DICTIONARY LEARNING ON GRAPH DATA WITH WEISFIELER-LEHMAN SUB-TREE KERNEL AND KSVD

Kaveen Liyanage[⋆]*, Reese Pearsall*[†]*, Clemente Izurieta*[†]*, Bradley M. Whitaker*[⋆]

[⋆]Electrical and Computer Engineering
[†] Gianforte School of Computing
Montana State University
Bozeman, MT, USA

ABSTRACT

Graph representation has gained wide popularity as a data representation method in many applications. Graph embedding methods convert graphs to a vector representation and are an important part of a data processing pipeline. In this paper, we utilize sparse dictionary learning techniques as a graph embedding solution. Sparse representation has notable applications in signal image processing. Inspired by the Graph2Vec algorithm, we aim to modify the Doc2Vec model training portion of the Graph2Vec by incorporating unsupervised dictionary learning. We investigate the viability of using the sparse dictionary learning technique KSVD for graph data. We train the dictionary on Weisfeiler-Lehman graph sub-tree kernel features. Furthermore, we use graph-based labeled data sets to compare classification results with several existing graph embedding methods. Findings show that using the learned sparse coefficients as features for a supervised machine learning algorithm provides on-par classification results when compared to other graph embedding methods.

Index Terms— Sparse representation, Dictionary Learning, Graph embedding, KSVD

1. INTRODUCTION

Graph data is usually highly dimensional and defined in a non-euclidean form. Hence, typical processing methods defined on euclidean spaces cannot be used on graph data. Graph embedding methods convert the graph data into a vector representation while trying to preserve original graph properties [1].Graph embedding methods can be classified as node embedding, edge embedding, hybrid embedding, and whole graph embedding. In literature, a distinction is made between graph representation learning and graph embedding [1, 2], where graph representation does not require the final vector to be low-dimensional. In this paper, we focus on whole graph embedding, where each entire graph is represented as a vector [3]. The vector representation can be used to compare graph similarity for important tasks, including classification and clustering. The main challenges in whole graph embedding are how to capture the properties of a whole graph and how to make a trade-off between expressiveness and efficiency [1]. Several methods have been proposed for whole graph embedding, including matrix factorization, deep learning, edge reconstruction, graph kernel, and generative models [1, 3].

Graph2Vec is a popular neural network-based architecture for graph embedding [4]. Some advantages of Graph2Vec are that the model is trained in an unsupervised manner, the learned model is task agnostic, the algorithm is data-driven, and resulting vectors capture structural equivalences. An overview of the implementation of the Grap2Vec is shown in Fig. 1, which consists of two procedures. First, a vocabulary of sub-tree structures is generated using a Weisfeiler-Lehman (WL) sub-tree kernel and a Doc2Vec model is trained on the selected vocabulary. Graph2Vec utilizes the non-linear WL kernel, which is shown to outperform other linear kernels [5]. WL kernel is used to rename the nodes using a hash value that represents a rooted sub-graph on the given node. These sets of node names are viewed as a set of words in a document. The techniques from the Natural language processing (NLP) domain are borrowed for learning an embedding. Doc2Vec is based on Word2Vec [6], in which a feed-forward neural network(NN) "SkipGram" model with negative sampling is used to learn a representation of word sequences [7]. Using the SkipGram model, the nodes with similar neighborhoods are embedded closer together [8]. The Graph2Vec is implemented in the "KarateClub" python package [9][1].

Some disadvantages of Graph2Vec are the nonlinearity of the learned embedding and the generated sub-tree structures. Due to the nonlinearity, it is difficult to identify which sub-tree structures are contributing to the similarities and differences among graphs. Hence, we propose a linear representa-

This research was conducted with the U.S. Department of Homeland Security (DHS) Science and Technology Directorate (S&T) under contract 70RSAT22CB0000005. Any opinions contained herein are those of the authors and do not necessarily reflect those of DHS S&T.

[1]https://karateclub.readthedocs.io/en/latest/

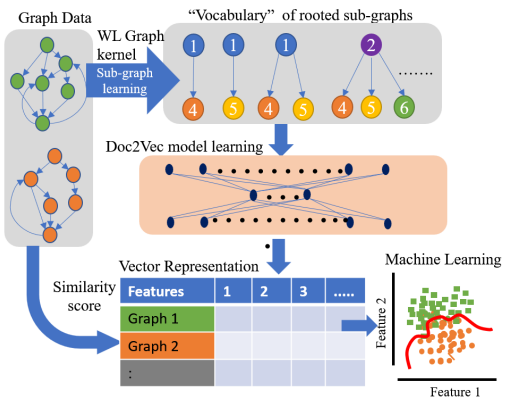

Fig. 1. Graph2Vec pipeline overview

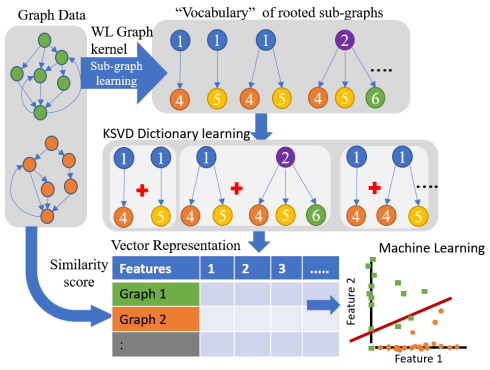

Fig. 2. Proposed WL+KSVD pipeline overview

tion model to replace the Doc2Vec NN architecture. Further, the SkipGram model is capable of embedding only a single node, rather than node combinations. In addition, the SkipGram model considers the neighborhood of the nodes, which depends on an arbitrary node numbering scheme that may not generalize between graphs in a given application.

Sparse representation is a technique used to learn a dictionary that lies in the original feature domain and calculate a representation using a linear combination of a few dictionary elements (atoms) [10]. The main advantages of using sparse representation are linearity and sparsity: the learned embedding consists of linear combinations of sub-tree structures; sparse representations allow using low-order classification models due to the low VC dimension [11]. Sparse representation was introduced in the signal and image processing domains and recently it has been utilized in graph-related processes. Several methods have been proposed to represent graph signals on a fixed graph topology with sparse representations with theoretical guarantees [12]. Recent work by Matsuo et al. [13] develops a method to represent different network topologies with sparse representation. However, their work is still limited by requiring graphs to be undirected and requiring all topologies to have the same number of nodes.

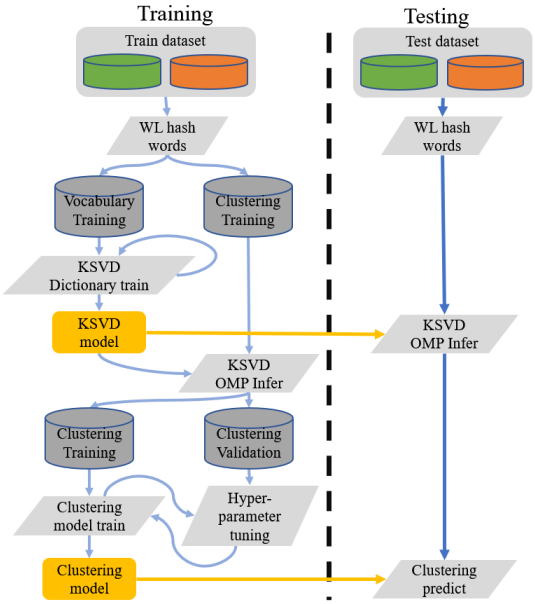

Fig. 3. Evaluation Workflow

To address the shortcomings in sparse vector-based graph representations, we introduce a framework to incorporate WL sub-tree kernel with sparse representation methods specifically aimed at machine learning classification tasks. Our framework allows sparse representation to be applied to graphs with different topologies and different numbers of nodes. In addition, the input graphs can be directed and can incorporate node features. An overview of the proposed WL+KSVD pipeline is shown in Fig. 2. The proposed method has the flexibility to swap different dictionary learning and graph kernel methods in the framework. The method is tested against several similar graph embedding methods with benchmark datasets. Finally, the python implementation of the framework and the experiments are currently available on Github[2].

2. METHODOLOGY

An overview of the workflow of the proposed method is shown in Fig. 3, where the training set is further divided in half into embedding training and classifier training to avoid overfitting.

2.1. Graph Notation and Definition

Let a graph be defined as $G = (V, E)$ which can be directed or undirected with unweighted edges, where node $v_i \in V$ and edge $e_{i,j} \in E$ connects v_i and v_j. Let the dataset be a set of M graphs with different topology and nodes $\mathbf{G} = [G_1, G_2, \ldots, G_M]$. Following the Graph2Vec algorithm, ff

[2]https://github.com/BMW-lab-MSU/WL-KSVD.git

node labels are not provided the nodes will be initialized with the degree of the node as its label. The degree of a node is a count of the number of edges the node receives and sends. We use the "NetworkX" python package as the graph data structure [3].

2.2. Weisfeiler-Lehman sub-tree kernel

WL sub-tree relabelling process (described in [5][4]) is used to relabel the nodes with a unique hash value for the rooted sub-tree structure. Note that the sub-tree structure learned is deterministic, so the same sub-tree structure in different graphs will have the same hash value. For each $G_i \in \mathbf{G}$, rooted sub-trees $sg^h_{i,j}$ are learned for each $v_j \in \mathbf{V}_i$, where i is the graph, j is the node and h is the WL rooted sub-tree depth. Now each graph is a set of hash words $G_i = [sg^h_{1,i}, sg^h_{2,i}, \ldots, sg^h_{l,i}]$, where l_i is the number of nodes in G_i.

2.3. Vocabulary Creation

Using the Doc2Vec implementation in Gensim python package [5] a raw vocabulary is created using the unique set of sub-tree hash words sg across all the training graphs [7]. If the raw vocabulary is too large it can be trimmed according to a trim rule. In this work, we trim the vocabulary by selecting the K highest frequency sub-tree hash words. Other possible trimming rules are the highest likelihood, highest prior, etc. Each graph G_i is then represented as the occurrences Y_i of the vocabulary elements, where $Y_i = [y_{i,1}, \ldots, y_{i,K}]$ and $y_{i,j}$ is the number of occurrences of vocabulary word j in graph i. Now the dataset can be represented as a collection of fixed-length vectors: $\mathbf{Y} = [Y_1, Y2, \ldots, Y_M] \in \mathbb{R}^{K \times M}$.

2.4. Sparse Representation

Let $\mathbf{Y} = [Y_1, Y_2, \ldots, Y_M] \in \mathbb{R}^{K \times M}$ be a set of M input signals with fixed length K. Sparse representation attempts to represent the input signal as a linear combination of elements $d_i \in \mathbb{R}^K$ in a dictionary $\mathbf{D} = [d_1, d_2, \ldots, d_N] \in \mathbb{R}^{K \times N}$ while limiting the number of atoms used to T (sparsity). The sparse coefficient vector $\alpha = [\alpha_1, \alpha_2, \ldots \alpha_M] \in \mathbb{R}^{N \times M}$ will be the sparse representation with $|\alpha_i|_0 \leq T$, where $|.|_0$ operator counts the number of non-zero elements in the vector. The general form of the sparse representation can be formulated as:

$$\underset{\mathbf{D},\alpha}{\operatorname{argmin}} ||\mathbf{Y} - \mathbf{D}\alpha||_2^2 + ||\alpha||_0. \quad (1)$$

This optimization problem is NP-hard, but an approximate solution can be provided using an iterative algorithm named K-means Singular Value Decomposition (KSVD) [14] [6]. First, a dictionary $\mathbf{D}$ is fixed and sparse vectors α are optimized using Orthogonal Matching Pursuit (OMP) [15]. Second, α is fixed and the dictionary $\mathbf{D}$ is updated with a generalized K-means algorithm. After many iterations, each graph is represented as a fixed-length *sparse* vector. In addition, using the trained dictionary, new graphs can be represented as sparse vectors. It should be noted that the KSVD is known to have convergence issues and newer methods have better performance and convergence [16]. Since this paper is mainly focused on the framework for using sparse coding for graph embedding. a simpler well-known method (KSVD) was chosen to demonstrate its viability.

3. EXPERIMENT

Several publicly available benchmark dataset were chosen to compare the embedding performance, namely MUTAG (MU), PTC, PROTEINS (PROT), NCI1, NCI109 [17][7]. The datasets were divided as $9:1$ for training and testing with random shuffling. Several graph embedding methods in KarateClub library are compared: Graph2Vec [4], GL2vec [18], and SF [19]. The considered embedding dimensions are $N = [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048]$. Default settings for each embedding method are used to avoid bias in training. In the proposed WL+KSVD method, the max size of learned vocabulary is chosen as $K <= 10000$ and the sparsity is chosen roughly as 10% of the dimensions, $T = ceil(N/10)$.

Several classification methods from the scikit-learn python package are used to evaluate the performances of the embedding methods [20][8]. The classifiers were used with default settings and without hyperparameter tuning for each method to avoid bias in training and for easy comparison. The classifiers and settings use the sci-kit learn example for easy recreation with 5 fold cross validation[9].

4. RESULTS

Figure 4 shows the two-dimensional (N=2) embedding of the MUTAG dataset with different graph embedding methods. (Note that using $N = 2$ dimensions does not result in optimal embeddings; this number is only chosen for demonstration and the complete set of results is available in Github repo.) The figure also shows several classifier methods evaluated on the embedded data. It can be seen that the sparse coding limits the data to a sparse domain in the final vector space.

Table 1 shows mean accuracy of the linear SVM clustering variation as a function of embedding dimension for each embedding method. It can be observed that the proposed method performs on par with Graph2Vec and GL2vec. SF

[3] https://networkx.org/

[4] https://github.com/benedekrozemberczki/karateclub/blob/master/karateclub/utils/treefeatures.py

[5] https://radimrehurek.com/gensim/

[6] https://github.com/nel215/ksvd

[7] https://chrsmrrs.github.io/datasets/docs/home/

[8] https://scikit-learn.org/

[9] https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

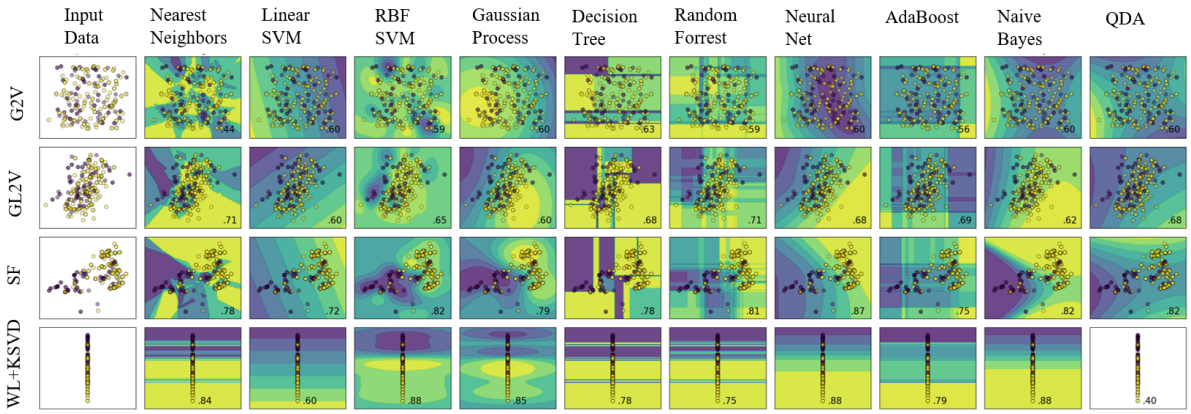

Fig. 4. Classification decision boundaries for MUTAG dataset with 2-dimensional embedding for four embedding algorithms (Graph2Vec, GL2V, SF, and WL+KSVD). Sparse structure is clearly seen in the WL+KSVD embedding (bottom row).

Table 1. Linear SVM validation accuracy for various embedding dimensions using the MUTAG dataset

N=	2	4	8	16	32	64	128	256	512	1024	2048
G2V	0.665	0.665	0.654	0.643	0.643	0.675	0.675	0.659	0.664	0.685	0.686
GL2V	0.670	0.676	0.718	0.734	0.750	0.766	0.760	0.733	0.765	0.749	0.766
SF	**0.724**	**0.788**	**0.824**	**0.840**	**0.835**	**0.835**	**0.835**	**0.835**	**0.835**	**0.835**	**0.835**
WL+KSVD	0.665	0.665	0.665	0.687	0.756	0.766	0.787	0.756	0.745	0.724	0.713

Table 2. Linear SVM accuracy with $N = 1024$ embedding

	MU	PTC	PROT	NCI1	NCI109
G2V	68.55 ±10.03	55.23 ±5.9	67.30 ±0.87	59.30 ±4.46	56.46 ±2.82
GL2V	74.92 ±7.8	52.04 ±6.5	69.09 ±1.38	64.52 ±1.99	62.98 ±2.97
SF	83.47 ±4.15	57.59 ±9.34	70.98 ±1.00	61.90 ±3.24	61.96 ±2.40
WL+ KSVD	72.38 ±3.20	54.36 ±2.31	64.60 ±2.00	64.16 ±2.19	62.93 ±0.24

methods have higher performance over all the dimensions for the linear classifier. Table 2 shows the linear SVM mean test accuracy and standard deviation (std) for the different datasets with an embedding dimension of 1024. The proposed WL+KSVD method has on-par performance with respect to others with relatively low std. It should be noted that the hyperparameters were not optimized for any method and executed with default settings. Hence the actual optimal results are better. However, this analysis provides a baseline comparison of the performance of the WL+KSVD method.

5. CONCLUSION

Sparse representation is a powerful tool in several signal-processing application domains because of its ability to extract inherent features. We aim to expand sparse representation tools into graph processing domains. The WL+KSVD framework is presented as an unsupervised whole graph embedding method. The input graphs can be directed or disconnected and may have node labels. Through benchmark datasets and several comparable graph embedding methods, it was shown that the proposed method has on-par or better performance in classification tasks. While the WL+KSVD embedding method is non-reversible and some information is lost in the vocabulary trimming process, this is not an issue for ML tasks like classification and clustering.

The framework is flexible in that the WL sub-tree kernel and the KSVD can be swapped with other graph kernels and dictionary learning methods. Discriminatory dictionary learning methods can be utilized to learn a more separable representation in a supervised manner. Some of the dictionary learning methods we plan to test are LC-KSVD [21] and frozen dictionary learning [22]. Further, we plan to use important graph sub-tree structures using feature ranking metrics. Lastly, the linear nature of the proposed method allows for the importance of sub-tree structures to be evaluated using dictionary mapping and dictionary utilization [23].

6. REFERENCES

[1] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowl-*

edge and Data Engineering, vol. 30, pp. 1616–1637, sep 2018.

[2] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy, "Machine learning on graphs: A model and comprehensive taxonomy," *Journal of Machine Learning Research*, vol. 23, no. 89, pp. 1–64, 2022.

[3] L. Maddalena, I. Manipur, M. Manzo, and M. R. Guarracino, "On whole-graph embedding techniques," in *Trends in Biomathematics: Chaos and Control in Epidemics, Ecosystems, and Cells*, pp. 115–131, Springer International Publishing, 2021.

[4] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs." 13th International Workshop on Mining and Learning with Graphs (MLGWorkshop 2017), 2017.

[5] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels.," *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[7] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning* (E. P. Xing and T. Jebara, eds.), vol. 32 of *Proceedings of Machine Learning Research*, (Bejing, China), pp. 1188–1196, PMLR, 22–24 Jun 2014.

[8] X. Rong, "word2vec parameter learning explained," *arXiv preprint arXiv:1411.2738*, 2014.

[9] B. Rozemberczki, O. Kiss, and R. Sarkar, "Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs," in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, p. 3125–3132, ACM, 2020.

[10] M. Elad, *Sparse and Redundant Representations*. Springer New York, 2010.

[11] T. Neylon, *Sparse Solutions for Linear Prediction Problems*. PhD thesis, New York University, USA, 2006. AAI3221982.

[12] Y. Yankelevsky and M. Elad, "Finding GEMS: Multiscale dictionaries for high-dimensional graph signals," *IEEE Transactions on Signal Processing*, vol. 67, pp. 1889–1901, apr 2019.

[13] R. Matsuo, R. Nakamura, and H. Ohsaki, "Sparse representation of network topology with k-SVD algorithm," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, IEEE, jul 2019.

[14] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, pp. 4311–4322, Nov. 2006.

[15] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, IEEE Comput. Soc. Press, 1993.

[16] C. Bao, H. Ji, Y. Quan, and Z. Shen, "Dictionary learning for sparse coding: Algorithms and convergence analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 1356–1369, jul 2016.

[17] C. Morris, G. Rattan, and P. Mutzel, "Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 21824–21840, Curran Associates, Inc., 2020.

[18] H. Chen and H. Koga, "GL2vec: Graph embedding enriched by line graphs with edge features," in *Neural Information Processing*, pp. 3–14, Springer International Publishing, 2019.

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

[20] L. Buitinck and et. el., "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.

[21] Z. Jiang, Z. Lin, and L. S. Davis, "Label consistent k-SVD: Learning a discriminative dictionary for recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 2651–2664, nov 2013.

[22] B. T. Carroll, B. M. Whitaker, W. Dayley, and D. V. Anderson, "Outlier learning via augmented frozen dictionaries," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, pp. 1207–1215, jun 2017.

[23] K. Liyanage and B. M. Whitaker, "Feature analysis in satellite image classification using LC-KSVD and frozen dictionary learning," in *2022 Intermountain Engineering, Technology and Computing (IETC)*, IEEE, may 2022.