on for the laplacian.
The constraints to be a permutationmatrix for P lead to a complicated optimization problem that looks in total as follows:

$$\min_{P\in\mathbb{R}^{N\times N}} \underbrace{Tr(L_1^+ + P^T L_2^+ P) - 2\cdot Tr(\sqrt{L_1^{+/2} P^T L_2^+ P L_1^{+/2}})}_{=\mathcal{W}_2(\nu^{G_1},\mu_P^{G_2})}$$
$$\text{s.t} \quad \begin{cases} P \in [0,1]^{N\times N} \\ P\cdot \mathbb{1}_{N\times 1} = \mathbb{1}_{N\times 1} \\ \mathbb{1}_{1\times N}\cdot P = \mathbb{1}_{1\times N} \\ P^T\cdot P = \mathbb{1}_{N\times N} \end{cases}.$$

To avoid the constraints of the optimization problem, they use instead of forcing P to be a permutation matrix the Sinkhorn operator $S_\tau(P) = A\cdot\exp(P/\tau)\cdot$ for any $N\times N$ matrix P. The idea behind this is, that the Sinkhorn operator leads to a permutation matrix in the limit for $\tau\to 0$. The matrices A and B can be computed iteratively (for details see section 3 of [2]). This reduces the optimization problem to

$$\min_{P\in\mathbb{R}^{N\times N}} \mathcal{W}_2(\nu^{G_1}, \mu^{G_2}_{S_\tau(P)}). \tag{1}$$

The problematic to solve this expression is, that $\mathcal{W}_2(\nu^{G_1}, \mu^{G_2}_{S_\tau(P)})$ is highly non-convex, so that it can easily happen that gradient decent converges towards a local minima and

not a global one. To avoid this, the authors used stochastic method.
Instead of minimizing over P in (1) they want to optimize over the expectation of the above function with respect to a probability distribution q_θ used to compute P. The idea behind this is, that in the minimum of the expectation the distribution puts all mass on the minimum of (1). Again the (multivariate) normal distribution is used which leaded again to further simplifications. By sampling over the distribution an approximation on the gradient of the respective optimization problem is obtained that is used for the gradient descent method. It is still unclear, if this leads to convergence towards a global minimum of the original problem.

Without any simplifications or modifications one iteration needs $\mathcal{O}(n^3)$ runningtime. For a faster implementation they proposed in [2] to approximate the square root based on SVD and to exploit the sparsity of the matrices there. Moreover they gave the idea to make a diagonal shift on the laplacian matrices and to compute the inverse instead of the pseudoinverse.

Experimental results:
In their paper, the author showed 3 different kind of experiments: The first two ones shows that GOT is able to capture structural information about graphs. The third one shows the relevance of the computed transportation plan.
In the first experiment they constructed some graphs of specific structure and a noisy version of each graph by randomly removing edges with a fixed probability and permutating the vertices. Then they compared 3 different alignment methods, one based on the distance from the paper. The relation between error in alignment and edge removement probability is compared to 2 other measures regarding the error term of the specific models. GOT performed very well in the experiments.
In the second experiment they analysed the quality of GOT regarding graph classification. For this they had graphs of different classes of graph structures with similar number of vertices and edges. Based on the computed distance by GOT and a 1-nearest-neighbour-classification algorithm they classified the graphs and compared the result using the confusionmatrix measuring the wrong classified graphs. Also for this setting GOT performed very well.
The last experiment focused on the computation of the transportationplan by GOT and not really on the distance. For this they used MNIST data of digit images. Regarding the different digitsclasses they constructed a single graph for each digit class representing the average properties of the underlying images. Then they computed via GOT the transportation plans between these digit-graphs. Starting from the class of the number zero they could identify the number of the underlying class computed from the transportation plan based on class zero. Analogue results were also recognizable using the FASHION MNIST data set.

Critic:
The Introduction is used in a good way to show the broad application fields of the presented method. The overall structure of the paper is clear. The shown graphics helps

the reader to get an better understanding and support the arguments of the authors for their used ideas.
Although there are no proof given and also a good mathematical background is provided from the authors, the basic structure of the framework and the main ideas are easy to understand, without comprehending all details. Nevertheless it would be useful to give the reader some more impressions about the theoretical details used in the computations by some background knowledge in the appendix.
In the way how the authors presented their work, some discussions and additional research is necessary to get the overall picture more clearly. Although the reader is not able (based on the knowledge given by the paper) to understand all presented formulas in detail, they are still useful to get an understanding how the different steps of the GOT framework work and to see the red line in the paper.
Regarding the experiments, the paper showed 2 different setups to convince the reader of the capability to capture significant information of graphs by GOT. But both setups for the first two experiments are very artificial. It would be nice to see how GOT behave on real world data regarding classification compared to other methods instead of the artificial experiments and method-specific error measures used in the paper. Interesting to see is, how good the transportationplan in the third experiment could visualize the number of the underlying digit class. So it is a advantage of the paper that they showed their results in the experiments in a visual and also in a more analytical way via error measures to give the reader more trust in their presented method.

2.2 Wasserstein Weisfeiler-Lehman Graph Kernels

The procedure:
In their paper "Wasserstein Weisfeiler-Lehman Graph Kernels" the authors Togninalli, Ghisu, Llinares-López, Rieck and Borgwardt showed a method to obtain a graph kernel to measure similarity between objects by comparing their substructures. Again the basis for the developed method is the Wasserstein distance, that solves an optimal transportation problem. In this paper, the authors explicit want to deal with graphs that have continuous or discrete node abels. In related work, graphs with such a structure (continuous attributes) lead to problems.
The method described in [3] can be splitted into two parts. First, the authors showed how to construct a graph embedding scheme based on the labels or attributed, where the Wasserstein distance can be applied on. A graph is labeled, if each node $v \in V$ has a label $l(v) \in \mathcal{L} \subset \mathbb{R}$ from a finite set $\mathcal{L}$. G is attributed, if for each node a continuous attribute vector $a(v) \in \mathbb{R}^m$ is given. After showing their embedding scheme they showed how the Wasserstein distance can be computed on that scheme and how this distance measure is used to obtain a kernel function.
The graph embedding scheme $f^H : G \to \mathbb{R}^{|V| \times m(H+1)}$ on graph G computed a matrix $f^H(G) =: X_G^H = [X_G^h(v)]_{v \in V, h=0,\dots,H} \in \mathbb{R}^m$ ($m = 1$ for labeled graphs) where each row represents the structure of one node with growing neighborhood within the graph. They are computed iterative and different if G is labeled ($m = 1$) or attributed. For a fixed node $v \in V$ initially $x_G^0(v)$ belongs to the label or attribute value of v. For $1 \leq h < H$,

the next value $x_G^{h+1}(v)$ is dependent on the previous values from the neighborhood, so

$$x_G^{h+1}(v) = \begin{cases} \text{hash}(x_G^h(v), \mathcal{N}^h(v)) & G \text{ labeled} \\ \frac{1}{2}\left(x_G^h(v) + \frac{1}{deg(v)} \sum_{u \in \mathcal{N}(v)} w((u,v)) \cdot x_G^h(u)\right) & G \text{ attributed} \end{cases},$$

where $\mathcal{N}^h(v) = \{x_G^h(u_o), ..., x_G^h(u_{deg(v)-1})\}$ and $\mathcal{N}(v)$ is the neighborhood of v. For two given (both labeled or attributed) graphs G_1 and G_2, first the corresponding graph embedding schemes $f^H(G_1)$ and $f^H(G_2)$ get computed. They are used to compute the 1-Wasserstein distance. By exploiting, that the embeddings form finite sets, this leads to the simplification

$$D_W^f(G_1, G_2) = \min_{P \in \Gamma(f^H(G_1), f^H(G_2))} \langle P, M \rangle,$$

where $\langle \cdot, \cdot \rangle$ is the Frobenius norm and $\Gamma(f^H(G_1), f^H(G_2))$ is the set of transportation plans from $f^H(G_1)$ to $f^H(G_2)$ and P a transportation matrix. M is the distance matrix containing the distances between each element of $f^H(G_1)$ and $f^H(G_2)$ for a defined ground distance $d(\cdot, \cdot)$. This distance function between each pair of vertices from the different graphs, are used in M. In [3] they distinguished for labeld and attributed graphs how the distance measure d looks like: For the attributed case the euclidean norm between the node embedding vectors, $X_G^H(v)$ is used. For the labeled case one counts how many of the $H+1$ entries in each node embedding vector differ and normalize this by deviding through $H+1$.
The running time using a naive simplex network implementation for solving the oprimization problem yields to $\mathcal{O}(n^3 \log(n))$ running time, where $n = |V_1| + |V_2|$.
A kernel given set of graphs $G_1, ..., G_N$ is given by $K_{\text{WWL}} = \exp(-\lambda D_W^f)$ and the kernel value between graph G_i and G_j by $K_{\text{WWL}}(G_i, G_j) = \exp(-\lambda D_W^f(G_i, G_j))$. K_{WWL} is positive definite for any $\lambda > 0$ in case of labeled graphs. For continuous graphs this is an open problem.

Experimental results:
The experiments used different datasets including both,real-world and artificial data, labeled and continuous attributed that should be cassified by WWL and other compared methods.For WWL a SVM to classify using the constructed kernel was used.
The different methods were compared in term of classification accuracy. In the different datasets WWL performed very good. For the attributed case still WWL performed very well and scored first on average rank.

Critic:
Comparison with [2], [3] does not use the abstract and introduction to show application fields of their work. It directly focuses on the idea of constructing a kernel without really explaining why this is important and what the advantages and applications of that are. One main point in their motivation part is the goal to be able to deal with continuous attributed graphs. But still they could not show the positive definitness of the kernel using their approach for this class of graphs. This is kind of contradictory to

the motivation of the paper.
All in all the paper has a good structure and one can easily follow the major steps presented by the authors. Similar to [2] no formal proofs are given directly, but often referred to the appendix, so that the interested reader can get there some more formal arguments of the presented method. More proof are not really necessary in the paper, because it shows more ideas than complicated mathematical connections.
The paper introduced all major tools and definitions in a formal and descriptive way, such that no extra research is necessary to understand the major topic. Only in the application for the obtained kernel function there could be some more background, especially because the author gave that property a huge priority.
The experiment they used uses both artificial and real-world data which is a huge advantage compared to [2].

2.3 A Graph Theoretic Additive Approximation of Optimal Transport

The procedure:
In the paper "A Graph Theoretic Additive Approximation of Optimal Transport" by Lahn, Mulchandani and Raghvendra ([1]) the author discussed a approximation method to solve the graph theoretical version of the optimal transport problem. One possible application would be in the computation of the Wasserstein distance or other similarity measure tasks, where exact cost are not necessary to know and tendencies should be sufficient for classification and alignment tasks. The goal of [1] is to present a fast method to approximate the graph theoretical optimization problem of optimal transport.

An instance of the algorithm is a fully connected bipartite graph $G = (A \dot{\cup} B, E)$ with demand nodes A and given demands $d_a,\ \forall a \in A$ and supply nodes B given supplies $s_b,\ \forall b \in B$ and transportation costs $c : E \to \mathbb{R}_{>0}$ for one unit of supply over the edges. There are no restrictions regarding costs, demands and supplies. Goal is to compute a transportation plan $\sigma : E \to \mathbb{R}_{\geq 0}$ that is feasible ($\sum_{b \in B} \sigma((a,b)) \leq d_a,\ \forall a \in A$) and maximal ($\sum_{a \in A} \sigma((a,b)) = s_b,\ \forall b \in B$). Moreover the cost of the transportation plan $w(\sigma) = \sum_{e \in E} \sigma(e) \cdot c(e)$ should be minimized by the choice of σ. For given δ, they show how to obtain a δ-close solution. This is a transportation plan σ with $w(\sigma) \leq w(\sigma^*) + \delta \cdot U$, where σ^* is the optimal maximal and feasible transportation plan.
Other methods for this problem either could deal with arbitrary costs, demands and supplies but where slow, or had restrictions regarding the input but were fast. The main goal of the paper is to present a fast method that is also able to deal with arbitrary costs, demands and supplies.
The main part of the method is similar to a method by Gabow and Tarjan for integral demands, supplies and costs using primal-dual-linear programming. To use a similar method the procedure is splitted into 3 parts. First the initial instance, notated with $\mathcal{I}$, is scaled and rounded to an instance $\bar{\mathcal{I}}$ with integral demand, supplies and costs. For this instance they use, similar to Gabow and Tarjan, a primal-dual-algorithm to compute a $\bar{\delta}$-close solution $\bar{\sigma}$ for $\bar{\mathcal{I}}$. Then a fixing and "backwards-rounding" method to convert $\bar{\sigma}$ towards a maximum and feasible transportation plan σ for the initial instance $\mathcal{I}$ is done.

Most complicated is, how to obtain a $\bar{\delta}$-close transportation plan for instance $\bar{\mathcal{I}}$ using a primal-dual algorithm. In the classical primal-dual-network one starts with a dual feasible solution, always remain dual feasible and move in direction of primal feasibility until this is obtained. Then by linear programming theory an optimal solution is found. The procedure in [1] is similar to this. The difference is that one does not start dual feasible in the classical sense, because an approximation instead of an optimal solution is sufficient. Besides working with a relaxed version of dual feasibility the method is similar to the typical primal-dual algorithm: always maintain "relaxed" dual feasible until primal feasibility is obtained. For this one updates the dual variables of the LP by using Dijkstras algorithm first. Based on the updated dual variables, the primal ones get updated by a partial DFS step. This procedure is repeated until primal feasibility (maximality and feasibility for instance $\bar{\mathcal{I}}$) is reached. Then an $\bar{\delta}$-close solution is found. The first and last step can be implemented in $\mathcal{O}(n^2)$, where $n = |A| + |B| = |V|$. The main procedure, the primal-dual-algorithm has a worst-case running time of $\mathcal{O}(n^2(C/\delta)+ n(C/\delta)^2)$, with $C = \max_{e \in E} c(e)$, which leads to the overall running time.
A consideration to obtain a better running time could be by using a nearest neighbor structure. If the updating and query time is poly-logarithmic w.r.t. n, the running time gets linear in n with additional logarithmic terms. Another idea is to get rid of Dikstras Algorithm for the dual variable updating step, to make parallel implementation more amenable.

Experimental results:
There are presented 2 different kind of experiments. Both use real-world-data from MNIST. The transportation costs for this setting measured the costs of transporting pixel colours from one image to the other one. In the first experiments they wanted to show that the worst case running time is not very practical and that the algorithm performs better in practice. The other experiments compares the running time with other algorithms that also compute a δ-close approximation. Here, the presented algorithm took fewer $\mathcal{O}(n^2)$ iterations. Also regarding the wall-clock running time compared to the Sinkhorn algorithm, the algorithm of [1] was faster for closer solutions. So the experiments show, that the algorithm can efficiently compute an δ-close approximation of the graph theoretical optimal transport problem, where, also for small values of *delta*, the running time does not "explode" compared to other methods.

Critic:
The paper has a very short introduction, where one application as similarity measured is mentioned. Then it directly moves towards the problem definition. Although the overall structure of the paper is clear, the reader can easily get lost. Compared to the other papers [2] and [3] it is really technical. The main part consists of statements and their proofs. Although the single proofs are good to understand, the most ones do not help to get an overview of the overall procedure. Instead, the reader can easily get lost in the mass of technical proof details without seeing the big picture. Visual schemes could help to avoid this.

A lot of background knowledge about linear programming is useful to understand the idea behind the primal-dual-algorithm, that forms the main part of the algorithm. For this the paper does not really give many background information. Instead of the detailed proofs it may would have been more useful to explain the major idea of a primal dual algorithm and how this is changed for the approximation case.
In the presented experiments, the authors could convince the reader of the practical efficiency of their method. It would be nice to see how the procedure performs in applications like graph classification or similarity measure in general, that was mentioned in the introduction. For this there are no experiments published in the paper.

3 Comparison between the presented papers

If we compare all papers, it directly can be noticed, that [2] and [3] are very similar compared to [1] in 2 ways: [2] and [3] focused more on explaining ideas than in detailed proofs, what is a huge contrast to [1]. Moreover [2] and [3] both had a central goal to compare graphs and used optimal transportation costs only for the computation of the Wasserstein distance which was part of their methods. In [1] they focused on the detailed computation of an approximation for the optimal transport problem and mentioned the application on graph comparison without going into details.
In the introduction all papers mentioned the application as similarity measure. [3] and [2] in a more detailed way. [3] was fixed more on the idea of obtaining a kernel.
[3] and [2] both aimed to use the Wasserstein distance to obtain a measure for graph similarity. In the paper [2] the Wasserstein distance was introduced only shortly compared to [3]. Here the authors gave a more visual and understandable explaination about the idea to use the Wasserstein distance for measuring graph similarity.
The key argument to simplify the computation of the Wasserstein distance is based for both papers on the way how they transformed the graphs. While [2] always exploits the properties of the normal distribution [3] could simplify the computation by dealing with finite node embeddings.
If we compare the experiment setup,[1] only used real-world-data. [3] only articial data while [2] uses both in more variation compared to the other papers. While [2] focused in their experiments to convince about the capability of their framework to recognize graph structures, [2] furthermore showed how the computed transportation plan looks like an not only how the costs of the plan is used in the graph comparison scheme. The experiments in [1] only focused to convince the reader about the efficiency of their algorithm compared to other ones.
The paper could maybe be connected by using the approximation algorithm from [1] instead to exactly solve the optimal transport problem in [3] and instead of the stochastic gradient descent approach of [2]. This could yield to a improvement of the running time as well. It would be interesting to see how the graph classification experiments of [2] would behave working with an approximation algorithm and on different graph transformation. One as probability distribution by [2] approach and one by the graph embedding scheme from [3].

References

[1] Nathaniel Lahn, Deepika Mulchandani, and Sharath Raghvendra. A graph theoretic additive approximation of optimal transport. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 13836–13846. Curran Associates, Inc., 2019.

[2] Hermina Petric Maretic, Mireille El Gheche, Giovanni Chierchia, and Pascal Frossard. Got: An optimal transport framework for graph comparison. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 13876–13887. Curran Associates, Inc., 2019.

[3] Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. Wasserstein weisfeiler-lehman graph kernels. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 6439–6449. Curran Associates, Inc., 2019.