

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
Motivation	1
Goals and Objectives	1
Goal	1
Objective	1
2. BACKGROUND	3
Graphs Embedding	3
Sparse Representation and Dictionary Learning	5
Feature Ranking	7
Hyperbolic Space	10
3. PRELIMINARY WORK	11
Task 1: Graph KSVD	11
Methodology	12
Graph Notation and Definition	13
Weisfeiler-Lehman sub-tree kernel	14
Vocabulary Creation	14
Sparse Representation	15
Task 2: Feature Ranking	15
Problem	15
A Solution	16
Proposed Sparse Representation Based FR metrics	17
Dictionary mapping	18
Dictionary utilization	19
Task 3: Hyperbolic graph embedding	24
4. PROPOSED WORK	25
Task 1:	25
Task 2: Feture Ranking	25
tasks	25
Task 3:	29
Timeline	29

LIST OF TABLES

Table	Page
4.1 Methods used in MATLAB environment.....	26

LIST OF FIGURES

Figure	Page
1.1 overview	2
2.1 Graph2Vec pipeline overview	5
2.2 Classification of the FR methods and summary of pros and cons.....	8
3.1 Proposed WL+KSVD pipeline overview	12
3.2 Evaluation Workflow	13
3.3 Issues with common FR methods, the common causes and the properties of SR which could improve the issues	18
3.4 Projection of dictionary elements and weights into input space. (a) <i>dictionary mapping</i> only projects the dictionary atoms. (b) <i>dictionary utilization</i> projects the weighted dictionary atoms according to the sparse representation.....	18
3.6 FR scores obtained using various methods on the Sat-6 dataset. The dictionary scores were obtained using frozen KSVD sparse representation.	22

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

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.

Motivation

Goals and Objectives

Goal

Objective

A Graph is represented as one sparse vector and two graphs with similar sub- structure are embedded to be closer.

The Graph can be

- Directed

- Cyclic

- Have node feature

1. To develop a sub-tree pattern based sparse graph embedding method
2. Framework for Identifying important sub-tree patterns
3. Develop a sparse graph representation in hyperbolic space

Objectives overview

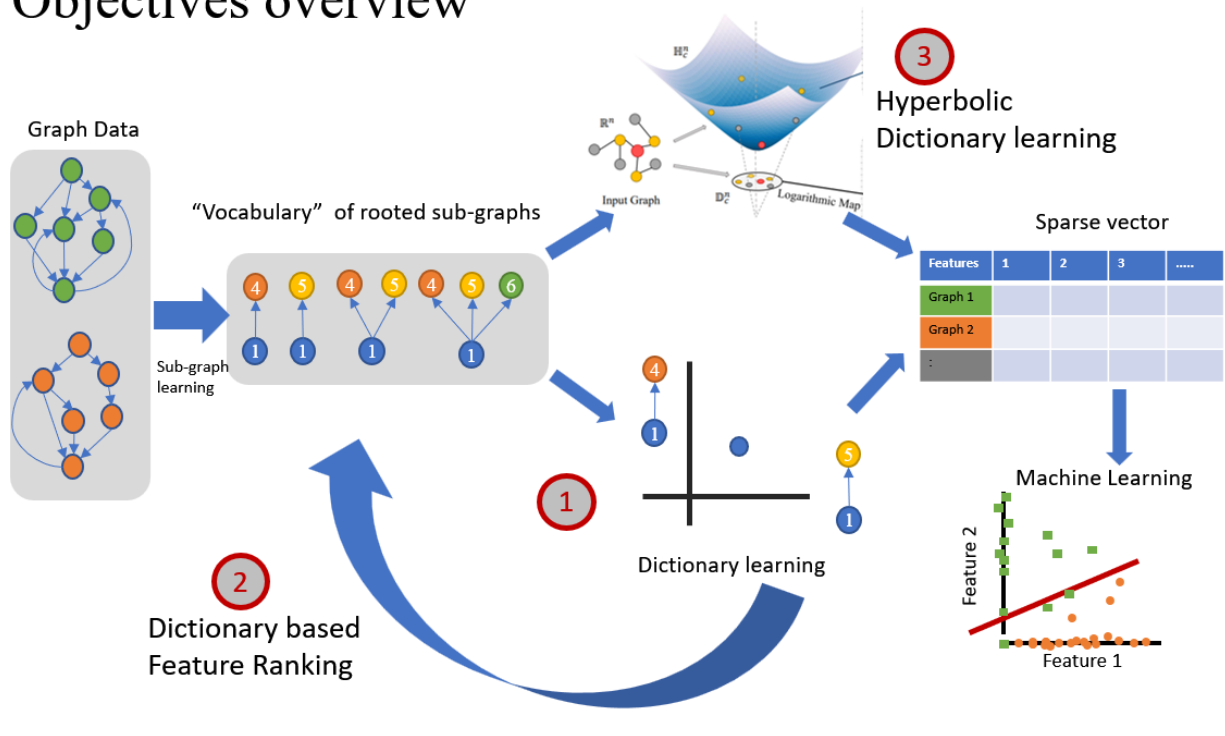


Figure 1.1: The proposal objectives overview

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[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. Graph2Vec utilizes the non-linear Weisfeiler-Lehman (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. 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.

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

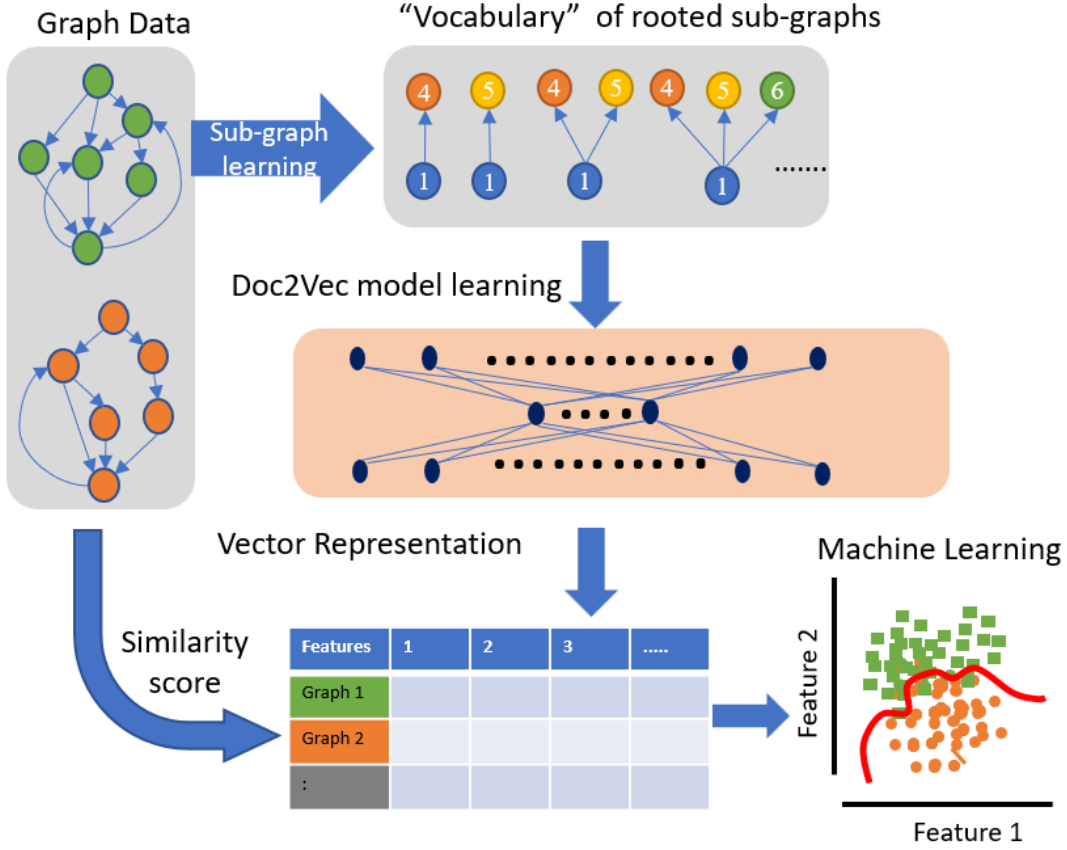


Figure 2.1: Graph2Vec pipeline overview

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[10]. 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 α_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 α with a fixed over-complete dictionary D . Second, dictionary learning is performed to update D with a fixed α . K-means Singular Value Decomposition (K-SVD)[11, 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

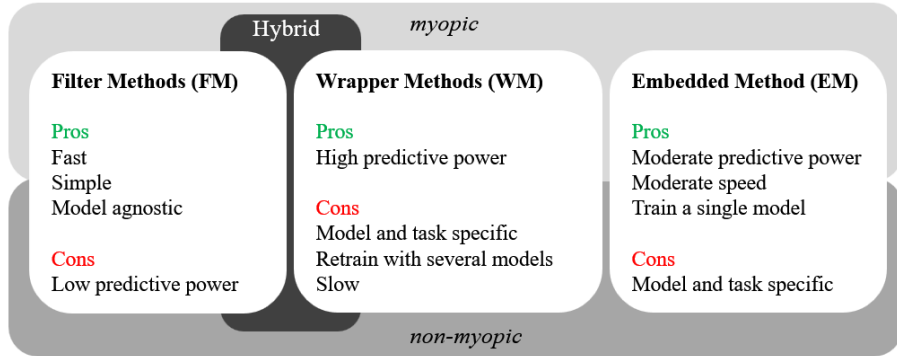


Figure 2.2: Classification of the FR methods and summary of pros and cons

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]. A summary of the method's pros and cons are shown in fig.2.2.

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)[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[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 Github¹.

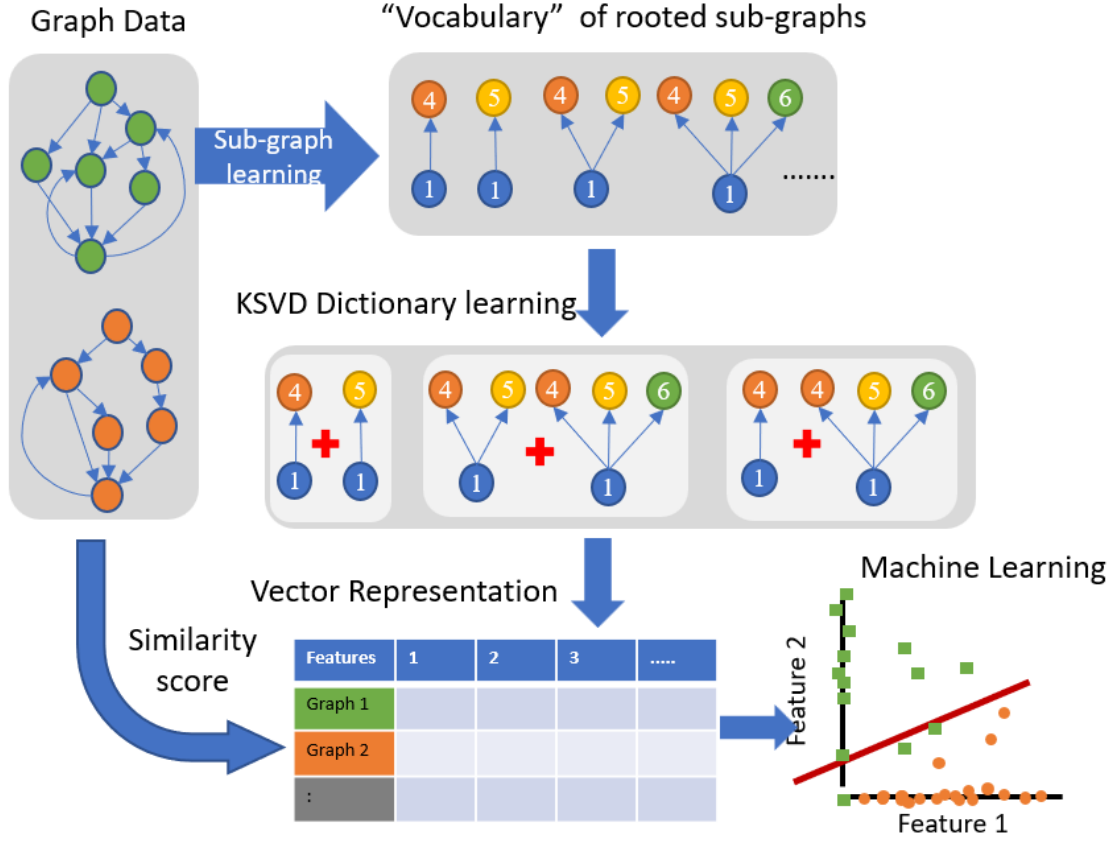


Figure 3.1: Proposed WL+KSVD pipeline overview

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.

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

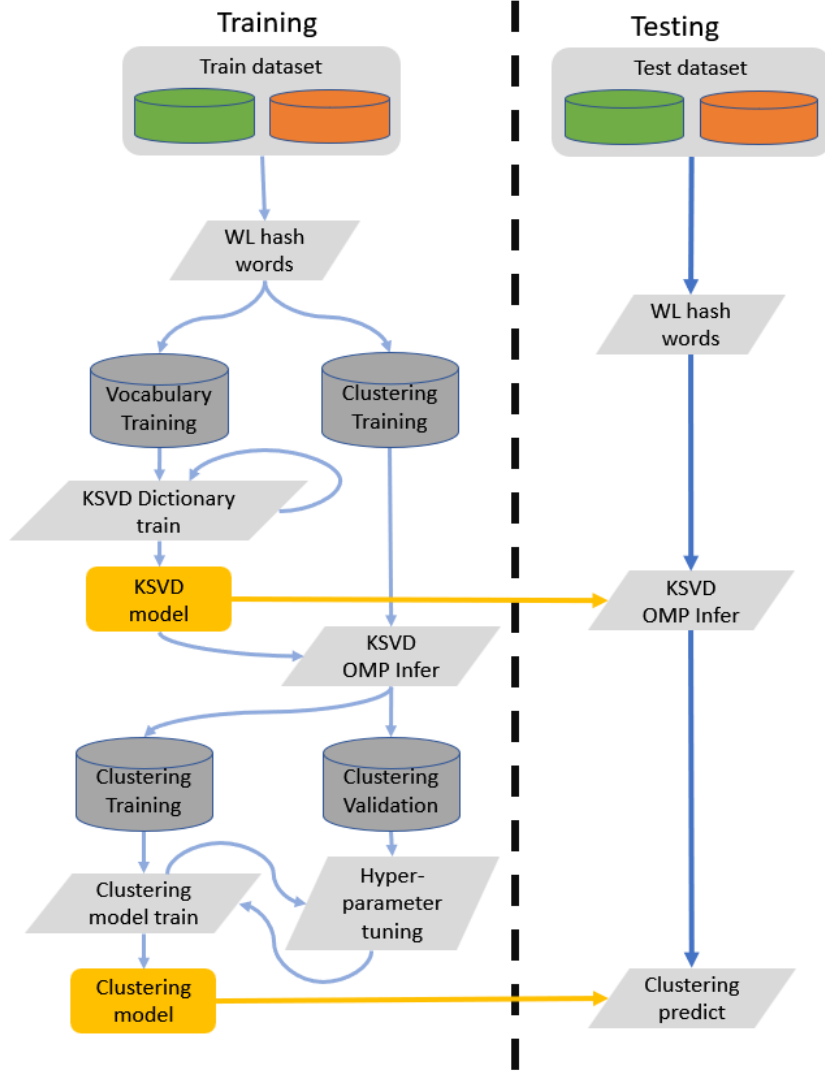


Figure 3.2: Evaluation Workflow

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 structure².

Weisfeiler-Lehman sub-tree kernel

WL sub-tree relabelling process (described in[5]³) 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 package⁴ 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 $Y_i = [y_{i,1}, \dots, y_{i,K}]$ 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: $\mathbf{Y} = [Y_1, Y_2, \dots, Y_M] \in \mathbb{R}^{K \times M}$.

²<https://networkx.org/>

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

⁴<https://radimrehurek.com/gensim/>

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)[11]⁵. First, a dictionary \mathbf{D} is fixed and sparse vectors α are optimized using Orthogonal Matching Pursuit (OMP)[13]. 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.

Task 2: Feature Ranking

Problem

Although many FR algorithms exist, there is no universal method. Each FR method focuses on different aspects of the features depending on applications requirements. The problems identified in common FR metrics and aiming to address are as follows.

1. Interpretability.

⁵<https://github.com/nel215/ksvd>

Some of these measures are highly non-linear and complex, it is hard to have an intuitive understanding of the FR process. There is no direct/linear relationship between the features and the ML model behavior.

2. Class-wise feature relevance.

Most of the FR methods cannot determine class-wise FR in the presence of multiple classes. Especially in FM, although FR can be applied to each class separately or as a whole, they cannot distinguish feature relevance for each class. The relationship between the classes is not considered when ranking is calculated for a single class.

3. Imbalance class data.

Some traditional FR methods do not consider the class distributions or the class imbalances. This affects unfavorably small classes of the dataset as the larger classes will overwhelm the FR metric. This becomes a significant problem when you are trying to detect abnormalities/ outliers or have small sample size classes. For example, in a medical computational topology (CT) scan you might have rare occurrences of lesions. If you just apply FR metrics to find the best features to represent the CT images it will find the “best” of features to represent the majority of the images. Which could omit features that would be specific to represent the small number of lesions.

A Solution

Sparse representation (SR) is a method of increasing the dimensions of the dataset to achieve a better representation of the data[10]. This process is linear and easier to understand. In SR, the process is divided into two steps. First, it learns a dictionary of subspaces of original feature space. These, subspaces or the dictionary elements are “atoms”. Second, it tries to approximately represent each datum as a linear combination of those dictionary atoms. In practice, these two processes are iteratively optimizing each

other until a stopping criterion is met. SR methods are effective in several under several assumptions. One is the original data is laying in a subspace of the original feature space This assumption seems to be true for applications like images and music analysis. The learned atoms reveal some information about the data distribution in the original feature space.

Hence, we are proposing a method that will use SR to identify important features for FR tasks. The *dictionary mapping*, and *dictionary utilization* are introduced as novel and simple metrics for FR tasks. These metrics evaluate the learned dictionary atoms and linearly transform them to the original feature space. This allows us to observe the atom distribution in the original feature space. These methods are *non-myopic* and a hybrid of FM and EM concepts. To calculate the 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. Unlike, EM methods these FR scores can be used in many applications. Furthermore, the proposed metrics do not depend on any particular dictionary learning 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*. Figure3.3 summarizes the issues identified and the aspects of the SR that would help improve/address each of the issues pertaining to common FR metrics. It should be noted that not every FR metric has these issues, and might address one or more issues. However, with SR methods we could improve all of the issues at once.

Proposed Sparse Representation Based FR metrics

Figure3.4 shows the linear mapping of the dictionary elements and weighted dictionary elements to the original input space. Even though the metrics are simple projections of the dictionary elements into input space, Learning dictionary elements are complex.

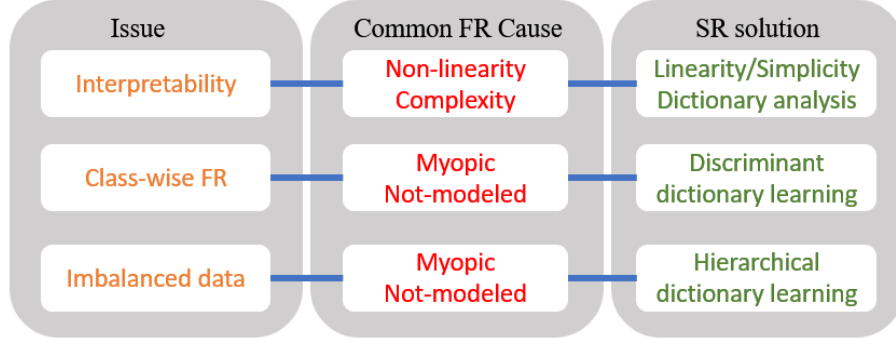


Figure 3.3: Issues with common FR methods, the common causes and the properties of SR which could improve the issues

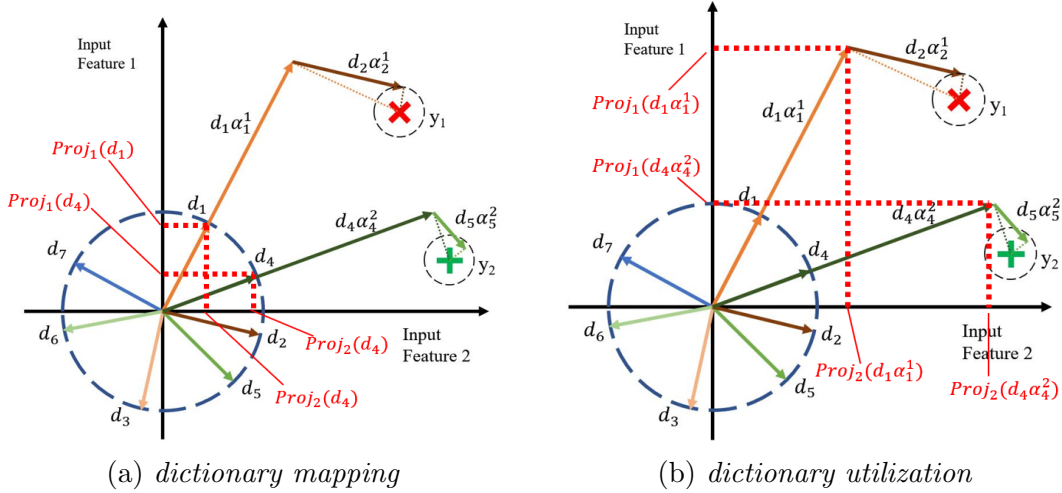


Figure 3.4: Projection of dictionary elements and weights into input space. (a) *dictionary mapping* only projects the dictionary atoms. (b) *dictionary utilization* projects the weighted dictionary atoms according to the sparse representation.

The dictionary elements are learned through the KSVD algorithm iteratively optimizing, considering all the training samples. Hence the learned dictionary elements capture the common patterns in the dataset.

Dictionary mapping 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 (3.2), where Proj_j is the projection operator into feature F_j .

$$\mathbf{D}_{\text{map}}(j) = \sum_{i=1}^m \text{Proj}_j(d_i)^2 = \sum_{i=1}^m \mathbf{D}_{(j,i)}^2 \quad (3.2)$$

Figure 3.4.a shows the projections used for *dictionary mapping*. The Sum of squares of the projections is used to calculate the metric. More dictionary elements will be concentrated in subspaces where the data is highly distributed. We hypothesize features with high variance should have a higher *dictionary mapping* score. This proposal will try to find mathematical and experimental validation for this claim.

Dictionary utilization 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 their original features. The equation is given in (3.3).

$$\mathbf{D}_{\text{util}}(j) = \sum_{i=1}^m \text{Proj}_j(d_i \cdot \sum_{k=1}^n |\alpha_{j,k}|) \quad (3.3)$$

Figure 3.4.b shows the projections used for *dictionary utilization*. Since we are learning an over-complete dictionary and representing data sparsely, Not all the dictionary atoms are used all the time. Hence, understanding which dictionary atoms are used by the sample can give insight into common patterns and rare occurrences. If identify the dictionary atom usage b class, we can develop class-wise feature importance. The more the dictionary atom is used the higher the score for its relevant feature.

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: $\bar{\mathbf{X}} = [x_1, x_2, \dots, x_n] \in \mathbb{R}^{d \times n}$. 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 ($d \ll$

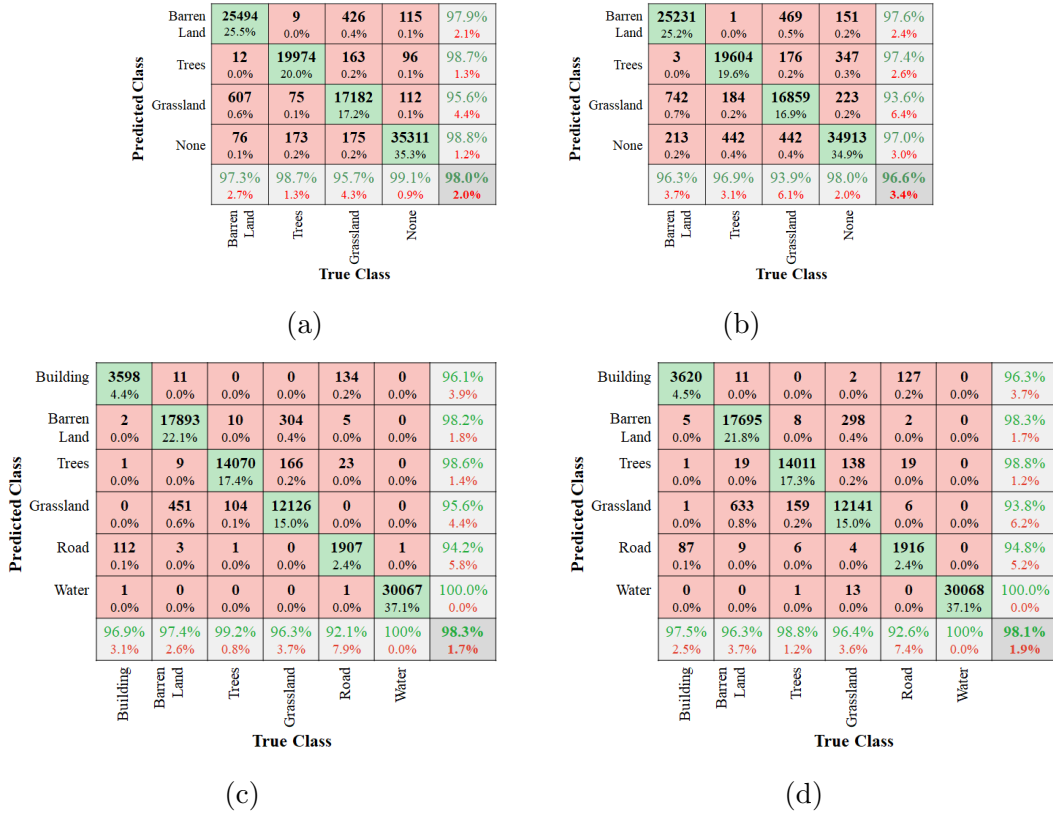


Figure 3.5: Confusion matrix for the Sat-4 (a,b) and Sat-6 (c,d) datasets with Frozen (a,c) and LC-KSVD (b,d) dictionary learning methods.

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 (3.4) with an ℓ_0 constraint on sparsity.

$$\underset{\mathbf{D}, \boldsymbol{\alpha}}{\operatorname{argmin}} \|\bar{\mathbf{X}} - \mathbf{D}\boldsymbol{\alpha}\|_F + \|\boldsymbol{\alpha}\|_0 \quad (3.4)$$

The KSVD algorithm is an efficient iterative method that solves the objective

function by, first fixing the \mathbf{D} and optimizing the α using orthogonal matching pursuit (OMP)[13]. Second, it fixes α 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 (3.2), where Proj_j is the projection operator into feature F_j .

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 (3.3).

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

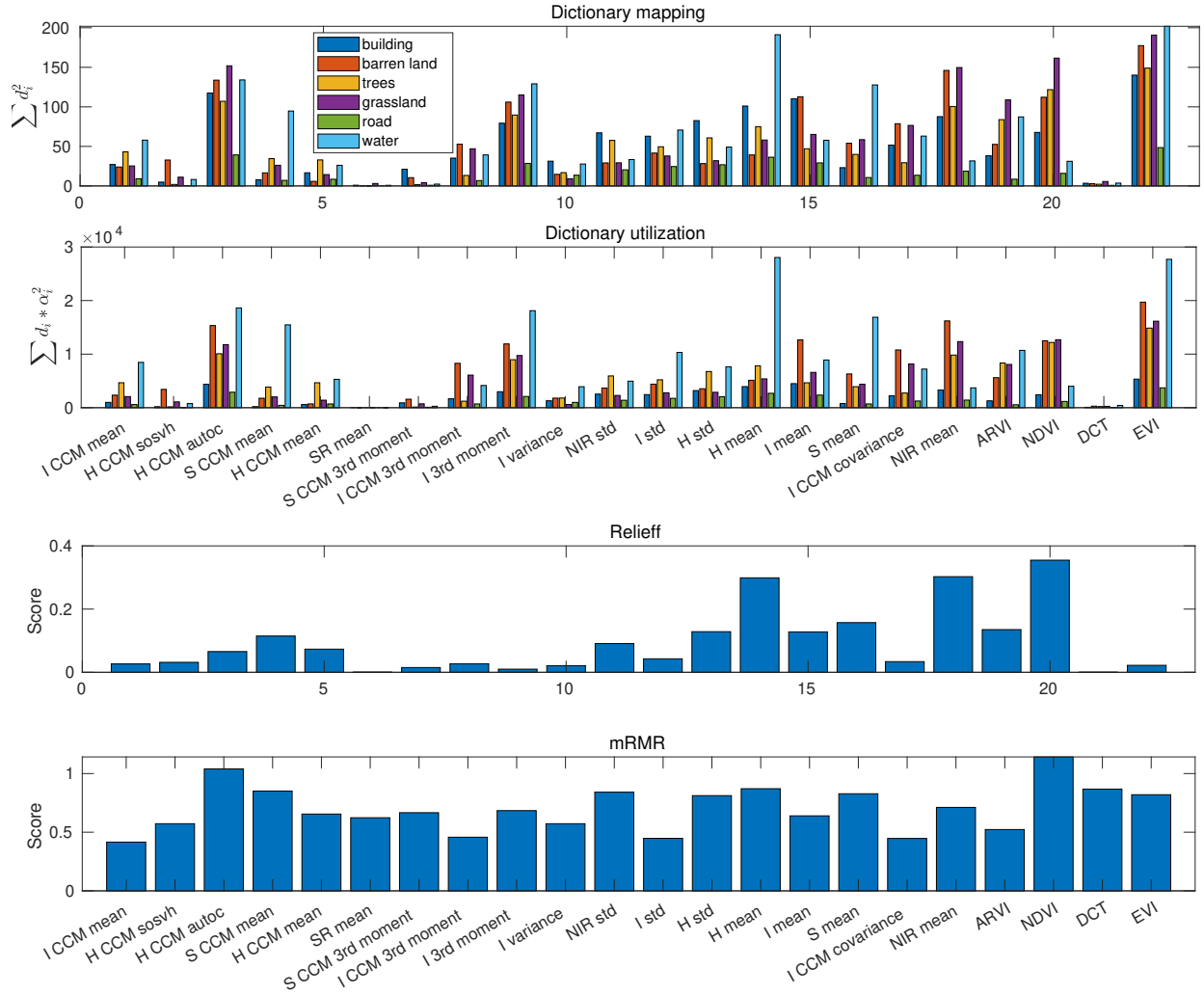


Figure 3.6: FR scores obtained using various methods on the Sat-6 dataset. The dictionary scores were obtained using frozen KSVD sparse representation.

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.

subsectionDictionary Elements Analysis

When we examine the dictionary atoms used by correctly and incorrectly classified images it can be observed that incorrectly classified images share some of the dictionary

atoms that are mostly allocated to the other class. When we transform these dictionary elements back to the feature space we can identify the features that give similar parameters to the two classes, indicating that these features are not able to distinguish between the two classes. Hence, we can decide whether to replace the features or add new features that are capable of separating the two classes. Since our feature domain is considerably small this will not drastically increase the computational burden of the system for this example. This ability to identify the features responsible for misclassifications is not intuitively simple with deep networks and other FR methods.

As an example, we can take a closer look at the most prevailing confusion between classes *barren land* and *grassland* in the Sat-6 dataset from the Frozen dictionary model. Most of the the misclassified images belonging to these classes contain soil and grass patches. For brevity, we do not present images of the misclassified data points, but the difficulty in producing accurate ground-truth labels likely contributes to the confusion and lower accuracy. This can be observed also in the misclassified *building* class, where many of the images misclassified as *roads* were parking lots. This, combined with the fact that *roads* have very small training data, would accumulate to the low performance of the *road* class. While a similar analysis could be done for images classified with a deep network, the linearity employed by dictionary learning and classification in this work allows for a more comprehensive analysis.

Fig.3.6 shows the *dictionary mapping* and the *dictionary utilization* of the sparse coefficients used by dictionary elements projected into the original feature space from the frozen KSVD method on Sat-6 data. Since each dictionary atom is associated with a class, the class-wise metric analysis is performed on the original features. It also shows the FR done by the non-class specific Relieff and mRMR methods. Both proposed metrics perform in a similar fashion. While their scores are closer to the Relieff method they are also able to capture some important features predicted by mRMR method (e.g. *H CCM autoc* and

EVI). The proposed methods also match the Relieff method in identifying SR mean and DCT as underutilized features.

The shortcomings of SR are well recorded in cases where an abundance of bare soil is present, and indices like Soil Adjusted Vegetation Indexes (SAVI) can be used to mitigate this problem[33]. Due to the presence of similar texture in both *barren land* and *grassland* classes, its DCT components also give similar results. Since we have used a modified and more statistical version of the 22 features selected by[34], some features do not perform well with the sparse model. Using sparse coding and linear classifiers, we can identify, evaluate, and improve the necessary input features that might increase the performance of the classification.

Concerning classification performance between the two SR methods used, Frozen KSVD performs slightly better than the LC-KSVD in the Sat-6 dataset. This is possibly due to the strategy Frozen KSVD employs in taking class sizes into account and learning dictionary elements hierarchically. This enables the algorithm to produce dictionary atoms that specifically model the smaller classes. For the Sat-4 dataset, Frozen KSVD performs notably better than LC-KSVD. This may be due to the high variability of the *none* class.

Task 3: Hyperbolic graph embedding

PROPOSED WORK

Task 1:

Task 2: Feture Ranking

subsectionPerformance evaluation

The proposed FR metrics will be tested and compared according to the different combinations and methods in table4.1. The features are removed one by one and classifiers are trained to observe the performance degradation. Preliminary results for *ionosphere* dataset with Gaussian SVM is shown in fig.

For this example dataset, we can observe that removing unnecessary features helps improve the classifier performance to some extent. Further, the proposed FR metrics outperform the existing method in identifying irrelevant features in the initial stage. These preliminary results show the potential of the proposed metrics and validate the need for further investigation. Similarly, performance will be analyzed for common, synthetic, and real-world datasets.

tasks

Following tasks will be completed for the achievement of the proposed goals and objectives

1. Achieve similar performance degradation to existing FR methods within 95% confidence interval within three months

Some of the common FR methods will be considering are Releiff[25], mRMR[26], NCA[35], χ^2 -test, and PPCA[36]. These methods will be tested using common dataset such as, *credit ratings*, *ionosphere*, *letters*, and synthetic data.

Table 4.1: Methods used in MATLAB environment

Dataset	FR methods	Classifier method
Credit ratings	mRMR	Gaus SVM
Ionosphere	RELEIFF	Lin SVM
letters	χ^2 -test	Quad SVM
	NCA	Pol SVM
	D_{map}	Logistic regression
	D_{util}	Neural Network

- (a) Perform mathematical analysis of the proposed metrics with common FR metrics

Each of the considered FR methods evaluates a different aspect of the features. We will discuss the theoretical similarities and differences of each metric compared to the proposed metrics.

- (b) Perform the test with common FR methods and common data sets in one month

The ranking given by different FR methods will be compared to the proposed metrics. The ranking given by common methods will be averaged and ranked. Then averaged ranking and proposed ranking will be compared to get a value of how many features matched the ranking. Will discuss the experimental differences and similarities to understand the effect of data type and training hyper-parameters. Identify sensitiveness to the different data.

- (c) Record performance degradation with feature removal and training times for each FR method

The features are removed one by one by the ranking and the machine learning classifier performance will be measured.

- (d) Evaluate, compare, and improve the results in two months to 95% confidence interval.

The performance degradation will be compared to establish the shortcomings. Improvements to the metrics such as normalization and log-mapping can be introduced to improve performance.

2. Improving the performance on imbalanced data by incorporating two other discriminatory SR methods in 4 months

K-means singular value decomposition (KSVD)[11] is an efficient algorithm for SR-based methods. However, it is an unsupervised dictionary learning method and cannot incorporate label information. Hence, supervised dictionary learning algorithms will be tested with proposed metrics to improve the feature interpretability.

- (a) Finalizing the LCKSVD and Frozen KSVD data representation pipeline in two months.

To incorporate class label information several variations of the KSVD algorithms exist. Two of the popular methods are label-consistent KSVD (LCKSVD)[15], and Frozen KSVD[17]. The pipeline will be finalized so the class-wise FR can be generated with the proposed metrics.

- (b) Testing two other discriminatory SR methods and evaluating the performance to identify the best methods for the FR metric.

Another two supervised dictionary learning methods are discriminatory KSVD (DKSVD)[37] and discriminative dictionary learning-based SR classification (DDL-SRC)[38]. These methods will be included in the pipeline so that the researchers would have multiple choices to carry out FR metrics to interpret the data.

3. Testing FR on Satellite images for classification for LULC tasks

Satellite image classification is a challenging problem due to the high variability of the images, lack of reliable ground truth labels, and an abundant presence of atmospheric noise. Researchers use hand-crafted features and evaluate these images to identify land usage and land coverage. We will use the proposed FR metrics to identify and rank important/relevant features in different LULC scenarios.

(a) Training ML model with Sat-4, and Sat-6 dataset in three months

Sat-4 and Sat-6[34] are two widely used dataset for LULC tasks and has been benchmarked by various researchers[39, 40, 41, 42, 43]. We will develop several ML models including SVM[44], logistic regression[45], and neural networks to train on the dataset.

(b) Evaluate the performance of models concerning other FR methods in five months

The features will be removed according to rankings given by different FR methods, and Models will be re-trained with the remaining features. The performance degradation will be evaluated to assess the performance of the proposed metrics.

4. Testing FR on Binary files to detect malware using control flow graphs (CFG)

Detecting computer malware by just examining the binary files is challenging. Since, most of the vulnerabilities occur as run-time events, and depend on platform weaknesses; detecting malicious behavior is hard, without running them on targeted platforms. A common approach is to generate control flow graphs (CFG) to analyze graph structure[]. Hence, we will be using the FR metric to identify the most important graph nodes and edges for the ML tasks.

(a) Creating dataset of Benign and Malware dataset in two months

A database with malware is provided by an anti-virus company *Hoplite*[], and a database of benign programs has to be curated by scrapping the online app-stores[]. These datasets will comprise different variations of malware and benign software.

- (b) Incorporating SR methods into graph embedding learning in three months

SR methods are not commonly used in graph analysis[] pre-processing. Hence, a pipeline to incorporate the SR method has to be developed.

- (c) Evaluate the performance of models concerning other FR methods in five months

The graph nodes and edges will be removed according to rankings given by different FR methods, and the performance degradation will be evaluated to assess the performance of the proposed metrics.

Task 3:

Timeline

Bibliography

- [1] 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.
- [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*, (Beijing, China), pp. 1188–1196, PMLR, 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: From Theory to Applications in Signal and Image Processing*. Springer, 2010.
- [11] 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.
- [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. Rezaeiifar, 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.
- [33] A. Huete, “A soil-adjusted vegetation index (savi),” *Remote Sensing of Environment*, vol. 25, no. 3, pp. 295–309, 1988.
- [34] S. Basu, S. Ganguly, S. Mukhopadhyay, R. DiBiano, M. Karki, and R. Nemani, “Deep-sat: a learning framework for satellite imagery,” in *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '15*, pp. 1–10, ACM Press, 2015.
- [35] J. Goldberger, G. E. Hinton, S. Roweis, and R. R. Salakhutdinov, “Neighbourhood components analysis,” in *Advances in Neural Information Processing Systems* (L. Saul, Y. Weiss, and L. Bottou, eds.), vol. 17, MIT Press, 2005.
- [36] M. E. Tipping and C. M. Bishop, “Probabilistic principal component analysis,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.
- [37] Q. Zhang and B. Li, “Discriminative k-SVD for dictionary learning in face recognition,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 2010.

- [38] Y. Kong, T. Wang, F. Chu, Z. Feng, and I. Selesnick, “Discriminative dictionary learning-based sparse classification framework for data-driven machinery fault diagnosis,” *IEEE Sensors Journal*, vol. 21, no. 6, pp. 8117–8129, 2021.
- [39] T. Dundar and T. Ince, “Sparse representation-based hyperspectral image classification using multiscale superpixels and guided filter,” *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 2, pp. 246–250, 2019.
- [40] B. Pan, Z. Shi, and X. Xu, “Multiobjective-based sparse representation classifier for hyperspectral imagery using limited samples,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 1, pp. 239–249, 2019.
- [41] J. Li, Q. Du, Y. Li, and W. Li, “Hyperspectral image classification with imbalanced data based on orthogonal complement subspace projection,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 7, pp. 3838–3851, 2018.
- [42] S. Huang, H. Zhang, and A. Pižurica, “A robust sparse representation model for hyperspectral image classification,” *Sensors*, vol. 17, p. 2087, sep 2017.
- [43] Y. Tang, X. Li, Y. Xu, Y. Liu, J. Wang, C. Liu, and S. Liu, “Hyperspectral image classification using sparse representation-based classifier,” in *2014 IEEE Geoscience and Remote Sensing Symposium*, IEEE, jul 2014.
- [44] V. N. Vapnik, *Statistical Learning Theory*. WILEY, 1998.
- [45] R. E. Wright, “Logistic regression,” *American Psychological Association*, 1995.