Invariant embedding for graph classification

Alexis Galland [*1] **Marc Lelarge** [*1]

Abstract

Learning on graphs requires a graph feature representation able to discriminate among different graphs while being amenable to fast computation. The graph isomorphism problem tells us that no fast representation of graphs is known if we require the representation to be both invariant to nodes permutation and able to discriminate two non-isomorphic graphs. Most graph representations explored so far require to be invariant. We explore new graph representations by relaxing this constraint. We present a generic embedding of graphs relying on spectral graph theory called Invariant Graph Embedding (IGE). We show that for a large family of graphs, our embedding is still invariant. To evaluate the quality and utility of our IGE, we apply them to the graph classification problem and show that IGE reaches the state-of-the-art on benchmark datasets.

1. Introduction

Many scientific fields study data with an underlying graph or manifold structure such as social networks, sensor networks, biomedical knowledge graphs. The need for new algorithms and neural network architectures that can accommodate these non-Euclidean structures is becoming increasingly clear. In parallel, there is a growing interest in how we can leverage insights from these domains to incorporate new kinds of relational and non-Euclidean inductive biases into deep learning. Recent years have seen a surge in research on these problems, see the recent surveys (Goyal & Ferrara, 2018) and (Battaglia et al., 2018).

Graph classification is an important problem with practical applications in a diverse set of fields: bioinformatics, chemoinformatics or social networks. To solve this problem, one needs to compute graph features that help discriminate between graphs of different classes. Methods to learn representations that encode structure information about the graph can be categorized to be graph-kernel based or graph neural network (GNN) based. Interestingly, in both cases, best perfoming methods are closely related to the Weisfeiler-Lehman (WL) graph isomorphism test (Weisfeiler & Lehman, 1968), see (Shervashidze et al., 2011) for its connection with graph kernel and (Xu et al., 2018) for its connection with GNN. The WL test iteratively updates a given node's feature vector by aggregating feature vectors of its neighbors. (Babai et al., 1982) proposes an alternative algorithm for the graph isomorphism problem based on spectral graph theory. Our aim in this paper is to demonstrate the relevance of ideas from spectral graph theory (Spielman, 2007) to the graph representation learning problem. We introduce a novel and powerful graph feature representation called Invariant Graph Embedding (IGE). We show that IGE is invariant for a large family of graphs and also give intuition on why IGE has a great discrimminative power by explicitly showing the steps in its construction where some graph information is lost. We demonstrate the excellent performances of our IGE on graph classification tasks.

2. Related Work

Graph classification techniques can be grouped into different categories. Graph kernel algorithms define a distance between graphs in order to compute the similarity of two graphs and classify them using this metric. Among these techniques we list the graphlets kernel (Shervashidze et al., 2009) that computes a list of subgraphs and counts the aparitions of these subgraphs in each network of our dataset. There exist algorithms that define different distances between graphs such as Random walk kernel (Gärtner et al., 2003), shortest path kernels (Borgwardt & Kriegel, 2005), Weisfeiler Lehman subtree kernel (Shervashidze et al., 2009) or deep graph kernels (Yanardag & Vishwanathan, 2015). Other articles are based on recent works in the field of signal processing and define convolutional neural networks on graphs. Convolutions can be defined in many ways, with Graph Fourier Transform and spectral methods (Defferrard et al., 2016; Henaff et al., 2015; Hammond et al., 2011; Kipf & Welling, 2016), or by defining aggregation processes on the features of the neighboors of each nodes (Scarselli et al., 2009; Atwood & Towsley, 2016; Niepert et al., 2016) and

[*] Equal contribution [1] Team Dyogene, Inria, Paris, France. Correspondence to: Alexis Galland <alexis.galland@gmail.com>, Marc Lelarge <marc.lelarge@ens.fr>.

Presented at the ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data Copyright 2019 by the author(s).

other compute graph features without training using a generalization of scattering transform to graphs, initially defined on images (Gama et al., 2018). A fundamental issue when using neural networks defined on graphs is that they are dependant on the number of nodes in the graph and thus their use is not straigh-forward for a graph classification task. A way to cope with this issue would be to add nodes that are not linked to the graph to have a corpus of graphs that are all of the same size. Another way is to define a parametrization of the neural network that is independant with the number of nodes of the graph (Bruna & Li, 2017).

3. Invariant graph embedding

3.1. Notations

We consider connected undirected graphs without self-loops, $G = (V, E)$, where V is the vertex set and E the edge set. The size of the graph is the number of nodes and is denoted by $n = |V|$. We denote by A its adjacency matrix. In the absence of edge weights, this is a binary, symmetric matrix, with $A_{ij} = 1$ if and only if there is an edge between nodes i and j. In the presence of edge weights, A_{ij} is the weight of the edge between nodes i and j, if any, and is equal to 0 otherwise. We will mostly deal with unweighted graphs but our analysis carries over provided that all edge weights are positive.

Let $d = A1$, where 1 is the n-dimensional vector of ones. The components $d(1), \dots, d(n)$ of the vector d are equal to the actual degrees in the absence of edge weights and to the total weights of incident edges otherwise. Let $D = \mathrm{diag}(d)$ be the diagonal matrix of these generalized degrees. The Laplacian matrix is defined by $L = D - A$. We will also use the matrix $P = D^{-1}A$ which is the transistion matrix for the random walk on the graph G.

Recall that two graphs $G = (V, E)$ and $H = (V, F)$ are isomorphic if there exists a permutation π of V such that $(a, b) \in E \Leftrightarrow (\pi(a), \pi(b)) \in F$. We can express this relation in terms of matrices associated with the graphs.

Every permutation may be realized by a permutation matrix. For the permutation π, this is the matrix Π with entries given by

$$\Pi(a, b) = \begin{cases} 1 & \text{if } \pi(a) = b, \\ 0 & \text{otherwise.} \end{cases}$$

For a vector x, we see that $(\Pi x)(a) = x(\pi(a))$. Let A be the adjacency matrix of G and let B be the adjacency matrix of H. We see that G and H are isomorphic if and only if there exists a permutation matrix Π such that $\Pi A \Pi^T = B$. Note that the same relation can be written for the respective Laplacians. In this case, we will write $\pi(G) = H$.

A graph embedding (or graph feature) is a function $\mathcal{F}$ mapping graphs to vectors in $\mathbb{R}^d$, where d is called the dimension of the embedding. A graph embedding is invariant if for any two isomorphic graphs G and H, we have $\mathcal{F}(G) = \mathcal{F}(H)$.

From an embedding of nodes, it is straightforward to create a graph embedding. We need only to deal with the fact that graphs can have different sizes. A very simple choice is just to sum the nodes' embeddings. Now, if the embedding of nodes $\mathcal{E}$ is equivariant, this will produce an invariant graph embedding $\mathcal{F}$, but the invariance property would be preserved if instead of the sum, we apply any symmetric function as defined below.

Definition 1. *A function $f : \mathcal{S}^n \to \mathbb{R}^m$ is symmetric if for any permutation σ and any $x \in \mathcal{S}^n$, we have $f(x(1), \dots, x(n)) = f(x(\sigma(1)), \dots, x(\sigma(n)))$.*

This generic construction of graph embedding from nodes'embeddings will be used repeatedly in the sequel.

3.2. Invariant embedding from eigenvalues

W define our first invariant embedding. Recall that the Laplacian of a graph of size n is positive semi-definite with eigenvalues $\lambda_1 = 0 < \lambda_2 \leq \lambda_3 \leq \dots \leq \lambda_n$. Note that $\lambda_2 > 0$ because we assumed that the graph is connected. For a fixed k_1, we define $\mathcal{F}_1(G) = (\lambda_2, \dots, \lambda_{k_1+1})$ if $n \geq k_1 + 1$ and if $n \leq k_1$, then $\mathcal{F}_1(G) = (0, \dots, 0, \lambda_2, \dots, \lambda_n)$ where this last vector is completed by zeros to reach size k_1. Note that adding zeros 'on the left preserves the ordering of the λ_i's. It is clear that this embedding is invariant and as noted in (de Lara & Pineau, 2018) it already has a good discriminative power on real graphs. However, there are still many pairs of graphs that are non-isomorphic but which have the same eigenvalues and we will now improve our embedding to deal with those.

3.3. Space embedding

We now construct an embedding capturing the property of a random walk on the graph. Remember that P^k_{ij} is the probability for a random walk started in node i to be in node j after k steps. If we have features F available on nodes like atoms encoded in a one-hot vector for biochemical datasets, we consider $P^k F = (x^k_1, ..., x^k_n)$ which is, for each node, the aggregation of the features of its k-hope neighbors. Clearly $P^k F$ is an equivariant embedding of nodes so that for any symmetric fucntion $f : \mathbb{R}^n \to \mathbb{R}^m$, $f(x^k(1), \dots, x^k(n))$ is an invariant feature of the graph. If we don't have features available on nodes, we use $F = (d_1, ..., d_n)$ the vector of node degrees.

We now specify the symmetric function f that we take: for a graph of size n, we define

$$f(x_1, \dots, x_n) = hist(x_1, \dots, x_n; t), \tag{1}$$

where $hist$ is the histogram function and t is the number of bins of the histogram. Note that $hist(.;t)$ is a symmetric function for any t and it characrerizes the multiset $\{x_1, \dots, x_n\}$.

In summary, our second graph embedding $\mathcal{F}_2(G) \in \mathbb{R}^{k_2 t_2}$ is obtained by concatenating the features defined in (1) with number of bins t_2 and applied to the vector $P^k F$ for values of $k \in \{1, \dots, k_2\}$ for a fixed parameter k_2.

4. Spectral theory for graph embedding

So far, we constructed invariant graph embeddings, so that two isomorphic graphs will produce exactly the same embeddings. We will now relax this constraint. In this section, we define our spectral embedding and show that it is still invariant for graphs in which all eigenvalues have multiplicity 1.

4.1. Invariant graph theory

Recall that $L = D - A$ is the Laplacian of the graph and the spectral theorem yields $L = U\Lambda U^T$, where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ is the diagonal matrix of eigenvalues of L, with $0 = \lambda_1 < \lambda_2 \leq \dots \leq \lambda_n$, and $U = (u_1, \dots, u_n)$ is the matrix of corresponding eigenvectors, with $U^T U = I$ and $u_1 = 1/\sqrt{n}$.

We now define a classical spectral embedding for nodes. Let $X = \sqrt{\Lambda^+} U^T$, where $\Lambda^+ = \text{diag}(0, 1/\lambda_2, \dots, 1/\lambda_n)$ denotes the pseudo-inverse of Λ. The columns $x(1), \dots, x(n)$ of the matrix X define an embedding of the nodes in $\mathbb{R}^n$, each dimension corresponding to an eigenvector of the Laplacian. Note that the first component of each vector $x(1), \dots, x(n)$ is equal to 0, reflecting the fact that the first eigenvector u_1 of L is not informative. Moreover since $Xu_1 = 0$, the centroid of the n vectors is the origin:

$$\sum_{i=1}^{n} x(i) = 0. \tag{2}$$

The Gram matix of X is the pseudo-inverse of the Laplacian:

$$X^T X = U\Lambda^+ U^T = L^+.$$

In particular, we can recover the Laplacian matrix by taking the pseudo-inverse of $X^T X$ and hence the adjacency matrix and the graph itself. In words, the graph is completly encoded in the nodes' embedding X, i.e. without any loss of information.

We now give a random walk interpretation of this embedding. We consider the random walk with transition rate A_{ij} from node i to node j: the walker stays at node i an exponential time with parameter d_i, then moves from node i to node j with probability $P_{ij} = A_{ij}/d_i$. This defines a continuous-time Markov chain with generator matrix $-L$ and uniform stationary distribution. The sequence of nodes visited forms a discrete-time Markov chain with transition matrix P. Let H_{ij} be the mean hitting time of node j from node i. Observe that $H_{ii} = 0$. The following results are standard, see (Lovász et al., 1993), we follow the notations in (Bonald et al., 2018): $H_{ij} = n(x(j) - x(i))^T x(j)$, hence the mean commute time between nodes i and j is:

$$C_{ij} = H_{ij} + H_{ji} = n\|x(i) - x(j)\|^2.$$

Hence, the geometry of the nodes'embedding $x(i)$ is related to the geometry of the graph. Indeed, the matrix of commute times characterizes the graph:

Proposition 1. *We can reconstruct the adjacency matrix from the matrix of commute times.*

Proof. Since $C_{ij}/n = \|x(i)\|^2 - 2x(i)^T x(j) + \|x(j)\|^2$, we can recover the matrix $X^T X = L^+$ from C if we know the $\|x(i)\|^2$'s. But thanks to (2), we have $\sum_j C_{ij}/n = n\|x(i)\|^2 + \sum_j \|x(j)\|^2$ and $\sum_{ij} C_{ij}/n^2 = 2\sum_j \|x(j)\|^2$, hence the claim follows. □

We should stress that also all the information about the graph is contained in the relative geometry of the $x(i)$'s, their intrinsic values also carries relevant information. Here, we state Fiedler's nodal domain theorem (Fiedler, 1975): for any $k \geq 2$, let $W_k = \{i \in V, x_k(i) \geq 0\}$. Then the graph induced by G on W_k has at most $k - 1$ connected components. Together with (2), this implies that for low values of k, nodes i with non-negative entries $x_k(i)$ tends to be well-connected.

4.2. From commute times to graph embedding

As shown by Proposition 1, the commute times contain all the information about the graph. In order to incorporate it to our embedding, we proceed in a very straightforward manner. Instead of the euclidean distances $\|x(i) - x(j)\|^2$, we compute the dot products $x(i)^T x(j)$ for $1 \leq i, j \leq n$. Then, we 'flatten' this matrix to obtain a vector of size n^2 and pass this vector through an histogram.

Formally, our embedding is defined by:

$$\mathcal{F}_3(G) = hist((x(i)^T x(k))_{1 \leq i,k \leq n}; t_3) \in \mathbb{R}^{t_3}, \tag{3}$$

which is clearly an invariant embedding of the graph. Note that even if we can reconstruct the original graph from the matrix $(x(i)^T x(k))_{1 \leq i,k \leq n}$, our embedding $\mathcal{F}_3$ defined in (3) will not allow to reconstruct the graph.

5. Invariant graph embedding (IGE)

5.1. Handcrafted embeddings

So far, we defined three different graph embeddings: $\mathcal{F}_1(G) \in \mathbb{R}^{k_1}$ defined in Section 3.2; $\mathcal{F}_2(G) \in \mathbb{R}^{k_2 t_2}$

defined in Section 3.3 and $\mathcal{F}_3(G) \in \mathbb{R}^{t_3}$ defined in Section 4.2. Our final graph embedding is the concatenation of these embeddings for fixed parameters k_1, k_2 and t_2, t_3: $\mathcal{F}_{IGE}(G) = (\mathcal{F}_1(G), \mathcal{F}_2(G), \mathcal{F}_3(G))$ which is a vector of size $k_1 + t_2k_2 + t_3$, independent of the size of the graph.

All these embeddings are invariant graph embeddings. But two graphs that are not identical can produce an identical embedding $\mathcal{F}_{IGE}$. Finding the non-isomorphic graphs for which our embedding is not discriminative is out of the scope of this paper. In Section 6, we will assess the discriminative power of our embedding on benchmark datasets for graph classification.

6. Experiments and results

Datasets: We choose a wide variety of benchmark datasets for graph classification to evaluate our model. The datasets can be separated in two classes. 6 bioinformatics datasets: NCI1, NCI109, PROTEINS, MUTAG, PTC, D&D. And 5 social network datasets: REDDIT-BINARY and REDDIT-MULTI, IMDB-BINARY and IMDB-MULTI and COLLAB. In the 6 bioinformatics datasets, graphs represent chemical compounds. Nodes are atoms and edges represent connections between two atoms. The task is to discriminate molecules active against a cancer from those that are inactive against that same cancer. The social network datasets are composed of ego-networks the labels of the graphs are the nature of the entity from which we have generated the ego-network. More details can be found in (Yanardag & Vishwanathan, 2015).

Experimental setup: After having computed the features for each graph, we use a Random Forest for the classification. We compared the results with an SVM and we got slightly better results with a random foretst. We perform cross-validation by splitting each datasets in 10 folds and we use a grid search to fixe the parameters k_1, k_2, t_2 and t_3 respectively chosen from the sets $\{5, 10, 20, 50\}$, $\{1, 2, 3, 4, 5\}$, $\{5, 11, 21, 31\}$ and $\{5, 11, 21, 31\}$. We report the mean accuracy and the standard deviation over the 10 folds. We compare our method with 3 state-of-art algorithms: Family of Graph Spectral Distances (Verma & Zhang, 2017), Diffusion CNNs (DCNN) (Atwood & Towsley, 2016), Deep Graph Kernel (DGK) (Yanardag & Vishwanathan, 2015). For those algorithms we use the same setup as in (Verma & Zhang, 2017).

Results: From the results of Table 1 we can observe that IGE outperforms the other algorithms on all biochemichal datasets except for MUTAG dataset. We can note that MUTAG is a very small dataset. Thus the standard deviation is high because the size of the validation set is very small (19 graphs) when we split the dataset in 10 folds. From Table 2, IGE is challenging with all other algorithms on social network datasets and outperforms FSGD on both REDDIT datasets.

Dataset	MUTAG	PTC	PROTEINS	NCI1	NCI109	D&D
Max	28	109	620	111	111	5748
Avg	17.93	25.56	39.06	29.87	29.68	284.32
#Graphs	188	344	1113	4110	4127	1178
DCNN	66.51	55.79	65.22	63.10	60.67	-
DGK	86.17	59.88	71.69	64.40	67.14	72.95
FSGD	**92.12**	62.80	73.42	79.80	78.84	77.10
IGE	90.01 ± 6.6	**64.54 ± 5.8**	**74.21 ± 3.1**	**81.46 ± 0.9**	**79.36 ± 1.0**	**78.86 ± 3.2**

Table 1. Classification accuracy on bioinformatics datasets

Dataset	IMDB-B	IMDB-M	REDDIT-B	REDDIT-M	COLLAB
Avg	19.77	13.00	429.63	508.52	74.49
#Graphs	1000	1500	2000	4999	5000
DGK	66.96	44.55	78.04	41.27	73.09
FSGD	**73.62**	**52.41**	86.50	47.76	**80.02**
IGE	**74.10 ± 3.2**	48.87 ± 2.1	**89.12 ± 2.2**	**50.92 ± 2.4**	78.95 ± 2.0

Table 2. Classification accuracy on social datasets

7. Conclusion

We presented a novel and simple method to handcraft invariant features for a task of graph classification based on graph spectral decomposition. We showed that these features were relevant for a graph classification task because they are easy to compute, invariant by node permutation and contain sufficient information of the structure of the graph.
In our future work we plan to develop deeper architectures based on these features and use the Invariant Graph Embedding to initialize the neural network in order to achieve better classification.

References

Atwood, J. and Towsley, D. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1993–2001, 2016.

Babai, L. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 684–697. ACM, 2016.

Babai, L., Grigoryev, D. Y., and Mount, D. M. Isomorphism of graphs with bounded eigenvalue multiplicity. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pp. 310–324. ACM, 1982.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Bonald, T., Hollocou, A., and Lelarge, M. Weighted spectral embedding of graphs. *arXiv preprint arXiv:1809.11115*, 2018.

Borgwardt, K. M. and Kriegel, H.-P. Shortest-path kernels on graphs. In *Data Mining, Fifth IEEE International Conference on*, pp. 8–pp. IEEE, 2005.

Bruna, J. and Li, X. Community detection with graph neural networks. *arXiv preprint arXiv:1705.08415*, 2017.

de Lara, N. and Pineau, E. A simple baseline algorithm for graph classification. *arXiv preprint arXiv:1810.09155*, 2018.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

Fiedler, M. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25(4):619–633, 1975.

Gama, F., Ribeiro, A., and Bruna, J. Diffusion scattering transforms on graphs. *arXiv preprint arXiv:1806.08829*, 2018.

Gärtner, T., Flach, P., and Wrobel, S. On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines*, pp. 129–143. Springer, 2003.

Goyal, P. and Ferrara, E. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

Hammond, D. K., Vandergheynst, P., and Gribonval, R. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

Helfgott, H. A. Isomorphismes de graphes en temps quasi-polynomial (d'apr\es babai et luks, weisfeiler-leman...). *arXiv preprint arXiv:1701.04372*, 2017.

Henaff, M., Bruna, J., and LeCun, Y. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

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

Lovász, L. et al. Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty*, 2(1):1–46, 1993.

Niepert, M., Ahmed, M., and Kutzkov, K. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pp. 2014–2023, 2016.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., and Borgwardt, K. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, pp. 488–495, 2009.

Shervashidze, N., Schweitzer, P., Leeuwen, E. J. v., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

Spielman, D. A. Spectral graph theory and its applications. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pp. 29–38. IEEE, 2007.

Verma, S. and Zhang, Z.-L. Hunt for the unique, stable, sparse and fast feature learning on graphs. In *Advances in Neural Information Processing Systems*, pp. 88–98, 2017.

Weisfeiler, B. and Lehman, A. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9): 12–16, 1968.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1365–1374. ACM, 2015.