

# SPARSE REPRESENTATION METHOD FOR WHOLE GRAPH EMBEDDING

by

Kaveen Gayasara Liyanage

A comprehensive exam submitted in partial fulfillment  
of the requirements for the degree

of

Doctor of Philosophy

in

Electrical Engineering

MONTANA STATE UNIVERSITY  
Bozeman, Montana


December 2022

©COPYRIGHT

by

Kaveen Gayasara Liyanage

2022

Creative Commons Attribution 4.0 International License 

## ACKNOWLEDGEMENTS

Part of the proposal is funded by the Department of Homeland Security Science and Technology Directorate under contract number 70RSAT22CB0000005.

## TABLE OF CONTENTS

1. INTRODUCTION .....	1
figure example .....	1
2. BACKGROUND.....	3
Graphs Embedding .....	3
Sparse Representation and Dictionary Learning .....	5
Feature Ranking.....	7
Hyperbolic Space.....	9
3. PRELIMINARY WORK .....	10
Task 1: Graph KSVD .....	10
Methodology .....	12
Graph Notation and Definition.....	13
Weisfeiler-Lehman sub-tree kernel .....	13
Vocabulary Creation.....	13
Sparse Representation.....	14
Task 2: Feature Ranking .....	14
Task 3: Hyperbolic graph embedding .....	17

## LIST OF FIGURES

Figure	Page
1.1 Example duck .....	2
2.1 Graph2Vec pipeline overview .....	5
3.1 Proposed WL+KSVD pipeline overview .....	11
3.2 Evaluation Workflow .....	12

## LIST OF ALGORITHMS

Algorithm

Page

## ABSTRACT

Sparse representation has gained popularity in the domains of signal and image processing domains.

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.

## INTRODUCTION

Sparse representation has gained

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.

Random references[1].

figure example



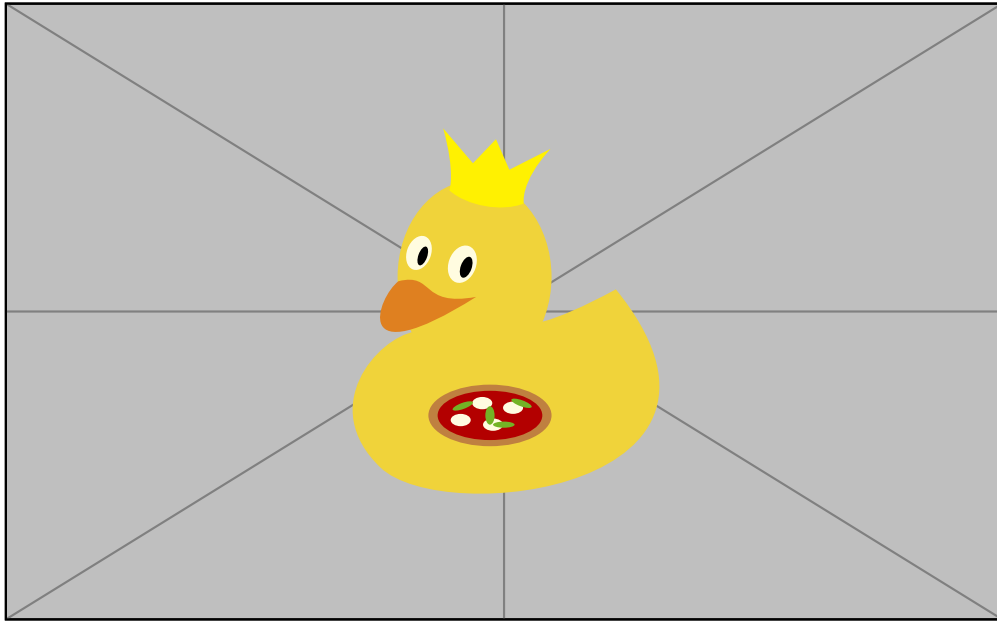


Figure 1.1: Example duck.

## BACKGROUND

### Graphs Embedding

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[2]. 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[2, 3], 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[4]. 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[2]. Several methods have been proposed for whole graph embedding, including matrix factorization, deep learning, edge reconstruction, graph kernel, and generative models[2, 4].

Graph2Vec is a popular neural network-based architecture for graph embedding[5]. 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. Graph2Vec utilizes the non-linear Weisfeiler-Lehman (WL) kernel, which is shown to outperform other linear kernels[6]. 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[7], in which a feed-forward neural network (NN) “SkipGram” model with negative sampling is used to learn a representation of word sequences[8]. Using the SkipGram model, the nodes with similar neighborhoods are embedded closer together[9]. The Graph2Vec is implemented in the “KarateClub” python package[10]<sup>1</sup>. An overview of the implementation of the Grap2Vec is shown in Fig.2.1, where a vocabulary of sub-tree structures is generated using a WL sub-tree kernel and a Doc2Vec model is trained on the selected vocabulary.

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 representation 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.

---

<sup>1</sup><https://karateclub.readthedocs.io/en/latest/>

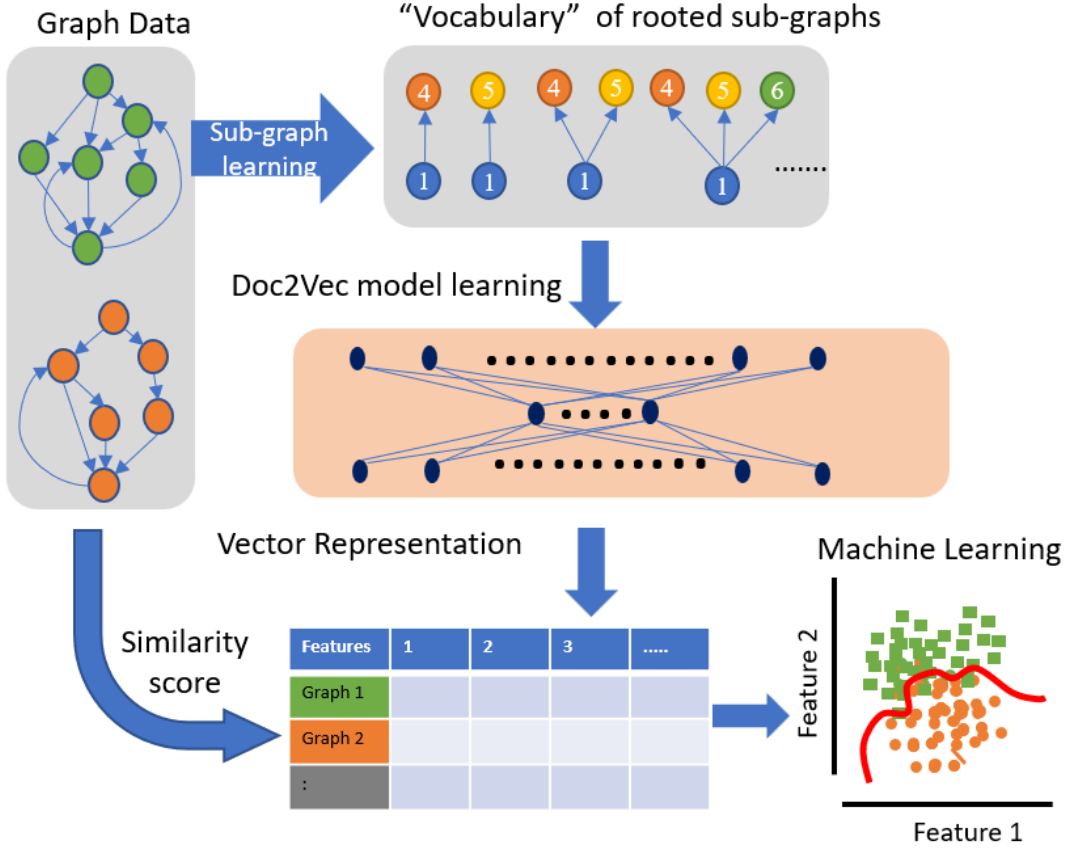


Figure 2.1: Graph2Vec pipeline overview

### Sparse Representation and Dictionary Learning

There has been a growing interest in the search for sparse representations of signals in recent years. In the field of computer vision it can be reasonably assumed that image patches do not populate or sample the whole input domain[11]. Sparse coding is a representation learning method which aims to find a sparse representation of an  $n$  dimensional input signal  $y_i \in \mathbb{R}^n$  in the form of a sparse linear combination, such that the reconstructed data is  $\tilde{y}_i = \alpha_{i,1}d_1 + \alpha_{i,2}d_2 + \dots + \alpha_{i,K}d_K$ . Where  $\alpha_i \in \mathbb{R}^K$  is the the sparse vector and  $d_i \in \mathbb{R}^n$  are the dictionary elements (atoms) of a Dictionary  $\mathbf{D}$ . Sparse representation algorithms

optimize (2.1) with a  $l_0$  regularization term:

$$\underset{D, \alpha}{\operatorname{argmin}} ||\mathbf{Y} - \mathbf{D}\alpha||_2^2 \text{ s.t. } \forall i, ||\alpha_i||_0 \leq S, \quad (2.1)$$

where  $\mathbf{Y} = [y_1, y_2, \dots, y_N] \in \mathbb{R}^{n \times N}$  denotes the  $N$  number of input signals,  $\mathbf{D} = [d_1, d_2, \dots, d_K] \in \mathbb{R}^{n \times K}$  is the learned dictionary of size  $K$ ,  $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_N] \in \mathbb{R}^{K \times N}$  is the sparse representation of the input signal, and  $S$  is the sparsity constraint of  $\alpha_i$  (maximum number of non-zero elements). Usually  $K > n$ , in which case the dictionary is called over-complete. If  $K = n$  the dictionary is called complete and if  $K < n$  it is called under-complete.

Equation (2.1) can be solved by alternating between the following two stages. First, sparse coding is to calculate  $\alpha$  with a fixed over-complete dictionary  $D$ . Second, dictionary learning is performed to update  $D$  with a fixed  $\alpha$ . K-means Singular Value Decomposition (K-SVD)[1, 12] has emerged as an effective and popular algorithm for sparse representation tasks. K-SVD first initializes a random dictionary. It then alternates between the two stages by utilizing Orthogonal Matching Pursuit (OMP)[13, 14] for the sparse coding and generalized k-means with Singular Value Decomposition (SVD) for the dictionary update. K-SVD efficiently learns an over-complete dictionary and has been effectively utilized for tasks including de-noising, restoration, and classification.

For classification tasks, in order to improve the performance a more discriminatory representation is required. Jiang *et al.*[15, 16] have presented a Label Consistent K-SVD (LC-KSVD) algorithm as an extension of the K-SVD framework, which is a supervised learning algorithm to learn a compact and discriminative dictionary. In LC-KSVD, class-specific dictionary elements are trained separately as an initialization and then combined to learn a discriminative dictionary. A label consistent constraint called “discriminative sparse-code error”, reconstruction error and classification error terms are combined to structure a unified objective function to optimize the discriminated dictionary. Due to the class

constraints in the sparse coding and dictionary update stages, the input data will be forced to be mapped to the dedicated dictionary atoms according to the label information. Consequently in the sparse dictionary domain, a majority of the input signals will be projected to a subspace belonging to a certain class. Hence, a lower order classifier can be trained for the classification.

Traditional dictionary learning models do not take into account the class imbalances of the training data. Hence the dictionary atoms can be biased towards the larger class. Therefore to address the class imbalances and the structure, a separate dictionary learning algorithm is also employed. Frozen dictionary learning modifies the dictionary learning process as a hierarchical structure to learn a dictionary that can effectively model imbalanced datasets[17]. In this algorithm, first, the dictionary learning step is carried out using the K-SVD algorithm on “normal” training data. Then the learned dictionary elements are frozen (held constant) and the dictionary is augmented with additional elements by dictionary elements is trained again on abnormalities. This process is repeated for all the remaining classes, by keeping the previously learned dictionaries frozen. The frozen elements of the dictionary represent the “normal” aspects of the data, hence the new elements (non-frozen) learn to represent the anomalous aspects of the data that are not present in the “normal” data. The frozen dictionary approach could be generally used and applied to the problems including data with or without abnormalities.

### Feature Ranking

Feature Ranking (FR) is an essential part of the machine learning pipeline to identify, reduce, remove, or craft features that benefits ML performance and reduce the cost of the operations. In general, FR methods evaluate features by looking at the amount of information they provide and ranking them accordingly so that the most relevant and complementary features can be used in ML training. There are three main categories of

methods for FR algorithms: Filter methods (FM), Wrapper methods (WM), and Embedded methods (EM). The FR methods can be further classified as *myopic* and *non-myopic*. Whereas *myopic* methods only evaluate the feature by itself, *non-myopic* methods take into the consideration of interrelationship between the features. Several surveys outline the current state-of-the-art in feature assessment techniques [18, 19, 20, 21].

In FM, intrinsic properties are evaluated to determine the relevance of the feature. FM methods are generally computationally inexpensive and do not depend on the ML model since the evaluation is done independently. In WM the feature ranking is tied to the ML model performance. This method is more computationally expensive than the FM as it needs to train multiple ML models to identify which features contribute more. WM is model- and task-specific and, as a result, gives better results. However, WM has to be performed again for a different task or a model. EM is similar to WM, however, it performs the feature selection while the ML model is trained. Therefore, time is saved by avoiding training multiple models by sacrificing performance compared to WM. EM is also model and task-specific.

The proposed SR-based methods can be considered as *non-myopic* and a hybrid of FM and EM concepts. To calculate the two metrics, dictionary learning has to be carried out (like EM), which is more computationally expensive than the typical FM. However, these metrics do not depend on any ML methods so these FR scores can be used in many applications, unlike EM methods. Furthermore, the proposed metrics do not even have to depend on any particular SR method. However, using a discriminatory dictionary learning and (semi-) supervised SR method would be able to improve the interpretation ability of the data. Since the atoms are a subspace of the original feature space when atoms are learned it takes into account the relationship between all the features hence, these metrics are *non-myopic*. Another main advantage is the ability to decompose the feature importance according to each class with supervised dictionary learning methods. Also, the learned dictionary is not

wasted as the dictionary and the calculated sparse coefficients can be used for the training of classifiers in the next stages of the machine learning pipeline.

Chang and Lin[22] conducted FR using the weights of the linear SVM. Compared with a variant of Fisher-score[23] (F-score) method, their method showed improved performance. However, it can be only used with a linear SVM. Jong, et al.[24], proposed an ensemble feature ranking (EFR) algorithm that aggregates results of multiple FR algorithms to gain higher performance.

Relieff[25] and mRMR[26] are two commonly used FR algorithms. Both the methods can be considered as *non-myopic* FM algorithms. Zhang et al.[27] implemented a novel SR-based feature assessment method called SRDA, where they employ both SR and information theory to identify dependencies and redundancies of the salient features. In this work, they evaluate the learned dictionary atoms (new features) for redundancy and complementary properties. In our work, we try to evaluate the original input features, not the derived sparse features. However, they provide some interesting frameworks for selecting candidate sparse features.

In recent years several deep learning methods have been proposed that achieve high accuracy for data sets. However, deep learning methods lack an intuitive relationship between the learned features and the input layer. Also, they require large computational resources for training and testing. It can be seen that almost all methods that have been used are some form of deep architecture. Usage of deep networks is popular due to the high performance and ability to learn features automatically[28]. We would urge readers to get familiarized with our previous work[29] for a detailed discussion about the advantages of the SR methods concerning deep learning methods.

### Hyperbolic Space



## PRELIMINARY WORK

Task 1: Graph KSVD

Sparse representation is a technique used to learn a dictionary that lies in the original feature domain and calculate a sparse representation using a linear combination of a few dictionary elements (atoms)[11]. 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[30]. Sparse representation was originally introduced in the signal and image processing domains, however 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[31]. Recent work by Matsuo et al.[32] 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.

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.3.1. 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<sup>1</sup>.

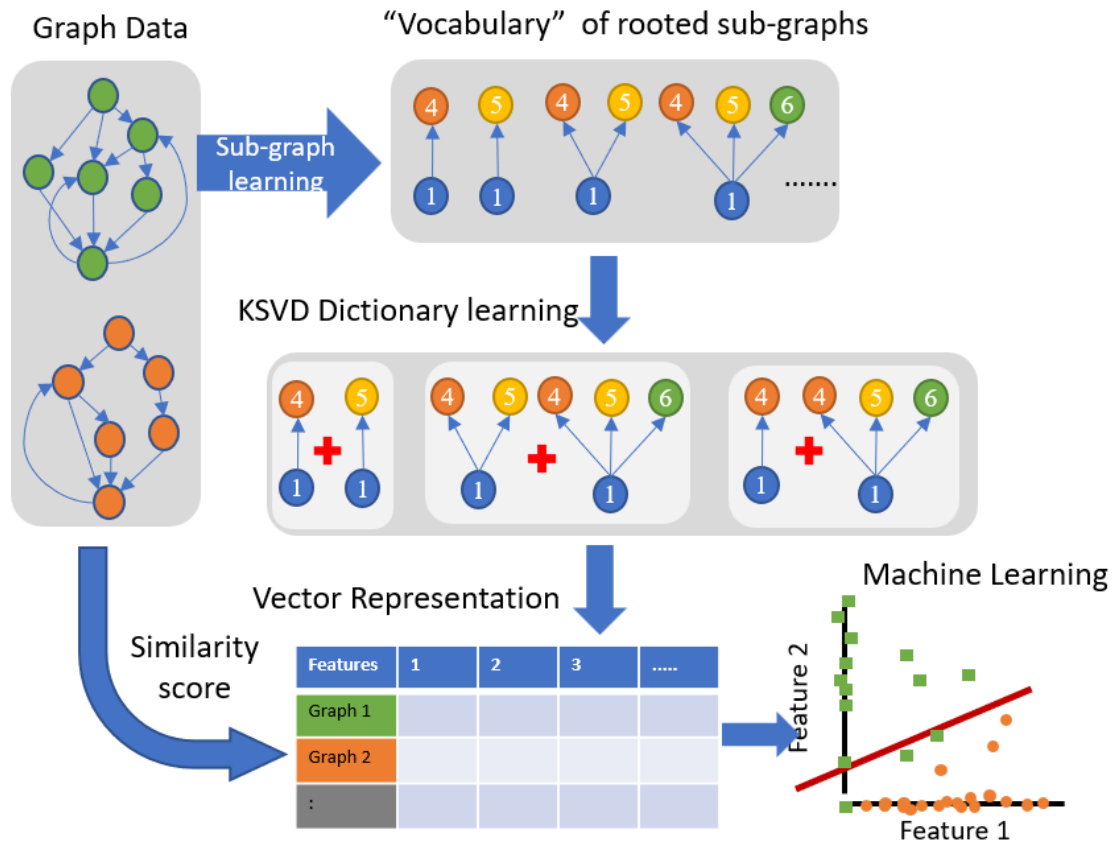


Figure 3.1: Proposed WL+KSVD pipeline overview

<sup>1</sup><https://github.com/BMW-lab-MSU/WL-KSVD.git>

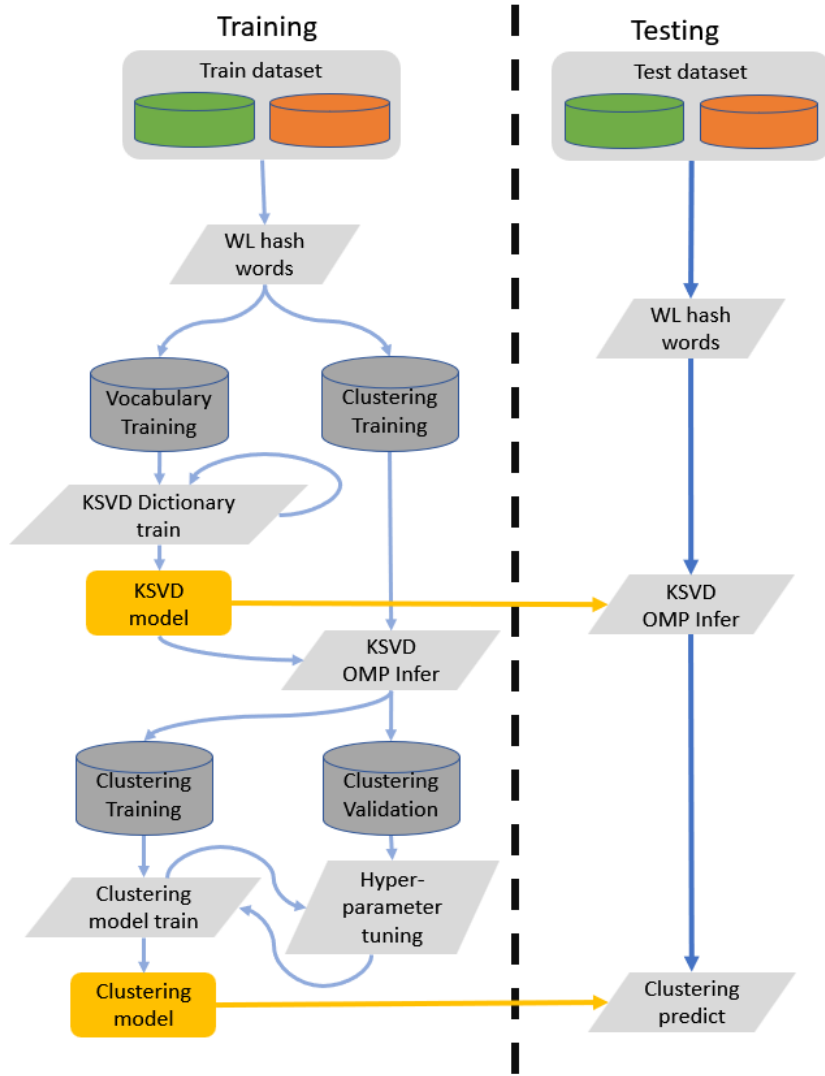


Figure 3.2: Evaluation Workflow

### Methodology

An overview of the workflow of the proposed method is shown in Fig.3.2, where the training set is further divided in half into embedding training and classifier training to avoid over fitting.

### 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  $\mathbf{G} = [G_1, G_2, \dots, G_M]$ . Following the Graph2Vec algorithm, if 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<sup>2</sup>.

### Weisfeiler-Lehman sub-tree kernel

WL sub-tree relabelling process (described in [6]<sup>3</sup>) 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  $sg_{i,j}^h$  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  $G_i = [sg_{1,i}^h, sg_{2,i}^h, \dots, sg_{l_i,i}^h]$ , where  $l_i$  is the number of nodes in  $G_i$ .

### Vocabulary Creation

Using the Doc2Vec implementation in Gensim python package<sup>4</sup> a raw vocabulary is created using the unique set of sub-tree hash words  $sg$  across all the training graphs [8]. 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.

---

<sup>2</sup><https://networkx.org/>

<sup>3</sup><https://github.com/benedekrozemberczki/karateclub/blob/master/karateclub/utils/treefeatures.py>

<sup>4</sup><https://radimrehurek.com/gensim/>

Each graph  $G_i$  is then represented as the occurrences  $Y_i$  of the vocabulary elements, where  $Y_i = [y_{i,1}, \dots, y_{i,K}]$  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:  $\mathbf{Y} = [Y_1, Y_2, \dots, Y_M] \in \mathbb{R}^{K \times M}$ .

### Sparse Representation

Let  $\mathbf{Y} = [Y_1, Y_2, \dots, Y_M] \in \mathbb{R}^{K \times M}$  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  $\mathbf{D} = [d_1, d_2, \dots, d_N] \in \mathbb{R}^{K \times N}$  while limiting the number of atoms used to  $T$  (sparsity). The sparse coefficient vector  $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_M] \in \mathbb{R}^{K \times M}$  will be the sparse representation with  $|\alpha|_0 \leq T$ , where  $|\cdot|_0$  operator counts the number of non-zero elements in the vector. The general form of the sparse representation can be formulated as:

$$\underset{\mathbf{D}, \alpha}{\operatorname{argmin}} \|\mathbf{Y} - \mathbf{D}\alpha\|_2^2 + \|\alpha\|_0. \quad (3.1)$$

This equation is NP-hard, but an approximate solution can be provided using an iterative algorithm named K-means Singular Value Decomposition (KSVD)[1]<sup>5</sup>. First, a dictionary  $\mathbf{D}$  is fixed and sparse vectors  $\alpha$  are optimized using Orthogonal Matching Pursuit (OMP)[13]. Second,  $\alpha$  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.

### Task 2: Feature Ranking

Let  $\mathbf{F} = [F_1, F_2, \dots, F_d]$  be a set of  $d$  features  $F$  which are collected or curated. The goal is to rank the feature set  $\mathbf{F}$  by evaluating a mean-removed training set of  $n$  samples:

---

<sup>5</sup><https://github.com/nel215/ksvd>

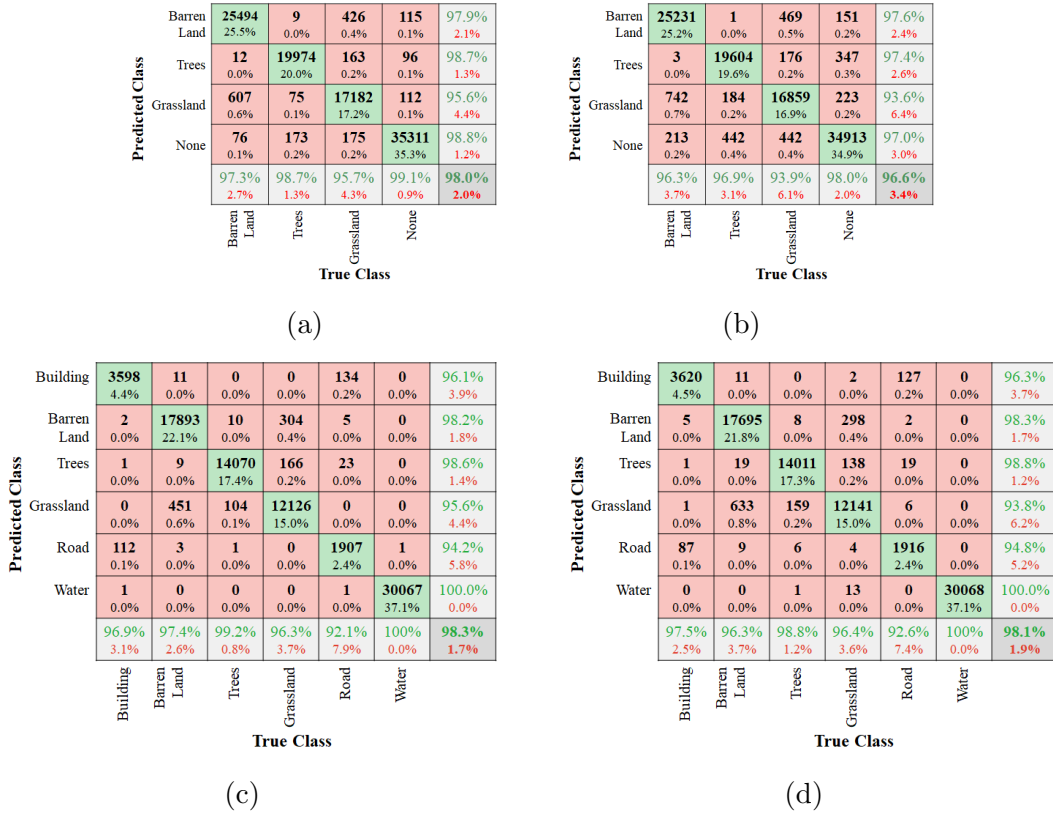


Figure 3.3: Confusion matrix for the Sat-4 (a,b) and Sat-6 (c,d) datasets with Frozen (a,c) and LC-KSVD (b,d) dictionary learning methods.

$\bar{\mathbf{X}} = [x_1, x_2, \dots, x_n] \in \mathbb{R}^{d \times n}$ . The set of  $k$  classes is defined as  $\mathbf{C} = [C_1, C_2, \dots, C_k]$ , where each of the  $x$  samples is assigned to a class  $C$ . In sparse representation the input sample is represented as a linear combination of dictionary elements  $D$  in a over-complete ( $d \ll m$ ) dictionary  $\mathbf{D} = [D_1, D_2, \dots, D_m] \in \mathbb{R}^{d \times m}$ , where the number of dictionary elements used,  $s$ , is far less than the number of dictionary atoms:  $s \ll m$ . The set coefficients  $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_n] \in \mathbb{R}^{m \times n}$  of the linear combination is called the sparse coefficients. Each coefficient vector contains  $s$  nonzero entries; the remaining  $m - s$  entries are exactly zero. The general objective function used for calculating the sparse representation is given by (3.2) with an  $\ell_0$  constraint on sparsity.

$$\underset{\mathbf{D}, \boldsymbol{\alpha}}{\operatorname{argmin}} ||\bar{\mathbf{X}} - \mathbf{D}\boldsymbol{\alpha}||_F + ||\boldsymbol{\alpha}||_0 \quad (3.2)$$

The KSVD algorithm is an efficient iterative method that solves the objective function by, first fixing the  $\mathbf{D}$  and optimizing the  $\boldsymbol{\alpha}$  using orthogonal matching pursuit (OMP)[13]. Second, it fixes  $\boldsymbol{\alpha}$  then optimize  $\mathbf{D}$  with generalized K-means and singular value decomposition (SVD). Since the learned dictionary is over-complete, the spread of the dictionary atoms can give an insight into which features are more relevant for the representation. Hence, we will be defining simple metrics that will quantify the spread of the dictionary elements in each of the features. First *Dictionary mapping*,  $\mathbf{D}_{\text{map}} \in \mathbb{R}^{1 \times d}$ , which calculates the sum of the squares of the projections of the dictionary atoms for each feature as given in (3.3), where  $\text{Proj}_j$  is the projection operator into feature  $F_j$ .

$$\mathbf{D}_{\text{map}}(j) = \sum_{i=1}^m \text{Proj}_j(D_i)^2 = \sum_{i=1}^m \mathbf{D}_{(j,i)}^2 \quad (3.3)$$

The second metric is the *Dictionary utilization*,  $\mathbf{D}_{\text{util}} \in \mathbb{R}^{1 \times d}$ , which calculated the utilization of the dictionary elements by the sparse coefficients. This acts as a weighted measure of the *dictionary mapping*. Finally, the weighted dictionary atoms are projected back to original features. The equation is given in (3.4).

$$\mathbf{D}_{\text{util}}(j) = \sum_{i=1}^m \text{Proj}_j(D_i \cdot \sum_{k=1}^n |\alpha_{j,k}|) \quad (3.4)$$

The KSVD algorithm learns the dictionary in an unsupervised manner, hence we cannot get dictionary elements that are optimized for class discrimination. Therefore several other methods have been proposed to learn a more discriminative dictionary by learning in a supervised manner, giving us a strong association of dictionary atoms with each class. Here we will be exploring two such methods, LC-KSVD and Frozen KSVD. By doing so we can

decompose the proposed metrics into classes, which gives more insight into feature behavior concerning class labels. In LC-KSVD the algorithm enforces two extra constraint terms related to dictionary association with each class and linear classifier performance. Hence, samples are forced to utilize a subset of the dictionary atoms for their representation leading to a more discriminatory dictionary. However, when imbalanced data is presented the LC-KSVD algorithm does not change the dictionary atom distribution. Therefore, Frozen KSVD is proposed to take class imbalances into consideration of dictionary learning. It first learns a dictionary only considering the largest class. Then next largest class is trained with added dictionary atoms while keeping the previously learned dictionary atoms fixed (frozen). This forces the algorithm to learn new dictionary atoms which are associated with only the new class. This is repeated for all the classes such that the added number of dictionary atoms and the sparsity of each class are reduced depending on the class size.

### Task 3: Hyperbolic graph embedding



## Bibliography

- [1] 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.
- [2] H. Cai, V. W. Zheng, and K. C.-C. Chang, “A comprehensive survey of graph embedding: Problems, techniques, and applications,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, pp. 1616–1637, sep 2018.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.
- [8] 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*, (Beijing, China), pp. 1188–1196, PMLR, 22–24 Jun 2014.
- [9] X. Rong, “word2vec parameter learning explained,” *arXiv preprint arXiv:1411.2738*, 2014.
- [10] 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.
- [11] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer, 2010.
- [12] R. Rubinstein, T. Peleg, and M. Elad, “Analysis k-SVD: A dictionary-learning algorithm for the analysis sparse model,” *IEEE Transactions on Signal Processing*, vol. 61, pp. 661–677, feb 2013.
- [13] 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*, pp. 40–44, IEEE, IEEE Comput. Soc. Press, 1993.
- [14] G. Davis, S. Mallat, and M. Avellaneda, “Adaptive greedy approximations,” *Constructive Approximation*, vol. 13, pp. 57–98, mar 1997.
- [15] Z. Jiang, Z. Lin, and L. S. Davis, “Learning a discriminative dictionary for sparse coding via label consistent k-SVD,” in *CVPR 2011*, IEEE, jun 2011.

- [16] 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.
- [17] 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.
- [18] K. A. Uthman, F. M. Ba-Alwi, and S. M. Othman, “A survey on feature selection in microarray data: Methods, algorithms and challenges,” *International Journal of Computer Sciences and Engineering*, 2020.
- [19] A. Sangodiah, R. Ahmad, and W. F. W. Ahmad, “A review in feature extraction approach in question classification using support vector machine,” in *2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014)*, pp. 536–541, IEEE, nov 2014.
- [20] D. Effrosynidis and A. Arampatzis, “An evaluation of feature selection methods for environmental data,” *Ecological Informatics*, vol. 61, p. 101224, mar 2021.
- [21] A. Jovic, K. Brkic, and N. Bogunovic, “A review of feature selection methods with applications,” in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, IEEE, may 2015.
- [22] Y.-W. Chang and C.-J. Lin, “Feature ranking using linear svm,” in *Proceedings of the Workshop on the Causation and Prediction Challenge at WCCI 2008* (I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, J.-P. Pellet, P. Spirtes, and A. Statnikov, eds.), vol. 3 of *Proceedings of Machine Learning Research*, (Hong Kong), pp. 53–64, PMLR, 03–04 Jun 2008.

- [23] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [24] K. Jong, J. Mary, A. Cornuéjols, E. Marchiori, and M. Sebag, “Ensemble feature ranking,” in *Knowledge Discovery in Databases: PKDD 2004* (J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, eds.), (Berlin, Heidelberg), pp. 267–278, Springer Berlin Heidelberg, 2004.
- [25] I. Kononenko, E. Šimec, and M. Robnik-Šikonja, “Overcoming the myopia of inductive learning algorithms with RELIEFF,” *Applied Intelligence*, vol. 7, no. 1, pp. 39–55, 1997.
- [26] C. Ding and H. Peng, “Minimum redundancy feature selection from microarray gene expression data,” *Journal of Bioinformatics and Computational Biology*, vol. 03, pp. 185–205, apr 2005.
- [27] Y. Zhang, Q. Zhang, Z. Chen, J. Shang, and H. Wei, “Feature assessment and ranking for classification with nonlinear sparse representation and approximate dependence analysis,” *Decision Support Systems*, vol. 122, p. 113064, jul 2019.
- [28] S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi, and J. A. Benediktsson, “Deep learning for hyperspectral image classification: An overview,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 9, pp. 6690–6709, 2019.
- [29] K. Liyanage and B. M. Whitaker, “Satellite image classification using LC-KSVD sparse coding,” in *2020 Intermountain Engineering, Technology and Computing (IETC)*, IEEE, 2020.
- [30] T. Neylon, *Sparse Solutions for Linear Prediction Problems*. PhD thesis, New York University, USA, 2006. AAI3221982.

- [31] Y. Yankelevsky and M. Elad, “Finding GEMS: Multi-scale dictionaries for high-dimensional graph signals,” *IEEE Transactions on Signal Processing*, vol. 67, pp. 1889–1901, apr 2019.
- [32] 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.