Graph Classification Method Based on Wasserstein Distance

Wei Wu a, Guangmin Hu b, Fucai Yu c
School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

a weiwucz@163.com, b hgm@uestc.edu.cn, c fcyu@uestc.edu.cn

Abstract. Graph classification is a challenging problem, which attracts more and more attention. The key to solving this problem is based on what metric to compare graphs, that is, how to define graph similarity. Common graph classification methods include graph kernel, graph editing distance, graph embedding and so on. We introduce a new graph similarity metric, namely GRD (Geometric gRaph Distance). Our model GRD is composed of three sub-modules, which capture the differences between the graph structures from different aspects. Finally, the graph distances defined by the three modules are fused to define the similarity between graphs. Experiments show that GRD is superior to the baseline methods on the benchmark datasets.

Keywords: Ricci curvature, graph similarity, Wasserstein distance, scalar curvature.

1. Introduction

In recent years, graphs are becoming ubiquitous, with applications in the domains of social network analysis, genetic research, and image processing. The complex structure of graphs poses a challenge to graph data analysis.

With the rise of neural network, people pay more and more attention to using neural network model to analyze graph structure data. Using methods such as graph convolution [1] and random walk [2], the neural network model can extract the information of graphs or networks. Such approaches are called graph embedding, which are mainly used to obtain the vector representations of the nodes that hold network information. These node representations are commonly used for node classification and link prediction, but are rarely used for graph classification.

In this paper, we focus on the problem of similarity for graphs, which has important applications in many fields. Graph node embedding can achieve better results only by using the local information of the graph, while the calculation of graph similarity involves the local structure and whole structure of the graphs, which makes it difficult to deal with. The main contributions of this paper are: 1) graph embeddings are obtained by weighting node embeddings with the scalar curvature of nodes, and then the distance between graph embeddings are considered as one of the factors to measure the similarity of graphs; 2) The graphs are coarsened and the distributions of Ricci curvature of the edges of the original graphs and the coarsen graphs are calculated. The distances between these curvature distributions are used as a factor to measure the similarity of the graphs. 3) The optimal transport distance between graphs in [3] is modified by using scalar curvature, which is consider as a factor to measure the similarity of graphs.
2. Related work

2.1. Graph Embedding

Graph embedding is one of the methods to calculate graph similarity. In addition, methods such as graph kernels, graph edit distance and so on can also be used to do this. For example, [4] scores the mapping of subgraphs according to the scoring scheme obtained by comparing the vertex and edge attributes using the graph kernels. [5] proposes deep graph kernels that take advantage of the dependencies between substructures of graphs by learning their underlying representations. [6] measures quantitatively the difference between two graphs sharing the same nodes by introducing HIM distance, and then establishes the HIM kernel function derived from HIM distance. Through the support vector machine (SVM) algorithm, it can transition from network comparison to network classification. However, graph kernel methods can’t obtain the global properties of graphs since they focus on comparing the structures of subgraphs. [7] discusses the graph similarity search problem, that is, how to search the graph similar to the given query graph under the constraint of the minimum graph edit distance. But it should be noted that it is expensive to calculate graph similarity using graph edit distance. [8] proposed a new graph matching network model based on graph embedding. The model takes a pair of graphs as input, and calculates the similarity score between a pair of graphs by common reasoning through a new attention matching mechanism based on cross graphs. H-GCN [9] firstly aggregates the nodes with similar structure into supernodes repeatedly, and then refines the coarse graph into the original graph to restore the node representations. In this way, the receptive domain of the model is expanded and the global information can be effectively utilized. DEMO-Net [10] proposes a multi-task graph convolution, where each task is used to learn the node representations that hold degree-specific graph structure. Node2vec [11] uses biased second-order random walk to sample the neighborhood of the target nodes, and uses the information obtained by sampling to train the skip-gram model, so as to obtain node embeddings. LINE [12] obtains node embeddings by maintaining the first and second order proximity of the target nodes. Because most of the models based on neighborhood aggregation are shallow and lack of pooling mechanism, they cannot get enough global information. HARP [13] continuously coarses the input graph, so that the coarsened graph can approximately maintain the global structure of the original graph, and then obtains the node embeddings of the original graph by improving the node embeddings of the coarsened graph. However, the literature on network embedding mainly focuses on node classification and link prediction, with little reference to graph similarity or graph classification.

2.2. Wassertein distance and Ricci curvature

**Definition 1** [14]. Assume that \((U,d)\) be a metric space and \(\mu_1, \mu_2\) be the probability measures on this space. The Wasserstein distance between the probability measures is defined as:

\[
W(\mu_1, \mu_2) = \inf_{\pi \in \Pi(\mu_1, \mu_2)} \int d(u,v) \, d\pi(u,v)
\]

(1)

Where \(\Pi(\mu_1, \mu_2)\) is the set of measures on \(U \times U\) projecting to \(\mu_1\) and \(\mu_2\).

**Definition 2** [15,16]. Let \(G\) be an undirected weighted graph. Then for each node \(u\) in \(G\), the probability measure is defined as:

\[
m_u(v) = \begin{cases} 
\frac{w_{uv}}{d_u}, & \text{if } v \in \Gamma(u); \\
0, & \text{otherwise.}
\end{cases}
\]

(2)

Where \(\Gamma(u)\) is the set of nodes that are neighbors of \(u\) and \(d_u\) is \(\sum_{v \in \Gamma(u)} w_{uv}\).

**Definition 3** [15]. Let \(G\) be an undirected weighted graph, \(d\) is the graph distance metric and \(m_u, m_v\) be two probability measures on the metric space \((G,d)\). Then the Wasserstein distance between them is defined as
\[ W(m_u, m_v) = \inf_{\rho} \sum_{u \in \Gamma(u)} \sum_{v \in \Gamma(v)} d(u', v') \rho(u', v'), \]  

(3)

Where \( w_{u'u}, w_{v'v} \) are the weights of edges \((u, u'), (v, v')\), \( d_u = \sum_{v \in \Gamma(u)} w_{u'v}, d_v = \sum_{v \in \Gamma(v)} w_{v'v} \) and \( \rho \) should satisfy the following conditions:

\[ \sum_{u \in \Gamma(u)} \rho(u', v') = \frac{w_{u'u}}{d_u}, \sum_{v \in \Gamma(v)} \rho(u', v') = \frac{w_{v'v}}{d_u}. \]  

(4)

Following the definition of [14] and let \((u,v)\) be a edge of graph \( G \), then Ricci curvature of edge \((u,v)\) is defined as

\[ k(u, v) = 1 - W(m_u, m_v). \]  

(5)

Base the definition of Ricci curvature, we can define the scalar curvature of a node \( u \) as follows [17]:

\[ S(u) = \sum_{v \in \Gamma(u)} k(u, v), \]  

(6)

Where \( \Gamma(u) \) is the set consist of neighbor nodes of node \( u \).

3. Methods

Comparing the similarity of two graphs can also be regarded as calculating the distance between the two graphs. Of course, different models may adopt different distance measures. Our model is divided into three modules: (1) The first module is used to calculate the optimal transport distance between two graphs, namely Wasserstein distance; (2) In the second module, the node embeddings are weighted by scalar curvature to obtain the graph embeddings, which are then used to measure the distance between the two graphs; (3) The third module calculates the Ricci curvature distribution of the edges for the original graphs and the coarsened graphs, and then calculates the Wasserstein distance between the curvature distributions of the original graphs and the coarsened graphs. It should be noted that the first two modules can also operate on coarsened graphs, but this is not done in this paper, which will be considered in future work.

3.1. Optimal Transport Distance Between Graphs

Inspired by literature [3], we define the optimal transport distance between two graphs, namely Wasserstein distance. However, our definition is different from [3], which will be explained in detail below.

First, we use the variant of GCN [1], RCGCN [21], to learn node embedding, so each graph corresponds to a set composed of node embeddings. Given two graphs \( G_i = (V_i, E_i) \), where \( V_i, E_i \) are the node and edge sets of \( G_i \) and \( i \in \{1,2\} \). The Wasserstein distance between the two graphs \( G_1 \) and \( G_2 \) is defined as

\[ W_{ot}(G_1, G_2) = \min \sum_{i=1}^{n_i} \sum_{j=1}^{n_j} \rho_{ij} d(u_i, v_j) \]  

(7)

s.t. \( \sum_{j=1}^{n_j} \rho_{ij} = S_{G_1}(j), \forall j \in \{1, K, n_j\} \)

\[ \sum_{i=1}^{n_i} \rho_{ij} = S_{G_2}(i), \forall i \in \{1, K, n_i\}. \]  

(8)

Where \( S_{G_1}, S_{G_2} \) are node scalar curvature distributions of two graphs \( G_1 \) and \( G_2 \), and \( d \) is calculated according to node embeddings. The optimal transmission scheme expressed in Eq. 7 and 8 has the following meanings: 1) Firstly, we assume that the total mass distributed on the nodes of graph \( G_1 \) is 1, and the mass will be distributed according to the magnitude of the scalar curvatures of the nodes in
The greater the scalar curvature of a node, the more mass distributed to that node. 2) $\rho$ is the transport plan between the two graphs, and its elements $\rho_{ij}$ represents the quality of transfer from node $u_i$ in graph $G_1$ to node $v_j$ in graph $G_2$. 3) Then, according to the transmission plan $\rho$, we transfer the mass of the nodes in graph $G_1$ to the nodes in graph $G_2$, and finally the mass distributed on the nodes in graph $G_2$ should correspond to the magnitude of the scalar curvatures of the nodes in graph $G_2$.

3.2. Graph Embedding

The learning result of the graph neural network is usually the embedding of nodes, which will be used for downstream applications such as node classification and link prediction. At this point, the graph corresponds to a set of node embeddings. Assume that graph $G$ consists of nodes $u_1, K, u_n$. We denote the embedding of node $u_i$ as $\bar{H}_i$.

To measure the similarity between two graphs, we need to calculate the distance between the two graphs. For this reason, we first normalized the scalar curvatures of nodes in the graph, then regarded the scalar curvatures as a kind of mass distribution, and weighted average the node embeddings according to such distribution, so as to obtain the embedding of corresponding graph. Let the scalar curvature of node $u_i$ be $S(u_i)$ and the embedding of graph $G$ be $\bar{G}$. So the distance between the two graphs can be calculated according to the graph embedding as follows:

$$d_s(G_1,G_2) = \left\| \sum_{i} S_{G_1}(u_i)\bar{H}_i - \sum_{j} S_{G_2}(v_j)\bar{H}_j \right\|,$$

(9)

Which can be used as one of the factors to calculate the similarity between graphs.

3.3. Ricci Curvature Distributions of Graphs

Graph embedding methods usually focus on the local structure of the graph and extract information from it to obtain node embedding, which leads to insufficient preservation of global information. In order to preserve the global structure information better, we try to convert the whole graph into a quantity, namely the distribution of Ricci curvature of the edges of the graph. At the same time, we adopt the H-GCN coarsening strategy in [9] to continuously coarsen the granularity of the graph, so as to facilitate the analysis of the graph from different scales. Then we calculate the curvature distribution of the edges in the figure as follows: (1) Since the numerical range of the Ricci curvature of the edge is $[\text{-}2,1]$, we divide $[-2,1]$ into k bins. (2) If the Ricci curvature of an edge in the graph is in the range of a bin, the count corresponding to that bin is incremented by 1. (3) Then the count value of each bin is divided by the number of edges of the graph to obtain the Ricci curvature distribution of the edges of the graph. (4) Finally, the Ricci curvature distribution of the coarsened graph can be obtained by performing the above operations on each level of the coarsened graph.
Suppose that the graph is coarsened $M-1$ times, so for every graph, if we add up the original graph, we get $M$ graphs. Let $P_m(G)$ be the Ricci curvature distribution of the graph at level $m$. Thus, we can calculate the Wasserstein distance between the corresponding distributions of $m$-level graphs, namely $W_c^m(P_m(G_1), P_m(G_2))$. We regard this distance as the score of similarity between graphs. Since the granularities of coarsened graphs are relatively large, even if the scores are relatively high, they can only indicate that the general structures are relatively similar. Therefore, the score between coarsened graphs should be adjusted appropriately. Then the curvature distribution distance between the two graphs is defined as

$$W_c(G_1, G_2) = \sum_{m=0}^{M-1} \frac{1}{2^m} W_c^m(P_m(G_1), P_m(G_2)).$$  \hspace{1cm} (10)

### 3.4. Graph Similarity

With the three modules discussed above, we can now start to define graph similarity. According to Eq. 7-10, we define the distance between the two graphs as

$$d(G_1, G_2) = \alpha \cdot W_c(G_1, G_2) + \beta \cdot d_\delta(G_1, G_2) + \gamma \cdot W_c(G_1, G_2)$$  \hspace{1cm} (11)

Considering the fact that the greater the distance between two graphs, the smaller the similarity, the similarity of two graphs is defined as

$$Sim(G_1, G_2) = e^{-d(G_1, G_2)}.$$  \hspace{1cm} (12)

### 4. Experiment

Next we will first describe the datasets we used for the experiment. Then we will give baselines which are used to compare with our method. Finally, we will report the performance comparison between our approach and the baseline methods.

#### 4.1. Datasets

We run our model on three datasets: MUTAG, PTC-MR, ENZYMES. The MUTAG dataset consists of 188 nitro compounds. Each compound is treated as a node in the graph and labels are generated based on whether the compounds have the mutagenic effect on a particular bacterium. The PTC-MR dataset composed of 344 organic molecules that were labeled according to their carcinogenicity. ENZYMES,
which consisting of 600 enzymes, is a dataset of protein tertiary structures. Dataset statistics are listed in Table 1.

### Table 1. Dataset statics

| Dataset     | MUTAG | PTC-MR | ENZYMES |
|-------------|-------|--------|---------|
| average nodes | 17.93 | 25.56  | 32.63   |
| average edges | 19.79 | 25.96  | 62.14   |
| node attributes | ×     | ×      | √       |
| node labels   | 7     | 19     | 3       |
| graphs        | 188   | 344    | 600     |
| classes       | 2     | 2      | 6       |

The row in Table 1 named average nodes corresponds to the average number of nodes of all graphs in the dataset, while the row in Table 1 called average edges corresponds to the average number of edges of all graphs in the dataset. The row in Table 1 named node attributes indicates whether the graph in the dataset has node attributes. The last three rows in Table 1 correspond to the number of node labels, the number of graphs and the number of categories of graphs respectively.

### 4.2. Baseline Methods

We use the graph classification performance to evaluate our model. To do this, we compare our approach with the baseline approach described below.

- **DCNN** [18]: It introduces the "diffusion-convolution" operation, which constructs the node representations by scanning the diffusion process of each node in the input of the graph structure, and then classifies the graphs by taking the average activation value of the node representations.

- **DeepWL** [19]: It uses the heat wavelet diffusion model to represent the network neighborhood of each node through low dimensional embedding. It proves that even though these nodes may be located in different parts of the network, as long as the nodes have similar network neighborhoods, they will have similar GraphWave embeddings.

- **PATCHY-SAN** [20]: It converts the data of a graph structure into a structure that CNN can process efficiently, based on which the graph representation can be obtained.

### 4.3. Results

We used three baselines mentioned above to evaluate the graph classification performance of our model GRD.

### Table 2. Classification accuracy

| Dataset       | MUTAG | PTC-MR | ENZYMES |
|---------------|-------|--------|---------|
| DCNN [18]     | 0.670 | 0.572  | 0.160   |
| DeepWL [19]   | 0.733 | 0.537  | 0.210   |
| PATCHY-SAN [20] | 0.795 | 0.568  | 0.170   |
| GRD           | 0.856 | 0.613  | 0.243   |

The experimental results of all methods on the benchmark dataset are shown in Table 2. The results of the baseline methods are derived from [10]. Results indicate that GRD achieves the best performance on the three benchmark datasets compared to baselines.

### 5. Conclusion

We introduce a new graph similarity model, namely GRD. The model captures the differences between graph structures from three aspects. Firstly, starting from the problem of mass transfer between graphs based on scalar curvature, a new optimal transport distance between two graphs is defined. Secondly,
Graph embedding is obtained by weighting node embeddings through scalar curvatures, and the distance between graphs is measured accordingly. Then the graph distance is measured by coarsing the graph structure and comparing the Ricci curvature distribution of the edges of the graphs at all levels. Finally, the above three elements are fused to define the similarity between graphs. Our future work will be to layer the first two modules of the model to further improve the model performance.

References

[1] T.N. Kipf, M. Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

[2] B. Perozzi, R. Al-Rfou, S. Skiena. Deepwalk: Online learning of social representations. Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 701–710, August 2014.

[3] G. Nikolentzos, P. Meladianos, and M. Vazirgiannis. Matching node embeddings for graph similarity. Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31, February 2017.

[4] N. Kriege, P. Mutzel. Subgraph matching kernels for attributed graphs. arXiv preprint arXiv:1206.6483, 2012.

[5] P. Yanardag, S. Vishwanathan. Deep graph kernels. Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1365–1374, 2015.

[6] G. Jurman, R. Visintainer, M. Filosi, S. Riccadonna, C. Furlanello. The HIM glocal metric and kernel for network comparison and classification. 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp 1–10, 2015.

[7] W. Zheng, L. Zou, X. Lian, D. Wang, D. Zhao. Graph similarity search with edit distance constraint in large graph databases. Proceedings of the 22nd ACM international conference on Information & Knowledge Management, pp 1595–1600, 2013.

[8] Y. Li, C. Gu, T. Dullien, Q. Vinyals, P. Kohli. Graph matching networks for learning the similarity of graph structured objects. International Conference on Machine Learning, pp 3835–3845, 2019.

[9] F. Hu, Y. Zhu, S. Wu, L. Wang, T. Tan. Hierarchical graph convolutional networks for semi-supervised node classification. arXiv preprint arXiv:1902.06667, 2019.

[10] J. Wu, J. He, J. Xu. Net: Degree-specific graph neural networks for node and graph classification. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 406–415, 2019.

[11] A. Grover, J. Leskovec. node2vec: Scalable feature learning for networks. Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp 855–864, 2016.

[12] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei. Line: Large-scale information network embedding. Proceedings of the 24th international conference on world wide web, pp. 1067–1077, 2015.

[13] H. Chen, B. Perozzi, Y. Hu, S. Skiena. Harp: Hierarchical representation learning for networks. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.

[14] Y. Ollivier. Ricci curvature of Markov chains on metric spaces. Journal of Functional Analysis, vol. 256, pp 810–864, 2009.

[15] J. Jost, S. Liu. Ollivier’s Ricci curvature, local clustering and curvature-dimension inequalities on graphs. Discrete & Computational Geometry, vol. 51, pp 300–322, 2014.

[16] Y. Lin, L. Lu, S.T. Yau. Ricci curvature of graphs. Tohoku Mathematical Journal, Second Series, vol. 63, pp 605–627, 2011.

[17] M. Pouryahya, J. Mathews, A. Tannenbaum. Comparing three notions of discrete ricci curvature on biological networks. arXiv preprint arXiv:1712.02943, 2017.

[18] J. Atwood, D. Towsley. Diffusion-convolutional neural networks. arXiv preprint arXiv:1511.02136, 2015.
[19] C. Donnat, M. Zitnik, D. Hallac, J. Leskovec. Learning structural node embeddings via diffusion wavelets. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1320–1329, 2018.

[20] M. Niepert, M. Ahmed, K. Kutzkov. Learning convolutional neural networks for graphs. International conference on machine learning, pp 2014–2023, 2016.

[21] W. Wu, G. Hu, F. Yu. Ricci Curvature-Based Semi-Supervised Learning on an Attributed Network. Entropy, vol. 23, 2021.