EXTRACTING AND HARNESSING INTERPRETATION IN DATA MINING

A Dissertation

by

NINGHAO LIU

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Xia Hu
Committee Members,	James Caverlee
	Nick Duffield
	Ruihong Huang
Head of Department,	Scott Schaefer

May 2021

Major Subject: Computer Science

ABSTRACT

Machine learning, especially the recent deep learning technique, has aroused significant development to various data mining applications, including recommender systems, misinformation detection, outlier detection, and health informatics. Unfortunately, while complex models have achieved unprecedented prediction capability, they are often criticized as "black boxes" due to multiple layers of non-linear transformation and the hardly understandable working mechanism.

To tackle the opacity issue, interpretable machine learning has attracted increasing attentions. Traditional interpretation methods mainly focus on explaining predictions of classification models with gradient based methods or local approximation methods. However, the natural characteristics of data mining applications are not considered, and the internal mechanisms of models are not fully explored. Meanwhile, it is unknown how to utilize interpretation to improve models. To bridge the gap, I developed a series of interpretation methods that gradually increase the transparency of data mining models. First, a fundamental goal of interpretation is providing the attribution of input features to model outputs. To adapt feature attribution to explaining outlier detection, I propose Contextual Outlier Interpretation (COIN). Second, to overcome the limitation of attribution methods that do not explain internal information inside models, I further propose representation interpretation methods to extract knowledge as a taxonomy. However, these post-hoc methods may suffer from interpretation accuracy and the inability to directly control model training process. Therefore, I propose an interpretable network embedding framework to explicitly control the meaning of latent dimensions. Finally, besides obtaining explanation, I propose to use interpretation to discover the vulnerability of models in adversarial circumstances, and then actively prepare models using adversarial training to improve their robustness against potential threats.

My research of interpretable machine learning enables data scientists to better understand their models and discover defects for further improvement, as well as improves the experiences of customers who benefit from data mining systems. It broadly impacts fields such as Information Retrieval, Information Security, Social Computing, and Health Informatics.

ACKNOWLEDGMENTS

Throughout the writing of this thesis, I have received a great deal of support and assistance. Here I would like to express my sincere thanks and appreciation to all the people who have helped me in my research and the completion of the thesis.

First, I would like to thank my supervisor, Professor Xia Hu, for his help, inspiration, and professional guidance that made this thesis possible. His expertise and research thinking were invaluable in formulating the research questions and methodology.

Second, I would like to thank my dissertation committee members, Professor James Caverlee, Professor Nick Duffield, and Professor Ruihong Huang, for their time, advice, and support.

Also, I would like to acknowledge my lab mates and collaborators at the DATA (Data Analytics at Texas A&M) Lab in the Department of Computer Science and Engineering, for helping me and providing advice for my work.

Most importantly, I owe my sincerest gratitude to my parents. Their love and encouragement have provided strong support for my pursuit of the doctoral study and the completion of the thesis.

Finally, I would like to thank the Texas A&M University and all the funding agencies, including National Science Foundation and Defense Advanced Research Projects Agency.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by the dissertation committee consisting of Professor Xia Hu [advisor], Professor James Caverlee and Professor Ruihong Huang from the Department of Computer Science and Engineering, and Professor Nick Duffield from the Department of Electrical and Computer Engineering.

All the work conducted for the dissertation was completed by the student independently.

Funding Sources

This work is, in part, supported by DARPA (#N66001-17-2-4031, #W911NF-16-1-0565), and NSF (#IIS-1657196, #IIS-1718840, #IIS-1750074). The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION [1]

1.1 Background and Motivation

Data mining techniques are progressing at an astounding rate, powered by the advances of machine learning techniques such as deep neural networks (DNNs). Data mining covers a broad range of application scenarios, such as recommender systems, cybersecurity analysis, sentiment analysis, healthcare and outlier detection. Different applications rely on specific machine learning models to achieve the fundamental functionalities. For example, recommender systems make use of embedding models to map users and products into latent space to express their similarities. Convolutional Neural Networks (CNNs) could also be used to learn the representation of product appearance or textual descriptions. Healthcare or sentiment analysis utilizes Recurrent Neural Networks (RNNs) to identify the semantic meaning of texts. Outlier detection may also make use of deep autoencoders to map instances into more effective low-dimensional representations, which facilitates subsequent measurement of outlierness degree.

Despite the success, machine learning techniques have been criticized due to their opacity in information processing and decision making. Such lack of transparency could prevent end users from trusting the machine learning models, and reduce the potential of further improving the system. For examples, in healthcare domains, it is difficult for patients and medical doctors to fully trust the diagnosis from the model if it fails to provide convincing reasons to support the

[1] Part of this chapter is reprinted with permission from "Contextual outlier interpretation" by Ninghao Liu, Donghwa Shin, and Xia Hu, Proceedings of the 27th International Joint Conference on Artificial Intelligence, Pages 2461–2467, Copyright 2018 by the IJCAI Organization, "Adversarial detection with model interpretation" by Ninghao Liu, Hongxia Yang, and Xia Hu, Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Pages 1803–1811, Copyright 2018 by Association for Computing Machinery, "Representation interpretation with spatial encoding and multimodal analytics" by Ninghao Liu, Mengnan Du, and Xia Hu, Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, Pages 60–68, Copyright 2019 by Association for Computing Machinery, "Is a single vector enough? exploring node polysemy for network embedding" by Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Pages 932–940, Copyright 2019 by Association for Computing Machinery, "Explainable Recommender Systems via Resolving Learning Representations" by Ninghao Liu, Yong Ge, Li Li, Xia Hu, Rui Chen, and Soo-Hyun Choi, Proceedings of the 2020 ACM on Conference on Information and Knowledge Management, Pages 895–904, Copyright 2020 by Association for Computing Machinery.

diagnosis. In addition, for autonomous vehicles, it is easier to debugging the self-driving system after accidents if relevant explanations are available towards the history self-driving behaviors.

Interpretation would be an effective tool to mitigate the problems above. It gives machine learning models the ability to explain or to present their behaviors in human-understandable terms. However, there is no such a unified definition of interpretation. The definition of interpretation, as well as the design of interpretation methods, depend on the target scenario and what questions we would like the interpretation methods to answer. In this thesis, by gradually increasing the level of modeling transparency, I propose to answer the questions below via interpretation. First, given a model prediction result, how to attribute the result to the most responsible input features? Second, besides explaining predictions, what additional insights can we obtain from the internal intermediate model states (i.e., representations) of input? Third, besides passively explaining the given predictions or models, can we actively design models in which interpretability is already taken into account during its construction and training stages? Additionally, given the interpretations at hand, can we turn back and utilize interpretations to improve models? The detailed analysis of each question are as follows.

1.1.1 The Difficulty of Explaining Model Predictions

The existing major efforts paid towards interpretability focus on providing locally discriminating features for classification models or refining the explanation from readily interpretable models. For examples, CAM [7] and GradCAM [8] generate saliency maps to input images with respect to image classification results. Specific neurons in CNNs could be visualized to unveil their perceived patterns [9]. In text classification, existing methods could also provide saliency scores to text segments [10]. Some interpretation methods, such as LIME [11] and Mimic Learning [12], are model-agnostic and can be applied to different types of models.

Despite the advances above, we are still facing several challenges. First, besides explaining classifiers, how to design methods that work on different scenarios such as outlier detection and clustering? Second, how to evaluate whether the obtained interpretations are accurate? Third, how to make use of interpretations to improve models in terms of robustness and credibility? We will

show how to tackle these challenges in this thesis.

1.1.2 The Difficulty of Understanding Latent Representations

Representation learning has a fundamental impact on the performance of downstream machine learning models, especially when processing raw data inputs such as images [13], texts [14, 15] and networks [16, 17]. Representation learning is sometimes explicitly pursued by models such as autoencoders [18] or embedding models [14, 17], while it may also implicitly make an effect in many applications such as multimodal learning [19], recommender systems [20] and anomaly detection [21].

Although representation learning lays the foundation of the success of various machine learning tasks, its interpretability has not been fully exploited. The opacity of representation space originates from the fact that each of its dimensions may not have any specific meaning. Current techniques for evaluating the quality of representation learning focus on the performance of subsequent tasks such as classification accuracy and recommendation precision. However, how representation vectors distribute in the latent space, which directly affects the results of subsequent tasks, is usually opaque to users. The interpretation of representation result could benefit the application of machine learning models in several ways. On one hand, since representation learning encodes various features or different types of information into the same space, we are interested in which information sources play an important role and which ones are actually useless in the learning process. It may help us evaluate the validity of representation result and discover the potential weakness of the learning method. On the other hand, interpretation could help justify the decisions made by machine learning models. For example, in recommender systems (especially those built upon deep models [22, 23] that are usually regarded as black boxes), we may want to know why certain products are recommended to the target user.

1.1.3 The Difficulty of Designing Interpretable Models

Post-hoc interpretation methods, either explaining individual predictions or latent representations, only work on trained models. There are several limitations of post-hoc methods. First,

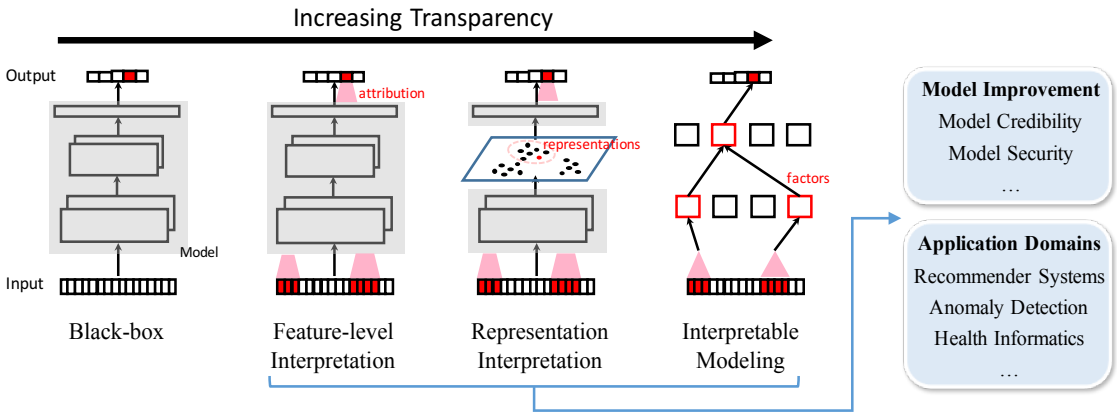


Figure 1.1: Summary of the dissertation contributions.

post-hoc interpretation suffers from the issue of explanation accuracy [24], which is undesirable especially for applications that involve high-stake decisions. Second, it is difficult to explicitly control the process of model training with only post-hoc interpretation. In many applications, such as healthcare and financial business, we have accumulated significant amount of domain knowledge that we hope to make them interact with model components. It thus would be more desirable if we are able to consider interpretability during model establishment.

1.2 Thesis Contributions

The contributions of the thesis are illustrated in Figure 1.1. Several research questions are addressed towards improving the transparency and reliability of data mining models. (Q1) How to understand which features are important in a given prediction of models? (Q2) How to understand intermediate data representations? (Q3) Instead of resorting to post-hoc methods after a black-box model is given, can we directly build models that are intrinsically interpretable? (Q4) After obtaining interpretations, how to leverage them towards constructing more reliable data mining models? The key contributions can be summarized as below.

- We propose a new local feature-level interpretation method, called Contextual Outlier Interpretation (COIN), to explain outlier detection results. We consider neighboring data points around the target outlier as its context. The interpretation is aggregated by comparing the

differences between the outlier and each cluster in the context. By clustering contexts, COIN could handle complex distributions in datasets. COIN is model-agnostic, so it can be applied to any outlier detection models.

- We propose an interpretation framework for understanding representation vectors. Instead of performing local interpretation to individual vectors, our method focuses on capturing the global characteristics of latent space. The interpretation result is presented in the form of a taxonomy. We first extract the backbone of the taxonomy to know how instances are distributed in the latent space, and then identify the important attributes in different taxonomy concepts as the description to them.

- For designing interpretable models, we propose to revise traditional network embedding models by enabling latent dimensions to encode specific meanings. We develop a polysemous network embedding method to take into account multiple facets of nodes in network data. Each facet of a node is represented using an embedding segment. The proposed method is flexible to be applied to various base embedding models without making radical changes to their base formulation. We first show how to revise Deepwalk to tackle node polysemy, and then extend our discussion to more base embedding models (e.g., graph convolutional networks) and more complex application scenarios (e.g., heterogeneous networks).

- As an exploration of utilizing interpretation to improve models, we design a novel training method for classification models under the malicious accounts detection scenario. We assume that malicious accounts will actively evolve themselves to avoid being detected, which leads to the adversarial circumstance between the classifier and the malicious accounts. We proactively simulate adversarial samples based on a set of local interpreters of the target model. The robustness of ML models is then improved through adversarial training with the adversarial samples. The efficiency of this interpretation-based attack-defense pipeline promoted by considering only the low-confidence regions of ML classifiers. Our framework is model-agnostic as the interpretation technique is general to any type of classifier. Finally,

we design a data-driven method to estimate the costs of perturbing different features.

1.3 Related Work

The work in this thesis covers several research topics, including model interpretation, outlier detection, representation learning, and adversarial machine learning.

1.3.1 Traditional Interpretation Methods

The interpretability of machine learning models receives increasing attention recently [25, 26, 27]. The efforts can be divided into two categories, post-hoc interpretation methods and intrinsically interpretable models. The post-hoc methods can be further divided into subcategories based on different metrics. For examples, according to the interpretation scope, there are global and local methods, where the former targets to transform black-box models into understandable structured systems [28], and the latter tries to explain individual predictions [29, 9]. According to the applied technique, there are gradient-based methods that measure feature contributions on output or intermediate neurons through backpropagation [9], perturbation-based methods that examine the influence of output via input changes [30], approximation-based methods that approximate complex models with simpler ones [29], and entropy-based methods that analyze model inference with information theories [31]. Intrinsically interpretability focuses on developing transparent model components. A typical strategy is to pose regularization on parameters to conform with certain prior, such as promoting independence between latent dimensions in order to let each dimension learn a unique concept [32]. [33] proposes to incorporate expert knowledge to penalize model weights. [34] develops disentangled MLP models to explicitly regulate feature interactions between layers.

1.3.2 Outlier Detection

Many outlier detection approaches have been developed over the past decades. These approaches can be divided into three categories: density-based, distance-based and model-based approaches. Some notable density-based detection methods include [35, 36, 37, 38, 39]. Representative distance-based approaches include [40, 41, 42, 43, 44]. For model-based approaches,

some well-known examples are [45, 46, 47]. Varios approaches have been proposed to tackle the challenges including the curse of dimensionality [36, 48, 49], the massive data volumn [41, 43], and heterogenous information sources [39, 50]. Ensemble learning, which is widely used in supervised learning settings, can also be applied for outlier detection with non-trivial improvements in performance [51, 52]. [53] combines results from multiple outlier detectors, each of which apply only a subset of features. In contrast, each individual detector can subsamples data instances to form a ensemble of detectors [51]. Some recent work starts to realize the importance about the explanations of detection results. In heterogeneous network anomaly detection, [39, 54, 55, 52] utilize attributes of nodes as auxiliary information for explaining the abnormality of resultant anomaly nodes. The motivation of this work is different from them, as we try to infer the reasons that why the given outliers are regarded as outlying, instead of developing new detection methods.

1.3.3 Representation Learning

Representation learning maps data instances into an expressive and usually low-dimensional latent space, where similar instances are mapped close to each other in the latent space [56]. Representation learning is fundamental to the success of the downstream machine learning models, especially when dealing with data of raw formats such as images [13], texts [14, 15] and networks [16, 17]. Some models such as autoencoders [18] or embedding models [14, 17] explicitly aim at learning the representation of input data, while many applications such as multimodal learning [19], recommender systems [20] and anomaly detection [21] rely on effective representations for improving performance although they may not pursue learning representations as the goal. The diversity of the application scenarios and model structures put forward high request on the flexibility of the corresponding interpretation framework.

Representation learning on network data is also called network embedding. Network embedding models have received a lot of attention recently. Existing work on network embedding can be categorized based on various criteria. Many basic models such as Deepwalk [17], LINE [16], node2vec [3] and SDNE [57] focus on analyzing plain networks, while more recent work starts tackling more complex networks such as considering attributes [58, 59], community structures [60]

and node heterogeneity [61, 62]. Many methods solve the embedding problem by resorting to matrix factorization [63], either explicitly [64] or implicitly [17, 16], while recently feed-forward modules such as graph convolutional networks [65, 66] are attracting more and more attention. In addition, some challenges that have been tackled in network embedding include model scalability improvement [67], modeling dynamic networks [68], considering human-in-the-loop scenarios [69]. Finally, network embedding has been applied in applications such as recommender systems [70], fraud detection [71] and behavioral modeling [72].

1.3.4 Adversarial Attacks and Defenses

Adversarial classification has been investigated using game theories, where adversaries and the classifier act as opposite players [73, 74]. Typically, the single-shot version of adversarial classification game is considered, and an equilibrium is achieved to obtain the solution [75]. A problem with these methods is that they are usually designed specifically for certain type of classifiers. As deep learning (DL) has become popular on many ML problems, recent studies show that DL is vulnerable to adversarial samples [76, 77, 78]. Fast gradient sign method is one of the most popular approaches for creating adversarial instances [79, 80]. Some other attacking methods include box-constrained L-BFGS method [76], iterative least likely method [81], Jacobian-based saliency map attack [82], adversarial samples generation based on an ensemble of deep models [83]. Some typical defensive methods include defensive distillation [84], region-based classification [85] and feature squeezing [86].

2. LOCAL INTERPRETATION FOR OUTLIER DETECTION [1]

Outliers refer to isolated instances with rare patterns in a dataset [87]. Outlier detection is a crucial fundamental task in many data-driven applications such as misinformation detection [88], fraud detection [89] and change detection [90]. However, one challenge in deploying outlier detection systems to real applications is the lack of interpretability in detection results, which will bring two problems. First, many outlier detection tasks involve high-stake decisions, such as in misinformation and fraud detection, where administrators need to decide whether to take actions against entities recognized as outlying. It is difficult to make reliable decisions without truly understanding the reasons why these entities are regarded as outliers. Second, it is hard to obtain a large number of human labels in outlier detection, which will then set obstacles in evaluating outlier detection performance. The interpretation would provide useful information to help humans decide whether the detection results are accurate.

One straightforward way for outlier interpretation is to apply feature selection to identify a subset of features that distinguish outliers from normal instances [91, 92, 93], or to apply existing post-hoc interpretation methods such as LIME [11]. However, first, it is difficult for some existing methods to efficiently handle datasets of large size or high dimensions, or effectively obtain interpretations from complex data types and distributions. Second, there is no explicit classification boundary, so interpretation methods such as LIME cannot be directly applied.

In this work, we propose a novel local interpretation method for outlier detection. The interpretation identifies those important features responsible for making instances to be outliers. Our method could work for datasets with large sizes or complex distributions.

[1] Part of this chapter is reprinted with permission from "Contextual outlier interpretation" by Ninghao Liu, Donghwa Shin, and Xia Hu, Proceedings of the 27th International Joint Conference on Artificial Intelligence, Pages 2461–2467, Copyright 2018 by the IJCAI Organization.

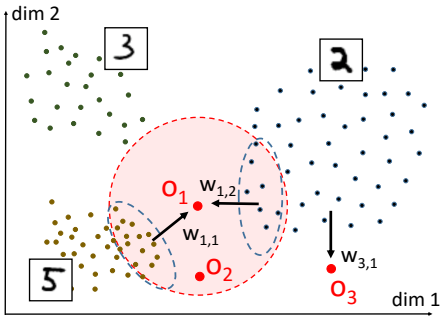


Figure 2.1: A toy example of outlier interpretation after resolving its context into clusters. Reprinted from [1].

2.1 Problem Statement

We first introduce the notations and define the outlier interpretation problem. Given a dataset $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^M | i \in [1, N]\}$ and the query outliers $\mathcal{O}$ detected therefrom, the *interpretation* for each outlier $\mathbf{o}_i \in \mathcal{O}$ is defined as a composite set: $\mathcal{E}_i = \{\mathcal{A}_i, d(\mathbf{o}_i), \mathcal{C}_i = \{\mathcal{C}_{i,l} | l \in [1, L]\}\}$. Here $\mathcal{C}_i$ denotes the *context* (i.e., k-nearest normal instances) of the outlier, $\mathcal{C}_{i,1}, \mathcal{C}_{i,2}, ..., \mathcal{C}_{i,L}$ are clusters in $\mathcal{C}_i$, and L is the number of clusters. $\mathcal{A}_i$ represents the abnormal attributes of $\mathbf{o}_i$ in contrast to $\mathcal{C}_i$. $d(\mathbf{o}_i) \in \mathbb{R}_{\geq 0}$ is the outlierness score of $\mathbf{o}_i$. The reason for clustering the context is illustrated in Figure 2.1. There are three clusters, each of which represents images of a digit. Red points are the detected outliers. Clusters of digit "2" and "5" compose the context of outlier $\mathbf{o}_1$. The interpretation of $\mathbf{o}_1$ can be obtained by comparing it with the two clusters respectively. However, it would be difficult to explain the outlierness of $\mathbf{o}_1$ if clusters of digit "2" and "5" are not differentiated.

2.2 The Contextual Outlier INterpretation (COIN) Framework

We illustrate the overall framework of Contextual Outlier INterpretation (COIN) in Figure 2.2. Given a dataset $\mathcal{X}$ and a set of outliers $\mathcal{O}$, we first map the interpretation task to a classification problem. Then, the classification problem over the whole dataset is partitioned to a series of regional problems around each outlier query. After that, a collection of simple and local interpreters g are built around each outlier. Finally, interpretation including the outlying attributes and outlier-

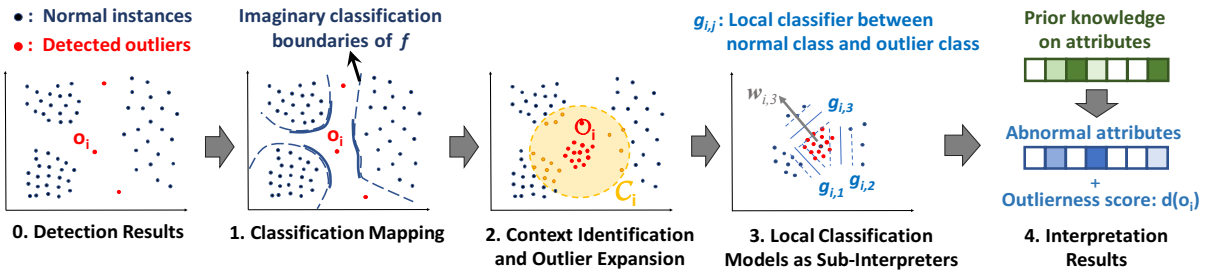


Figure 2.2: The framework for Contextual Outlier Interpretation. Reprinted from [1].

ness score can be obtained from the parameters of regional classification models. The details of each step are discussed in the following subsections.

2.2.1 Explaining Outlier Detection With Classification

To begin with, we establish the correlation between outlier detection and traditional supervised classification problems. Formally, an outlier detector can be denoted as $h(\mathbf{x}|\theta, \mathcal{X})$, where θ denotes the parameters. Here $\mathcal{X}$ is also treated as parameters since data instances affect the outlierness of each other. The abnormality of input $\mathbf{x}$ is typically represented by either a binary or continuous score, while the latter case can be easily transformed to the former if a threshold is set to separate inliers and outliers. This motivates us to explain outlier detectors using classification models. Although outlier detection is usually tackled as an unsupervised learning problem, there exists an imaginary hyperplane specified by certain decision function $f(\mathbf{x}|\theta') : \mathbb{R}^M \rightarrow \{0, 1\}$ that separates outliers from normal instances. Here θ' represents the parameters of f. An example can be found in Step 1 of Figure 2.2. Blue points and red points are normal instances and outliers, respectively, while dotted curves indicate the decision boundaries. The problem of building the decision function f is formulated as below,

$$\min_{f} \ \mathcal{L}(h, f; \mathcal{O}, \mathcal{X} - \mathcal{O}), \tag{2.1}$$

where $\mathcal{L}$ is the loss function including classification error and regularization terms. $\mathcal{O}$ and $\mathcal{X} - \mathcal{O}$ represent outlier class and inlier class, respectively. The difficulty of interpretation is that we do not have a concrete formula for f, so traditional interpretation methods for classifiers such as LIME [11] cannot be used. Meanwhile, as the boundary of f could be complex, traditional methods that linearly approximate functions suffer from interpretation accuracy.

To tackle the challenge, by utilizing the isolation property of outliers, we can further decompose the problem in Equation (4.1) into multiple regional tasks of explaining individual outliers:

$$\begin{aligned} &\min_f \mathcal{L}(h, f; \mathcal{O}, \mathcal{X} - \mathcal{O}) \Rightarrow \min_f \sum_i \mathcal{L}(h, f; \mathbf{o}_i, \mathcal{C}_i) \\ &\Rightarrow \sum_i \min_{g_i} \mathcal{L}(h, g_i; \mathbf{o}_i, \mathcal{C}_i) \Rightarrow \sum_i \min_{g_i} \mathcal{L}(h, g_i; \mathcal{O}_i, \mathcal{C}_i). \end{aligned} \tag{2.2}$$

In this way, the original problem is transformed to explaining each outlier $\mathbf{o}_i$ with respect to its context counterpart $\mathcal{C}_i$. Note that it is computationally efficient, given that the number of outliers is usually small. Here g_i represents the local parts of f exclusively for classifying $\mathbf{o}_i$ and $\mathcal{C}_i$. In Figure 2.2, for example, g_i is highlighted by the bold boundaries around $\mathbf{o}_1$ in Step 1, and $\mathcal{C}_i$ consists of the normal instances enclosed in the yellow circle in Step 2. Since there is a data imbalance between the two classes, we adopt synthetic sampling [94] to expand $\mathbf{o}_i$ to an outlier class $\mathcal{O}_i$ with comparable size to $\mathcal{C}_i$. Local interpretation, encoded in g_i, can be obtained by approximating the local behavior of h between $\mathcal{O}_i$ and $\mathcal{C}_i$.

2.2.2 Resolving Context for Outlier Explanations

In this subsection, we focus on interpreting each single outlier $\mathbf{o}_i$ by solving g_i, from which we can extract interpretation results. Let $p_{\mathcal{O}_i}(\mathbf{x})$ and $p_{\mathcal{C}_i}(\mathbf{x})$ denote the probability density functions of the outlier class and inlier context class, respectively. Since the context $\mathcal{C}_i$ may contain complex cluster structures as shown in Figure 2.1, it is difficult to directly measure the degree of separation between $\mathcal{O}_i$ and $\mathcal{C}_i$ or to discover the attributes that discriminate the two classes. Therefore, we further decompose $\mathcal{L}(h, g_i; \mathcal{O}_i, \mathcal{C}_i)$ to a set of simpler problems. According to Bayesian decision

theory, the error of classifying between $\mathcal{O}_i$ and $\mathcal{C}_i$ is

$$
\begin{aligned}
P^{err}(\mathcal{O}_i, \mathcal{C}_i) = \; & P(\mathcal{O}_i) \int_{\mathcal{C}_i} p(\mathbf{x}|\mathcal{O}_i) d\mathbf{x} + P(\mathcal{C}_i) \int_{\mathcal{O}_i} p(\mathbf{x}|\mathcal{C}_i) d\mathbf{x} \\
\approx & \big(\sum_{l \in [1,L]} P(\mathcal{O}_i) \int_{\mathcal{C}_{i,l}} p(\mathbf{x}|\mathcal{O}_i) d\mathbf{x} \big) + \big(\sum_{l \in [1,L]} P(\mathcal{C}_{i,l}) \int_{\mathcal{O}_i} p(\mathbf{x}|\mathcal{C}_{i,l}) d\mathbf{x} \big) \\
= & \sum_{l \in [1,L]} \big(P(\mathcal{O}_i) \int_{\mathcal{C}_{i,l}} p(\mathbf{x}|\mathcal{O}_i) d\mathbf{x} + P(\mathcal{C}_{i,l}) \int_{\mathcal{O}_i} p(\mathbf{x}|\mathcal{C}_{i,l}) d\mathbf{x} \big) \\
\approx & \sum_{l \in [1,L]} P^{err}(\mathcal{O}_{i,l}, \mathcal{C}_{i,l}).
\end{aligned} \tag{2.3}
$$

Suppose we can split the context $\mathcal{C}_i$ into multiple clusters $\{\mathcal{C}_{i,l} | l \in [1, L]\}$ that are well separated from each other, then each term in the summation can be treated as an independent subproblem without mutual inference. $\mathcal{O}_{i,l}$ is a subset of $\mathcal{O}_i$ close to $\mathcal{C}_{i,l}$. By combining Equation (2.2) and Equation (2.3), our final interpretation task is formulated as:

$$
\min_f \mathcal{L}(h, f; \mathcal{O}, \mathcal{X} - \mathcal{O}) \Rightarrow \min_{g_{i,l}} \sum_i \sum_l \mathcal{L}(h, g_{i,l}; \mathcal{O}_{i,l}, \mathcal{C}_{i,l}). \tag{2.4}
$$

By now we are able to classify $\mathcal{O}_{i,l}$ and $\mathcal{C}_{i,l}$ with a simple and explainable model $g_{i,l}$ such as linear models and decision trees, where the abnormal attributes $\mathcal{A}_{i,l}$ can be extracted from *model parameters*. The overall interpretation for $\mathbf{o}_i$ is obtained by integrating the results across all $\mathcal{C}_{i,l}$, $l \in [1, L]$. The estimated time complexity for implementing the framework above is $O(|\mathcal{O}| \times L \times T_g)$, where T_g is the average time cost of constructing $g_{i,l}$. Due to the scarcity of outliers, $|\mathcal{O}|$ is expected to be small. Each $g_{i,l}$ involves $\mathcal{O}_{i,l}$ and $\mathcal{C}_{i,l}$. T_g is also expected to be small since $\mathcal{C}_{i,l}$ and $\mathcal{O}_{i,l}$ are of small sizes. Moreover, the interpretation processes of different outliers are independent of each other, thus can be implemented in parallel to further reduce the time cost.

2.3 Distilling Interpretation from Classifiers

We have introduced the general framework of COIN as mapping outlier interpretation into a collection of classification tasks around each individual outlier. In this section, we propose a

concrete modeling method to explain each outlier, including discovering its abnormal attributes and measuring the outlierness score.

2.3.1 Context Identification and Clustering

Given an outlier $\mathbf{o}_i$ spotted by the detector h, first we need to identify its context $\mathcal{C}_i$ in the data space. As introduced in Section 2, $\mathcal{C}_i$ consists of the nearest neighbors of $\mathbf{o}_i$. Here we use Euclidean distance as the point-to-point distance measure. The neighbors are chosen only from normal instances. The instances in $\mathcal{C}_i$ are regarded as the representatives for the local background around the outlier. Although $\mathcal{C}_i$ contains only a small number of data instances compared to the size of the whole dataset, they constitute the border regions of the inlier class and thus are adequate to discriminate between inlier and outlier classes, as shown in the Step 2 of Figure 2.2.

As local context may indicate some interesting structures (e.g., instances with similar semantics are located close to each other in the attribute space), we further segment $\mathcal{C}_i$ into multiple disjoint clusters. To determine the number of clusters L in $\mathcal{C}_i$, we adopt the measure of *prediction strength* [95] which shows good performance even when dealing with high-dimensional data. After choosing the value of L, common clustering algorithms such as K-means or hierarchical clustering are applied to divide $\mathcal{C}_i$ into multiple clusters as $\mathcal{C}_i = \{\mathcal{C}_{i,1}, \mathcal{C}_{i,2}, \cdots, \mathcal{C}_{i,L}\}$. Clusters of small size, i.e., $|\mathcal{C}_{i,l}| \leq 0.03 \cdot |\mathcal{C}_i|$, are abandoned in subsequent procedures.

2.3.2 Maximal-Margin Linear Explanations

The model $g_{i,l}$ should have the following properties. First, it is desirable to keep $g \in G$ simple in form. For example, we may expect the number of non-zero weights to be small for linear models, or the rules to be concise in decision trees [11]. Here we let $g \in G$ belong to linear models, i.e., $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. We impose the l_1-norm constraint on $\mathbf{w}$, where attributes a_m that correspond to large $|w[m]|$ values are regarded as abnormal. Second, since outliers are usually highly isolated from their context, there could be multiple solutions all of which could classify the outliers and inliers almost perfectly, but we want to choose the one that best reflects such isolation property. This motivates us to choose l_1 norm support vector machine [96] to build g. The local

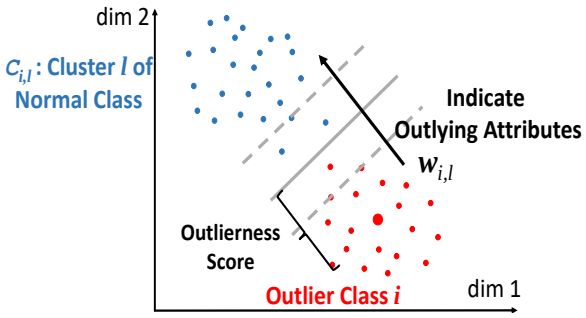


Figure 2.3: Outlier interpretation from SVM parameters. Reprinted from [1].

loss $\mathcal{L}(h, g_{i,l}; \mathcal{O}_{i,l}, \mathcal{C}_{i,l})$ to be minimized in Equation (2.4) is thus as below:

$$\begin{aligned} &\sum_{n=1}^{N_{i,l}} (1 - y_n g(\mathbf{x}_n) - \xi_n)_+ + c \sum_{n=1}^{N_{i,l}} \xi_n, \\ \text{s.t.} \quad &\xi_n \geq 0, \quad \|\mathbf{w}\|_1 \leq b \end{aligned} \tag{2.5}$$

where $N_{i,l} = |\mathcal{O}_{i,l} \cup \mathcal{C}_{i,l}|$, $(.)_+$ is the hinge loss, ξ_n is the slack variable, b and c are the parameters. Here $y_n = 1$ if $\mathbf{x}_n \in \mathcal{C}_{i,l}$ and $y_n = -1$ if $\mathbf{x}_n \in \mathcal{O}_{i,l}$.

The abnormal attributes and the outlierness score could be estimated from the parameters of the local model $g_{i,l}$. Let $\mathbf{w}_{i,l}$ denote the weight vector of $g_{i,l}$, the importance of attribute a_m with respect to the context of $\mathcal{C}_{i,l}$ is thus defined as $s_{i,l}(a_m) = |w_{i,l}[m]|/\gamma_{i,l}^m$. Here $\gamma_{i,l}^m$ denotes the resolution of attribute a_m in $\mathcal{C}_{i,l}$, i.e., the average distance along the m^{th} axis between an instance in $\mathcal{C}_{i,l}$ and its closest neighbors. The overall score of a_m for $\mathbf{o}_i$ is

$$s_i(a_m) = (1/|\mathcal{C}_i|) \sum_l |\mathcal{C}_{i,l}| s_{i,l}(a_m), \tag{2.6}$$

which is the weighted average score for a_m over all L clusters. Attributes a_m with large $s_i(a_m)$ are regarded as the abnormal attributes for $\mathbf{o}_i$ (i.e., $a_m \in \mathcal{A}_i$). For the outlierness score $d(\mathbf{o}_i)$, we define it as:

$$d_l(\mathbf{o}_i) = |g_{i,l}(\mathbf{o}_i)|/\|\mathbf{w}_{i,l}\|_2. \tag{2.7}$$

This measure is robust to high dimensional data, as $\mathbf{w}$ is sparse and $d_l(\mathbf{o}_i)$ is calculated in a low dimensional space. An example is shown in Figure 2.3, where abnormal attributes are indicated from weight vector $\mathbf{w}$ and the outlierness score is shown. The overall outlierness score for $\mathbf{o}_i$ across all context clusters is:

$$d(\mathbf{o}_i) = (1/|\mathcal{C}_i|) \sum_l |\mathcal{C}_{i,l}|\, d_l(\mathbf{o}_i)/\gamma_{i,l}, \tag{2.8}$$

which is the weighted summation over different context clusters. Here the normalization term $\gamma_{i,l}$ is the average distance from an instance to its closest neighbor in $\mathcal{C}_{i,l}$. Now we have obtained all of the three aspects of interpretation $\mathcal{E}_i = \{\, \mathcal{A}_i, d(\mathbf{o}_i), \mathcal{C}_i = \{\mathcal{C}_{i,l} | l \in [1, L]\} \,\}$.

2.3.3 Filtering Outliers with Interpretation and Prior Knowledge

In real-world applications, the importance of different attributes varies according to different scenarios [97, 98]. Take social network spammer detection as an example. We have two account attributes: the number of followers (N_{fer}) and the ratio of tweets posted by API (R_{API}). A spammer account tends to have a small N_{fer} value as it is socially inactive, but large R_{API} to conveniently generate malevolent content. However, it is easy for spammers to intentionally increase their N_{fer} by purchasing followers, but manually decreasing R_{API} is more difficult due to expensive human labors. In this sense, R_{API} is more robust than N_{fer} in translating detected outliers as spammers. Therefore, we introduce two vectors $\boldsymbol{\beta}$ and $\boldsymbol{p}$, where $\beta_m \in \mathbb{R}_{\geq 0}$ denotes the prior knowledge about the robustness of a_m, and $p_m \in \{-1, 0, 1\}$ denotes the expected perturbation direction of a abnormal attribute. $p_m = -1$ means we expect outliers to have small value for a_m (e.g., N_{fer}), $p_m = 1$ means the opposite (e.g., R_{API}), while $p_m = 0$ means there is no preference. Thus, the outlierness score of $\mathbf{o}_i$ with respect to $\mathcal{C}_{i,l}$ is refined as:

$$d_l(\mathbf{o}_i) = \| \frac{|g_{i,l}(\mathbf{o}_i)|}{\|\mathbf{w}_{i,l}\|_2} \frac{\mathbf{w}'_{i,l}}{\|\mathbf{w}_{i,l}\|_2} \circ \boldsymbol{\beta} \|, \tag{2.9}$$

where the operator $\circ$ denotes element-wise multiplication, $w'[m] = \min(0, w[m])$ if $p_m = 1$, and $w'[m] = \max(0, w[m])$ if $p_m = -1$. If we label outliers with 1 and inliers with -1, the sign of $\mathbf{p}$ is reversed. The reason for introducing $\mathbf{w}'$ is that, if interpretation does not conform with the prior

knowledge, such as an outlier in spammer detection is interpreted as having low R_{API}, then the outlierness score of the outlier should be deducted.

2.4 Experiments

In this section, we present experiment results to assess the effectiveness of our framework. We answer the following questions: 1) How accurate is the proposed framework in identifying abnormal attributes of given outliers? 2) Can we accurately measure the outlierness score of outliers? 3) How effective is the prior knowledge of attributes in refining outlier detection results?

2.4.1 Datasets

We use both real and synthetic datasets in experiments. We follow the procedures in [99] and create two synthetic datasets with ground-truth abnormal attributes for each outlier. In the first synthetic dataset, each outlier is close to only one normal cluster and far away from the others. In the second synthetic dataset, each outlier is in the vicinity of several normal clusters simultaneously, so the scenario is more complicated. The real-world datasets used in our experiments include Wisconsin Breast Cancer (WBC) dataset [100], MNIST dataset and Twitter spammer dataset [97]. WBC dataset records the measurements for breast cancer cases with two classes, i.e. benign and malignant. The former class is considered as normal, while we downsampled 25 malignant cases as the outliers. MNIST dataset includes a collection of 28×28 images of handwritten digits. Here we use the training set which contains 42,000 examples. Instead of using raw pixels as attributes, we build a Restricted Boltzmann Machine (RBM) with 150 latent units to map images to a low-dimensional space which is more proper for interpretation than raw pixels. A multi-label logistic classifier is then built to classify digits, and the ground-truth outliers are selected as the misclassified instances downsampled to $1,000$. The Twitter dataset contains information of normal users and spammers crawled from Twitter. Following [97], we divide attributes into two categories according to whether they are robust to the spammers in disguise. Attributes of low robustness refer to those which can be easily controlled by spammers to avoid being detected, while attributes of high robustness are the opposite.

	SYN1	SYN2	WBC	Twitter	MNIST
N	405	405	458	11,000	42,000
M	15	15	9	16	150
$\|\mathcal{O}\|$	30	30	25	1,000	1,000

Table 2.1: Details of the datasets in experiments. Reprinted from [1].

2.4.2 Baseline Methods

We include several recent interpretation methods for outlier detection or classification as baseline methods for comparison:

- CA-lasso (CAL) [92]: An interpretation method that analyzes the separability between outlier and inliers as a linear classification problem solved with LASSO, without further clustering the context of outliers.
- IPS-BS [101]: An interpretation method that applies isolation path score to measure outlierness. Beam Search is then applied to look for the abnormal attributes.
- LIME [11]: A global classifier is first constructed to classify outliers and inliers. Then the abnormal attributes for each outlier is identified by locally interpreting the classification model around the outlier. A neural network is used as the global classifier for MNIST data, and SVMs with RBF kernel are used for other datasets.

2.4.3 Abnormal Attributes Evaluation

The goal of this experiment is to measure the correctness of identified attributes in explaining the outliers. Since ground-truth abnormal attributes of real-world datasets are not available, we utilize synthetic data generation and append M Gaussian-noise attributes to all real-world data instances. Noise attributes are not expected to be identified as abnormal as they are of small magnitudes. In our experiments, we choose $0.08 \times N$ nearest neighbors of an outlier $\mathbf{o}_i$ as its context $\mathcal{C}_i$. The radius of synthetic sampling for building the outlier class $\mathcal{O}_i$ is set as half of the average distance to the inlier class $\mathcal{C}_i$ to avoid overlap between $\mathcal{O}_i$ and $\mathcal{C}_i$. The parameters of

	COIN			CAL			IPS-BS			LIME		
	Prec	Rec	F_1	Prec	Rec	F_1	Prec	Rec	F_1	Prec	Rec	F_1
SYN1	**0.97**	**0.89**	**0.93**	0.89	0.81	0.84	0.87	0.44	0.58	0.82	0.79	0.80
SYN2	**0.99**	**0.90**	**0.94**	0.92	0.70	0.80	**1.00**	0.37	0.54	0.91	0.70	0.79
WBC	0.86	0.37	**0.52**	0.84	0.37	0.51	**0.90**	0.15	0.26	0.35	**0.39**	0.37
Twitter	**0.91**	0.33	0.48	0.75	0.34	0.47	0.72	0.29	0.41	0.60	**0.67**	**0.63**

Table 2.2: Performance of abnormal attributes identification. Reprinted from [1].

SVMs are tuned by validation, where some samples from $\mathcal{O}_i$ and $\mathcal{C}_i$ are randomly selected as the validation set. The same parameter values are used for all outliers in the same dataset. We report the Precision, Recall and F_1 score averaged over all the outliers in Table 2.2. Besides finding that COIN shows relatively better performance, some observations can be made as follows:

- In general, the Recall value of SYN2 is lower than that of SYN1, because the context of each outlier in SYN2 has several clusters, and the true abnormal attributes vary among different clusters. In this case, retrieving all ground-truth attributes is more challenging.
- IPS-BS is more cautious in making decisions. It tends to stop early if the discovered abnormal attributes already make the outlier query well isolated. Therefore, IPS-BS has high Precision, but only a small portion of true attributes are discovered (low Recall).
- The Recall scores are low for real-world data since we treat all original attributes to be the ground truth, so low Recall values do not necessarily mean bad performances.

2.4.4 Outlierness Score Evaluation

We evaluate if interpretation methods are able to accurately measure the outlierness score of outlier queries. For each dataset, we randomly sample the same number of inliers as the outliers, and use them together as queries to interpreters. The label is 1 for each true outlier, and 0 for each inlier. For each query, interpreters are asked to estimate its outlierness score. After that, we rank the instances in a descending order with respect to their outlierness scores. Since true outliers are more isolated, an effective interpreter should convert such isolation degree to larger scores.

AUC	SYN1	SYN2	WBC	Twitter	MNIST
COIN	0.78	0.93	0.96	0.85	0.87
CAL	0.71	0.63	0.94	0.81	0.76
IPS_BS	0.69	0.91	0.90	0.79	0.82
LIME	0.74	0.62	0.94	0.83	0.78

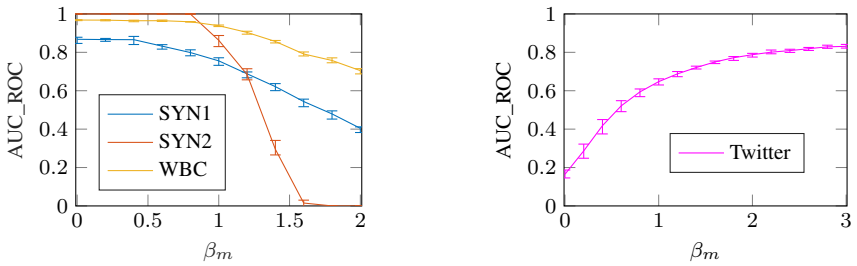
Table 2.3: Outlierness score ranking performance. Reprinted from [1].

We report the results in Table 2.3 with AUC as the evaluation metric. The proposed method achieves better performance than the baseline methods especially on SYN2 and MNIST. This can be explained by the more complex structures in these datasets, where an outlier may be close to several neighboring clusters. COIN resolves the contextual clusters around each outlier, so it can better handle such scenario. This also explains why IPS_BS is also more effective in complex datasets than the other two baseline methods. The isolation tree used in IPS_BS can handle complex cluster structures.

2.4.5 Filtering Outliers with Prior Knowledge

In this experiment, we discuss if interpretations, together with prior knowledge, can help filtering existing outliers to satisfy the demand of specific applications.

The experiment has two parts. In the first part, we append M new noise attributes to data instances, so each instance is augmented to $\mathbf{x} \in \mathbb{R}^{2M}$. Different from the noise attributes in Section 2.4.3 that are of small magnitude, the attributes here may turn inliers to “outliers". However, these new outliers are irrelevant to the ground truth. We sample $0.5 \times |\mathcal{O}|$ inliers, together with ground-truth outliers, as queries fed into COIN. We set $\boldsymbol{p}$ to be zero and run COIN with different $\boldsymbol{\beta}$ values. The weights corresponding to original attributes are fixed to 1 ($\beta_m = 1, m \in [1, M]$), and we only vary the weights of noise attributes ($\beta_m = \beta, m \in [M+1, 2M]$). Similar to Section 2.4.4, we obtain the outlierness score for all queries and rank them in a descending order according to the score. Ground-truth outliers are expected to rank higher. The ranking performance is reported in Figure 2.4a. The plot indicates that as we increase the weights of noise attributes, the performance of the interpreter degrades for all datasets, because it is more difficult to distinguish between real



(a) Data with noise attributes.

(b) Twitter spammer data.

Figure 2.4: The influence of the prior knowledge on outlierness score. Results averaged over 20 runs, bars depict 25-75%. Reprinted from [1].

outliers and noisy instances. From the opposite perspective, assigning large weights to important attributes will filter out mis-detected outliers.

The second part of the experiment uses the Twitter dataset in which features extracted from user profiles, posts and graph structures are used as attributes. According to [97], the robustness level varies for different attributes. Some attributes, such as the number of followers, hashtage ratio and reply ratio, can be easily controlled by spammers to avoid being captured, so they are of low robustness, while some other attributes such as account age, API ratio and URL ratio have high robustness. In this experiment, we fix the weight of low-robustness attributes to 1, and vary the weight β_m of high-robustness attributes. The remaining procedures are the same as first part of the experiment discussed above. The result of outlierness ranking is reported in Figure 2.4b. The rising curve shows that as more emphasis is put on high-robust attributes, we are able to refine the performance of spammer identification. The experiment result indicates that by resorting to the interpretation of outliers, we can gain more insights on their characteristics, and adaptively select those that are in accordance with the specific application.

2.4.6 Case Studies

At last, we conduct some case studies to illustrate interpretation results. The MNIST dataset is used here since images are more intuitive for understanding. The attributes are the hidden features extracted by the RBM as pre-processing instead of using raw pixels. The latent features learned

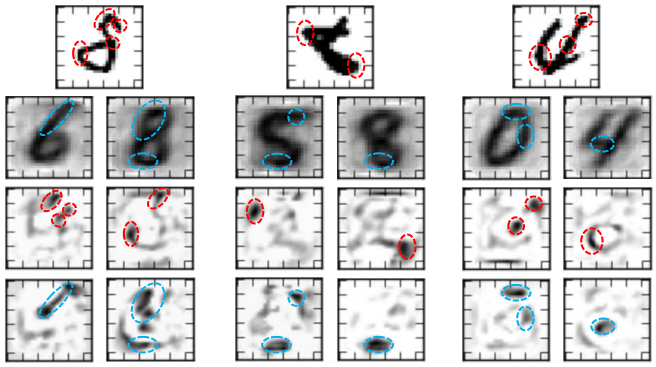


Figure 2.5: Visualization of outlier interpretation on MNIST dataset. Reprinted from [1].

from RBM can be seen as simple primitives that compose more complicated visual patterns. They are more suitable for interpretation than raw pixels as they are in accordance with the cognitive habits of humans, where we tend to rely on richer representations for explanation and action.

The case study results are shown in Figure 2.5. There are three query outlier images shown in the first row. We choose two neighboring clusters for each query, and compute the average image of each cluster, as shown in the second row. The average images can be seen as part of the contexts of outliers. Clear handwritten digits can be seen from the average images, so that the clusters are internally coherent. The third and fourth rows together indicate the noteworthy attributes of the query image with respect to the corresponding average images. The black strokes enclosed by red circles in third-row images represent positive abnormal attributes, i.e., the query image is regarded as an outlier instance because it possesses these attributes. The strokes enclosed by blue circles in fourth-row images are negative abnormal attributes, as the query outlier digit does not include them. These negative attributes, however, commonly appear in the neighbor images of the outlier. The positive and negative attributes together explain why the outlier image is different from its nearby normal images.

3. REPRESENTATION INTERPRETATION VIA TAXONOMY INDUCTION [1]

The local feature-based interpretation discussed in the last section identifies the end-to-end correlation between output and input features. However, this type of interpretation does not actually gain insights of the internal working mechanism of the model. A crucial perspective of understanding how models work is to examine the latent representations of input fed into the model. However, latent representations are difficult to be directly understood, since each dimension of the latent space may not have any concrete meaning. Meanwhile, the existing local feature-based interpretation methods cannot well solve the problem. First, the interpretation of representations requires a global understanding of the latent space, which cannot be handled by local interpretation methods. Second, there lacks class label information in representations, while most local interpretation methods only work on classification models.

In this work, we propose a novel interpretation method for understanding the latent representations learned by models. The interpretation outcome is a taxonomy. The hierarchical cluster structure provides a global description of the distribution of representation vectors, while the properties of each cluster are specified by the representative attributes of the instances within the cluster.

3.1 Problem Statement

We first introduce the notations and the problem definition. We use boldface uppercase alphabets (e.g., $\mathbf{A}$) to denote matrices, boldface lowercase alphabets (e.g., $\mathbf{z}$) as vectors, calligraphic alphabets (e.g., $\mathcal{T}$) as sets, and normal characters (e.g., i, K) as scalars. The size of a set $\mathcal{T}$ is denoted as $|\mathcal{T}|$. The i-th row, j-th column and (i, j) entry of matrix $\mathbf{A}$ are denoted as $\mathbf{A}_{i,:}$, $\mathbf{A}_{:,j}$ and $\mathbf{A}_{i,j}$, respectively. Let $\mathbf{X}$ be the matrix of input features fed into models. $\mathbf{X} \in \mathbb{R}^{N \times M}$, where N is the number of instances and there are M features. The output is a representation matrix $\mathbf{Z} \in \mathbb{R}^{N \times D}$, where $\mathbf{Z}_{i,:} \in \mathbb{R}^{D}$ denotes the *representation vector* of the i-th instance. In many appli-

[1] Part of this chapter is reprinted with permission from "On interpretation of network embedding via taxonomy induction" by Ninghao Liu, Xiao Huang, Jundong Li, and Xia Hu, Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Pages 1812-1820, Copyright 2018 by Association for Computing Machinery.

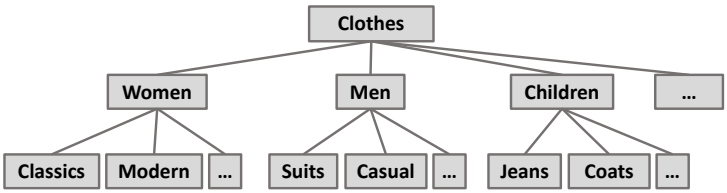


Figure 3.1: A toy example of taxonomy for "clothes". Reprinted from [2].

cations, representation learning is also called embedding, so we use representation and embedding interchangeably throughout this work.

In this work, we tackle the problem of explaining representation learning via taxonomy induction. A taxonomy is a structured organization of knowledge to facilitate the information searching. An example of taxonomy for clothes is shown in Figure 3.1, where different classes are organized in a hierarchical structure, and classes of coarse granularity are gradually split into refined ones. The backbone of a taxonomy is usually a hierarchy of "concepts", which correspond to clusters implicitly contained in $\mathbf{Z}$. Each concept owns certain characteristics shared by the representation vectors within it. The "hypernym" relation in a hierarchy is modeled by the directed link between a parent concept and child concept. Based on this, we propose to perform taxonomy induction in two steps. The first step is to extract the taxonomy backbone to get the hierarchy of concepts. The second step is to provide description to the extracted concepts. In this work, we also call the first step as global-view interpretation, as it extracts the overall distribution of representations. We call the second step as local-view interpretation, as it selects the important attributes of each cluster.

3.2 Taxonomy Backbone Construction

The goal of this section is to extract the backbone of the taxonomy based on the representation vectors. Concretely, the backbone is represented as a hierarchy of clusters in the representation space. In this way, we can gain more insights about how instances distribute in the representation space, and gradually unveil the structural patterns among instances.

3.2.1 Embedding-based Graph Construction and Clustering

Given the representation matrix $\mathbf{Z}$, we first build a graph G, whose affinity matrix is denoted as $\mathbf{A}^G$, to store the node-to-node similarity in the embedded space. The edge weight between each pair of nodes is defined as:

$$\mathbf{A}^G_{i,j} = \exp(-\|\mathbf{Z}_{i,:} - \mathbf{Z}_{j,:}\|_2^2/2\sigma^2). \tag{3.1}$$

Since computing and storing weights for all pairs of instances could be extremely expensive when the number of instances is large, here we only maintain a sparse affinity matrix defined as follows:

$$\mathbf{A}_{i,j} = \begin{cases} \mathbf{A}^G_{i,j}, & \text{if } j \in \text{Neighbors}(i) \text{ or } i \in \text{Neighbors}(j) \\ 0, & \text{otherwise} \end{cases}, \tag{3.2}$$

where the number of neighbors is $|\text{Neighbors}(\cdot)| = b\lceil \log_2 N \rceil$ according to [102], and b is a constant integer. The resultant affinity matrix $\mathbf{A}$ is symmetric. The cluster structure of representation vectors is obtained by solving the problem below:

$$\min_{\mathbf{W} \geq 0} \|\mathbf{A} - \mathbf{W}\mathbf{W}^T\|_F^2, \tag{3.3}$$

where $\mathbf{W} \in \mathbb{R}^{N \times C}$, C is the number of clusters and $\|\cdot\|_F$ denotes Frobenius norm. The above formulation is equivalent to performing kernel K-means clustering on $\mathbf{Z}$ with the orthogonality constraint $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ relaxed [103]. A larger value of $\mathbf{W}_{i,c}$ indicates a stronger affiliation of the instance i to cluster c. It is a nontrivial problem to solve Equation 3.3, since the objective function is a fourth-order function which is non-convex with respect to $\mathbf{W}$, so we reformulate the problem as below:

$$\min_{\mathbf{W},\mathbf{H} \geq 0} \|\mathbf{A} - \mathbf{W}\mathbf{H}^T\|_F^2 + \alpha\|\mathbf{W} - \mathbf{H}\|_F^2, \tag{3.4}$$

where $\alpha > 0$ controls the tradeoff between approximation error and factor matrices difference. Traditional iterative optimization methods such as coordinate descent methods [104, 105] can be applied to solve the problem by updating $\mathbf{W}$ and $\mathbf{H}$ alternatively. The value of α is gradually

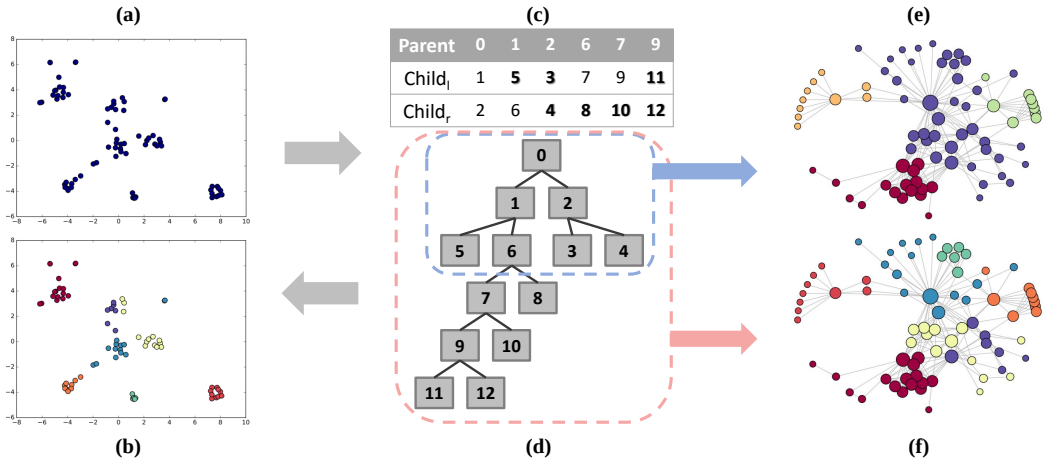


Figure 3.2: A taxonomy extracted by hierarchical clustering on embeddings from node2vec [3] on the Les-Misérables network. (a): Visualization of original embedding results. (b): Visualization of embeddings after clustering where $C = 7$. (c): Taxonomy represented in a table, where bold numbers denote leaves. (d): Taxomony represented by a tree. (e): Visualization of the network with 4 clusters obtained from a subtree. (f): Visualization of the network with 7 clusters obtained from the taxonomy, which corresponds to the embedding visualization in figure (b). Reprinted from [2].

increased to reduce the difference between **W** and **H** as iterations proceed until convergence.

3.2.2 Hierarchy Establishment from Embedding-based Graph

Instead of performing flat clustering on G, we leverage the hierarchical strategy to resolve the graph recursively. The reasons are two-fold. First, it is hard to know the optimal number of clusters in G. By adopting hierarchical clustering, we avoid starting over and exhaustively trying different K values, which is often time-consuming. Second, multilevel abstraction of concepts naturally exists in many real-world networks, such as product catalogues in e-commerce networks, and topic hierarchies in document networks. In addition, practitioners can trim the obtained hierarchical structure based on their own needs, which is helpful in interpretation where the interactions exist between users and models.

We adapt the previously mentioned NMF algorithm for clustering G in a hierarchical manner, as summarized in Algorithm 1. After initialization (line 1$\sim$2), we repeatedly split large clusters

Algorithm 1: Taxonomy Backbone Extraction from G

Input: Affinity matrix $\mathbf{A}$ of G, maximum number of clusters C, parameters $\alpha \in \mathbb{R}_+$, $\gamma \in (0, 1)$, $I \in \mathbb{N}_+$

Output: Tree-structured hierarchy $\mathcal{T}$

1 Create a root cluster $\mathcal{M}^1 = \mathcal{V}$ containing all graph nodes, and let the partition score $s(\mathcal{M}^1) \leftarrow 0$ (Note: $0 \leq s(\cdot) \leq 1$)
2 Number of leaf node $c \leftarrow 1$, time step $t \leftarrow 1$
3 **while** $c \leq C$ **do**
4 Choose a cluster $\mathcal{M}^t$ of the smallest $s(\mathcal{M}^t)$ to be partitioned
5 Outliers $\mathcal{O}^t = \emptyset$
6 **for** $i = 1 : I$ **do**
7 Obtain the submatrix $\mathbf{A}^t$ corresponding to $\mathcal{M}^t$
8 Run rank-K NMF on $\mathbf{A}^t$, and create K potential clusters $\{\mathcal{M}_k^t; k \in [1, K]\}$ based on $\mathbf{W}^t$ (or $\mathbf{H}^t$)
9 **if** $|\mathcal{M}_k^t| < \gamma|\mathcal{M}^t|$ && *$s(\mathcal{M}_k^t)$ is the largest among all leaf clusters,* $\exists k \in [1, K]$, **then**
10 $\mathcal{M}^t \leftarrow \mathcal{M}^t - \mathcal{M}_k^t$, $\mathcal{O}^t \leftarrow \mathcal{O}^t \cup \mathcal{M}_k^t$
11 **else**
12 break
13 **if** $i \leq I$ **then**
14 Partition $\mathcal{M}^t$ as $\{\mathcal{M}_k^t; k \in [1, K]\}$, and compute $s(\mathcal{M}_k^t)$
15 **else**
16 $\mathcal{M}^t \leftarrow \mathcal{M}^t \cup \mathcal{O}^t$, $s(\mathcal{M}^t) \leftarrow 1$
17 $c \leftarrow c + K$, $t \leftarrow t + 1$

into smaller ones. An example can be found in Figure 3.2. The details of the hierarchical clustering process are introduced as follows.

Partition Score (line 4): At each step t, we need to choose the "best" cluster to be partitioned. A cluster is suitable for further partitions if it contains several smaller cluster components which are densely intra-connected and loosely inter-connected. We use the normalized cut ($ncut$) [106] as the partition score $s(\cdot)$:

$$s(\mathcal{M}^t) = ncut(\{\mathcal{M}_k^t; k \in [1, K]\}) = \sum_{k=1}^{K} \frac{cut(\mathcal{M}_k^t, \mathcal{V} - \mathcal{M}_k^t)}{assoc(\mathcal{M}_k^t, \mathcal{V})}, \tag{3.5}$$

where $cut(\mathcal{M}_1, \mathcal{M}_2) = \sum_{i \in \mathcal{M}_1, j \in \mathcal{M}_2} \mathbf{A}_{i,j}$ and $assoc(\mathcal{M}, \mathcal{V}) = \sum_{i \in \mathcal{M}, j \in \mathcal{V}} \mathbf{A}_{i,j}$. In this way, if each sub-cluster $\mathcal{M}_k^t$ is well isolated from other nodes, then $cut(\mathcal{M}_k^t, \mathcal{V} - \mathcal{M}_k^t)$ will be small. Besides, if the nodes within $\mathcal{M}_k^t$ are well connected, then $assoc(\mathcal{M}_k^t, \mathcal{V})$ is large, since $assoc(\mathcal{M}_k^t, \mathcal{V}) = assoc(\mathcal{M}_k^t, \mathcal{M}_k^t) + cut(\mathcal{M}_k^t, \mathcal{V} - \mathcal{M}_k^t)$. Therefore, the cluster with smallest partition score s is

assigned with top priority to be further split.

Rank-K NMF (line 7∼8, 14): Let $\mathbf{A}^t \in \mathbb{R}^{n\times n}$ denote the affinity matrix of cluster $\mathcal{M}^t$ which is selected to be further partitioned at step t. The size of $\mathcal{M}^t$ is n. We fix the number of sub-clusters as K, so each partition step is transformed into a local rank-K NMF problem: $\min_{\mathbf{W}^t,\mathbf{H}^t\geq 0} \|\mathbf{A}^t - \mathbf{W}^t\mathbf{H}^{tT}\|_F^2 + \alpha\|\mathbf{W}^t - \mathbf{H}^t\|_F^2$. For generality, in this work we set $K = 2$ if no other prior knowledge is available. After initializing $\mathbf{H}^t$ and α, the NMF problem is solved in an iterative manner until convergence [107]:

$$\begin{aligned} &\mathbf{W}^t \leftarrow \mathbf{H}^t, \;\; \alpha \leftarrow c_{scale}\cdot\alpha \\ &\mathbf{H}^t \leftarrow \underset{\mathbf{H}^t\in\mathbb{R}^{n\times K}_{\geq 0}}{\operatorname{argmin}} \left\| \begin{bmatrix} \mathbf{A}^t \\ \sqrt{\alpha}\mathbf{W}^{tT} \end{bmatrix} - \begin{bmatrix} \mathbf{W}^t \\ \sqrt{\alpha}\mathbf{I}_K \end{bmatrix} \mathbf{H}^{tT} \right\|_F^2, \end{aligned} \tag{3.6}$$

where c_{scale} is slightly larger than 1, so that α keeps increasing throughout iterations to force $\mathbf{H}^t$ and $\mathbf{W}^t$ to be gradually close to each other. Here $\mathbf{I}_K$ is the $K \times K$ identity matrix. After convergence, we assign each node i to the new sub-cluster $k_i = \operatorname{argmax}_k \mathbf{W}_{i,k}$. In the taxonomy tree $\mathcal{T}$, the new K sub-clusters are appended as the children of $\mathcal{M}^t$.

Outliers Identification (line 9∼12, 16): Outliers adversely affect the clustering quality as they are likely to be separated as sub-clusters but usually do not contain patterns of interest. We treat a sub-cluster $\mathcal{M}_k^t$ as outliers if it is of extremely small size compared with its parent (i.e., $|\mathcal{M}_k^t| < \gamma|\mathcal{M}^t|$ and $0 < \gamma < 1$) and has high partition scores (low priority). We keep excluding outliers for at most I rounds, so that $\mathcal{M}^t$ will not be partitioned if its main components are outliers (i.e., $i \leq I$ does not hold).

3.3 Cluster Attributes Importance Measurement

After obtaining the backbone of taxonomy through hierarchical clustering, in this subsection, we concentrate on each cluster in the taxonomy to summarize its unique characteristics. Specifically, we utilize feature importance to delineate the properties of clusters. Typical examples include user interests in social networks and recommender systems, and research areas of scholars

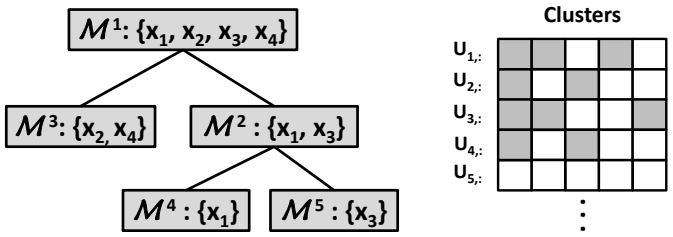


Figure 3.3: Left: A toy example of important features associated with each cluster in a subtree of the taxonomy. Right: The corresponding activated coefficients in the weight matrix $\mathbf{U}$. Reprinted from [2].

in academic networks. Our goal is to find the important features that significantly contribute to the representation learning results.

We tackle the problem of characterizing multiple clusters with important features as a multi-task feature selection problem. Each cluster is then characterized by the important features shared by the majority of instances in the cluster. The cluster structure identified by the global-view interpretation provides the discriminative information which supports feature selection to be implemented in a supervised manner. Concretely, let $\mathbf{Y} \in \mathbb{R}^{N \times C}$ denote the class label matrix, where $\mathbf{Y}_{n,t} = 1$ if node n belongs to cluster $\mathcal{M}^t$ and $\mathbf{Y}_{n,t} = 0$ otherwise. Here C is the number of tasks. If we want to describe all clusters in the taxonomy, then C equals to the total number of internal nodes and leaf nodes in the tree $\mathcal{T}$. Let $\mathbf{U} \in \mathbb{R}^{M \times C}$ denote the feature weight matrix, where $\mathbf{U}_{m,t}$ indicates the importance of feature m in cluster $\mathcal{M}^t$. For a pair of nodes i, j within a certain cluster $\mathcal{M}^t$, the consistency of node similarity in the embedding space and that in the selected feature space can be quantified as below:

$$\sigma(i,j) = \frac{1}{1 + \exp(-\mathbf{A}_{i,j}\mathbf{x}_i{}^T diag(\mathbf{U}_{:,t})\mathbf{x}_j)}, \tag{3.7}$$

where $diag(\mathbf{U}_{:,t})$ turns the column vector $\mathbf{U}_{:,t}$ into the diagonal matrix form. $\mathbf{x}_i{}^T diag(\mathbf{U}_{:,t})\mathbf{x}_j$ is the similarity between features of i and j weighted by $\mathbf{U}_{:,t}$, while $\mathbf{A}_{i,j}$ is the edge strength between

instance i and j in graph G. Then the objective function with respect to $\mathbf{U}$ is given as follows:

$$\max_{\mathbf{U}} \sum_{t=1}^{C} \Big(\sum_{i,j \in \mathcal{M}^t} \log(\sigma(i,j)) + \sum_{i^-,j^-} \log(1-\sigma(i^-,j^-)) \Big). \tag{3.8}$$

The samples i^- and j^- are negative samples from other clusters. By optimizing the above objective function, we obtain $\mathbf{U}$ that selects a subset of important features for each cluster $\mathcal{M}^t$ such that node proximity in the selected feature space are in line with the proximity in the embedding space.

To take advantage of the hierarchical structures contained in the taxonomy, some constraints are required to regularize $\mathbf{U}_{:,t}$. Let $\mathcal{P} = (t_0, t_1, t_2, ..., t_h)$ denote the path from the root to a leaf in the taxonomy tree $\mathcal{T}$, where t_i refers to a tree node and t_{i-1} is the parent of t_i. Two clusters $\mathcal{M}^{t_i}$ and $\mathcal{M}^{t_j}$ are expected to share some common characteristics if t_i and t_j lie on the same path $\mathcal{P}$. An example is shown in Figure 3.3. For $\mathcal{M}^5$, its important feature is x_3, while x_3 is also one of the important features of $\mathcal{M}^2$ and $\mathcal{M}^1$ as they are in the same root-to-leaf path. To incorporate the prior knowledge of tree-based structure in taxonomy, we introduce tree-based group lasso regularization [108] to the objective function. Therefore, the overall objective function for local-view interpretation is formulated as:

$$\begin{aligned} \max_{\mathbf{U}} & \sum_{t=1}^{C} \Big(\sum_{i,j \in \mathcal{M}^t} \log(\sigma(i,j)) + \sum_{i^-,j^-} \log(1-\sigma(i^-,j^-)) \Big) \\ & - \lambda \sum_{m} \sum_{\tau \in \mathcal{T}} l_\tau \|\mathbf{U}_m^\tau\|_2. \end{aligned} \tag{3.9}$$

For the regularization term, $\mathbf{U}_m^\tau$ is a vector of weight coefficients $\{\mathbf{U}_{m,t} : t \in \tau\}$, where τ refers to a tree node in taxonomy $\mathcal{T}$. Here $t \in \tau$ if the cluster $\mathcal{M}^t$ is part of $\mathcal{M}^\tau$. This also implies that if $t \in \tau$, then $t \in parent(\tau)$. Considering the tree structure, one way for selecting negative samples i^- and j^-, w.r.t. $i, j \in \mathcal{M}^t$, is to sample from clusters that are not in the same path as $\mathcal{M}^t$.

The above optimization problem is non-smooth due to the existence of L_1/L_2-norm regularization term and is hence difficult to optimize. Therefore, we use an alternative formulation previously

Algorithm 2: Optimization algorithm for Equation 3.10.

Input: Affinity matrix $\mathbf{A}$ of G, feature matrix $\mathbf{X}$, hierarchical clusters $\mathcal{T}$, $\lambda \in \mathbb{R}_+$.
Output: Weight matrix $\mathbf{U}$, coefficients $\{g_{m,\tau}\}$.

1 Initialize $\mathbf{U}$ and $\{g_{m,\tau}\}$
2 **while** *not converged* **do**
3 **for** $t = 1, 2, ..., C$ **do**
4 Collect samples $\{(i,j)|i,j \in \mathcal{M}^t\}$, and negative samples $\{(i^-, j^-)\}$ from other clusters
5 Update $\mathbf{U}_{:,t}$ using mini-batch stochastic gradient descent
6 Update $g_{m,\tau} = \frac{l_\tau \|\mathbf{U}_m^\tau\|_2}{\sum_m \sum_{\tau \in \mathcal{T}} l_\tau \|\mathbf{U}_m^\tau\|_2}$

introduced for group lasso [109]:

$$\begin{aligned} \max_{\mathbf{U}} \sum_{t=1}^{C} \Big(\sum_{i,j \in \mathcal{M}^t} \log(\sigma(i,j)) + \sum_{i^-,j^-} \log(1 - \sigma(i^-, j^-)) \Big) \\ - \lambda \sum_m \sum_{\tau \in \mathcal{T}} \frac{l_\tau^2 \|\mathbf{U}_m^\tau\|_2^2}{g_{m,\tau}} \\ \text{subject to } \sum_m \sum_{\tau \in \mathcal{T}} g_{m,\tau} = 1,\ g_{m,\tau} \geq 0, \end{aligned} \tag{3.10}$$

where additional variables $\{g_{m,\tau}\}$ are introduced. The way of setting l_τ is suggested in [108]. The problem above is then solved by alternatively optimizing $\mathbf{U}$ and $g_{m,\tau}$, as shown in Algorithm 2. In each iteration, we first hold coefficients $g_{m,\tau}$ as constant and update each column $\mathbf{U}_{:,t}$ using stochastic gradient descent. Then we fix $\mathbf{U}$ and update $g_{m,\tau}$, where the update equation is given in Line 6 of Algorithm 2.

3.4 Experiments

We evaluate the effectiveness of the proposed interpretation framework quantitatively on real-world datasets. A case study is also implemented to intuitively show the interpretation results.

3.4.1 Experimental Settings

Datasets. We use three real-world datasets to evaluate the interpretation results. The statistics of the datasets are in Table 3.1. Detailed descriptions of these datasets are as follows.

• **BlogCatalog**[2]: An online community network from which links between users and posts of

[2]http://people.tamu.edu/∼xhuang/Code.html

Dataset	N	M	$\|\mathcal{E}\|$	$\#class$
BlogCatalog	5,196	50	343,486	6
Flickr	7,575	60	479,476	9
20NewsGroups	18,774	60	401,108	$[6, 20]$

Table 3.1: Statistics of the datasets. Reprinted from [2].

users are extracted. Users are represented as nodes, and their associated posts are used as node attributes. Predefined class labels of users are also available.

- **Flickr**[2]: An image and video hosting website. The following relationships among users form a network, in which the tags of interest are used as node attributes. The groups that users joined are treated as class labels.

- **20NewsGroups**[3]: A collection of news documents, each of which belongs to one of the twenty different topics. The topics are used as class labels, and a topic hierarchy is directly available from the data source. We use a tf-idf vector to represent each document and measure the similarities among documents using cosine similarity. A network is constructed based on document similarities, and news texts are attached as attributes. The attributes are only used in interpretation, while for network embedding only link information is considered.

For all of the datasets above, we preprocess the attributes (i.e., word tokens) using LDA [110] to reduce the number of dimensions.

Alternative Methods for Global-View Interpretation. To evaluate whether the extracted hierarchical clusters by our approach are reasonable, we introduce two alternative clustering methods for comparative analysis. The goal is not to defeat these alternatives, since the proposed method can be seen as a variant of them.

- **SymNMF** [107]: A flat graph partitioning algorithm based on NMF. It has the same overall loss function as our method. The input is also the graph G constructed from embedding results.

[3]http://qwone.com/∼jason/20Newsgroups/

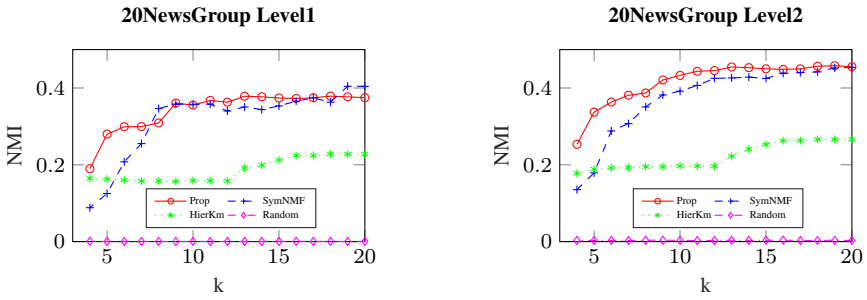


Figure 3.4: Hierarchical clustering performances for LINE on 20NewsGroup. Reprinted from [2].

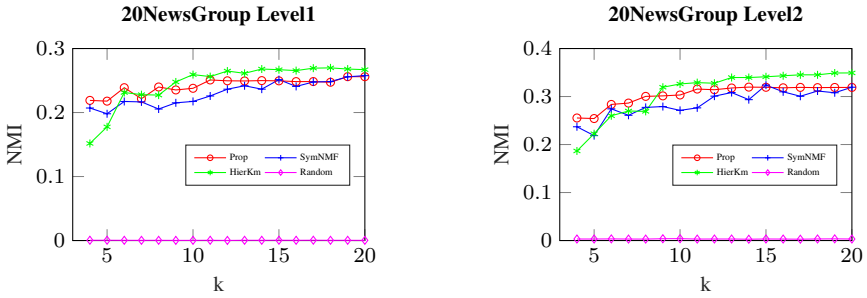


Figure 3.5: Hierarchical clustering performances for node2vec on 20NewsGroup. Reprinted from [2].

- **HierKm** [111]: A hierarchical clustering method based on the k-means algorithm. Its workflow is the same as Algorithm 1. It operates directly on embedding instances.

Baseline Methods. We further verify if the description of each cluster, selected from node attributes, correctly reflects its characteristics. The baseline methods are introduced as below.

- **MTGL** [108]: A multitask classification model also guided by tree-based group lasso. It shares the same global-view interpretation result as the proposed method.

- **NDFS** [112]: An unsupervised feature selection algorithm. We use the graph Laplacian built from embedding vectors and the attribute information of nodes as the input.

- **LIME** [11]: A model originally designed for interpreting local predictions in classifiers. In our experiments, we select a number of target embedding instances. For each instance, we select important attributes of its neighborhood compared with some other distant clusters.

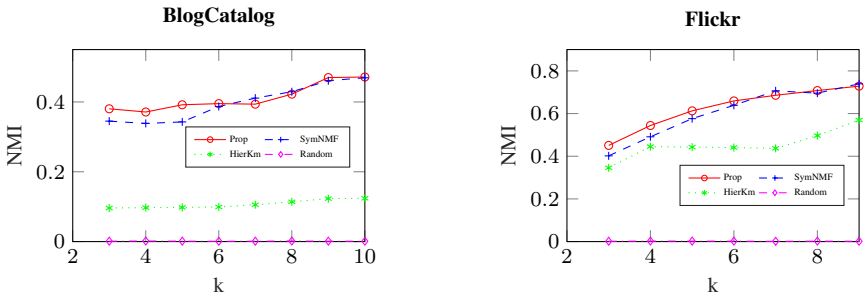


Figure 3.6: Hierarchical clustering performances for LANE on BlogCatalog and Flickr. Reprinted from [2].

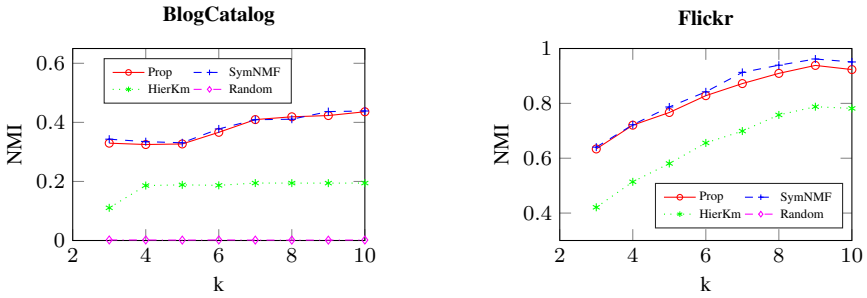


Figure 3.7: Hierarchical clustering performances for LCMF on BlogCatalog and Flickr. Reprinted from [2].

3.4.2 Evaluation on Global-View Interpretation

3.4.2.1 Evaluation Metric for Global-View Interpretation

The evaluation of global-view interpretation follows the common workflow of clustering evaluation. In particular, we use Normalized Mutual Information (NMI) [113] to measure the hierarchical clustering quality with respect to the ground-truth clusters. For datasets with defined hierarchy, we compute the NMI between the clustering results and the ground-truth clusters at each level.

3.4.2.2 Employed Network Embedding Methods

We introduce the employed network mebedding methods to be interpreted. Note that interpretation takes the network embedding results as input rather than the original network. We include the following embedding methods which: (1) preserve the first-order or high-order node proximity (e.g., LINE [16], SDNE [57], node2vec [3]); (2) incorporate labels with links and attributes (e.g., LCMF [114], LANE [58]). In particular, LINE, node2vec and SDNE are used to embed

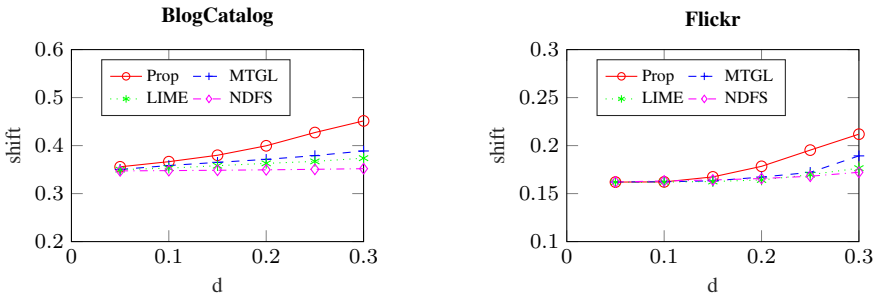


Figure 3.8: Local interpretation for LANE on BlogCatalog and Flickr. Reprinted from [2].

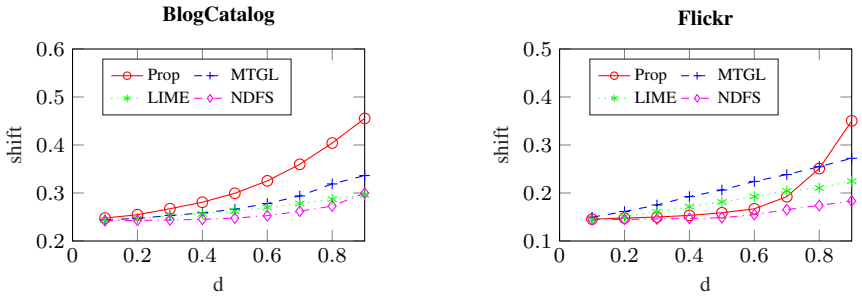


Figure 3.9: Local interpretation for LCMF on BlogCatalog and Flickr. Reprinted from [2].

the 20NewsGroup network, while LANE and LCMF are performed on BlogCatalog and Flickr networks. For LINE, we set the number of negative samples as 10, the number of samples as 200 millions, $\rho = 0.025$, and enable second-order proximity. For node2vec, we set $p = 1$, $q = 0.5$, walk length as 80 and the number of walk per node as 15. For SDNE, there are two encoding layers with 600 and 128 latent factors, respectively. For LANE, we set the balancing parameter of labels as 300 for BlogCatalog and 150 for Flickr. For LCMF, we set the balancing parameter of attributes as 3 for BlogCatalog data and 5 for Flickr data.

3.4.2.3 Global-View Interpretation Results and Analysis

The global-view interpretation results are shown in Figures 3.4∼3.7. We omit the results for SDNE on 20NewsGroup owing to lack of space, as they are similar to those for node2vec. Some observations are made as follows. First, in general, the proposed method and SymNMF have more stable clustering performances than HierKm, and even better performances in some cases. It means that, although we work on the embedding-based graph instead of directly on the embedding

instances, the clustering results are not significantly affected. In certain cases, we even benefit from clustering on embedding-based graphs. Second, the proposed method is comparable to SymNMF as k increases, as they actually share the same original loss function in Equation 3.3. The proposed method has better performance when k is small. This is probably because a small k is largely inconsistent with the real number of clusters in the datasets, which hinders the performance of flat clustering. In this case, hierarchically clustering can more effectively discover the internal structures of the data. In conclusion, the observations above validate the soundness of the developed global-view interpretation method.

3.4.3 Evaluation on Local-View Interpretation

The goal of local-view interpretation is to identify the attributes shared by the nodes in the same cluster obtained from the global-view interpretation. We design two sets of experiments to verify the correctness of the identified attributes, through adversarial perturbation and network reconstruction, respectively.

3.4.3.1 Adversarial Perturbation Based on Interpretation

As it is difficult to obtain the ground truth information (e.g., significant attributes), we first evaluate the accuracy of generated interpretation through *adversarial perturbation*. After identifying the significant attributes of nodes, we select a number of seed nodes for generating adversarial samples. We create a copy for each of the seeds, distort the value of their attributes and fed the new nodes along with the original network into the embedding algorithms. After that, we measure the relative *shift* of the new embeddings from the original embeddings of seeds. Let $\mathbf{z}_n$ and $\mathbf{z}'_n$ be the embedding instance of node n before and after perturbation, respectively, the shift is defined as:

$$shift(n, d) = 1 - \frac{|\text{Neighbors}(\mathbf{z}_n) \cap \text{Neighbors}(\mathbf{z}'_n)|}{|\text{Neighbors}(\mathbf{z}_n)|}, \tag{3.11}$$

where $\text{Neighbors}(\mathbf{z}_n)$ denotes the set of neighbor nodes around n in the embedding space, and d is the amplitude of adversarial perturbation. Large shift is expected if the generated interpretation faithfully reflects the mechanism of employed embedding models.

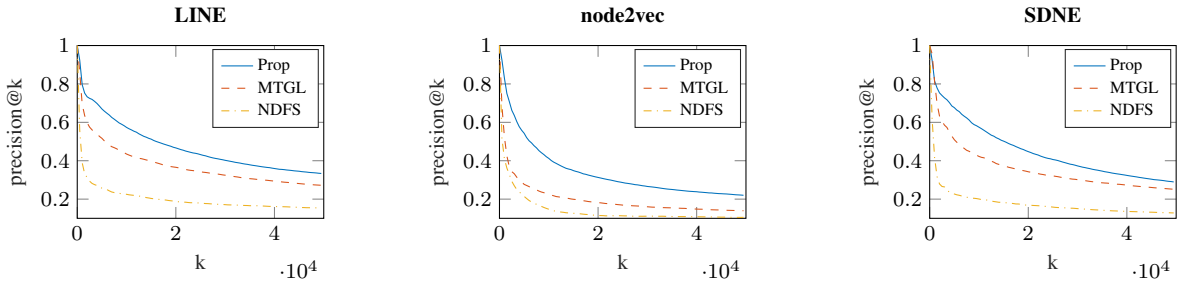
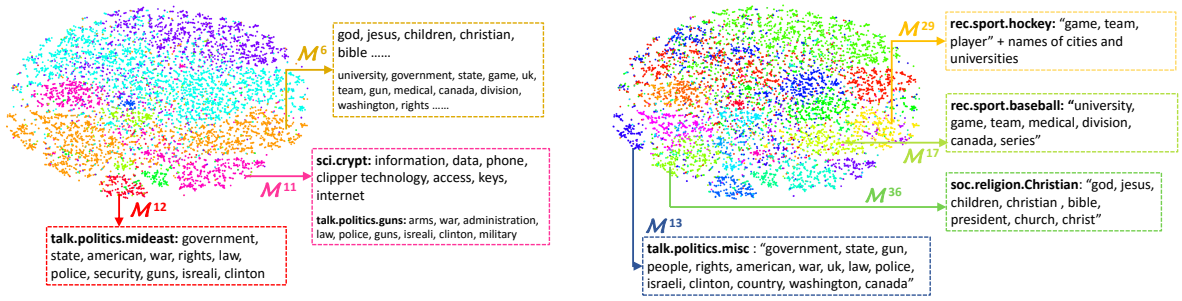


Figure 3.10: Network reconstruction on the 20NewsGroup dataset. Reprinted from [2].

Specifically, the number of seeds is set as $0.03 \times N$ for each network, where N is the total number of nodes. For each seed's embedding vector, the number of neighbors is chosen as $0.01 \times N$ for LANE and $0.03 \times N$ for LCMF. The parameters settings of LANE and LCMF remain the same as in the previous experiments. For LCMF, since it outputs several matrices besides the embedding matrix $\mathbf{Z}$, we fix the values of these matrices and only allow $\mathbf{Z}$ to be updated during learning process. For each seed node in cluster $\mathcal{M}^t$, we identify 4 positively significant attributes and 4 negatively significant ones to be perturbed, corresponding to the largest and smallest entries in $\mathbf{U}_{:,t}$, respectively. The value of the former ones are decreased, while the value of latter ones are increased. The L_2 norms of both types of perturbation are normalized to d.

The experiments are conducted on BlogCatalog and Flickr. The performance of adversarial perturbation is shown in Figure 3.8 for LANE and Figure 3.9 for LCMF. The figures show that the average shift of adversarial samples increases as we increase the perturbation amplitude. The adversarial samples created by the proposed method are more effective, as they are more dramatically shifted from their original neighborhood prior to adversarial perturbation. The proposed method has better performance than MTGL. The reason could be that the edge weights contained in $\mathbf{A}$ are more informative than binary class labels used in MTGL. In addition, interpretation approaches (e.g., Prop, MGTL), which focus on overall embedding instances, have better performances than LIME which interprets individual embeddings locally. It indicates the importance of considering a wider range of contexts in solving our problem. The embedding instances in other clusters provide contrastive information to filter out irrelevant attributes with respect to the current cluster, while



(a) (b)

Figure 3.11: (a): The inducted taxonomy with 7 leaf clusters. (b): The inducted taxonomy with 20 leaf clusters. Reprinted from [2].

instances within the same cluster help filter out noises in attribute selection.

3.4.3.2 Network Reconstruction from Interpretation

In this subsection, we evaluate interpretation results with respect to their capability of network reconstruction on the 20NewsGroup dataset. The motivation is that, while network embedding methods encode topological structures into embeddings, interpretation algorithms try to recover the attribute patterns consistent with the embedding distribution. We build an LDA model with 60 latent topics from the documents, and assign each node with a 60-dimensional attribute vector. After solving $\mathbf{U}$, in each leaf cluster $\mathcal{M}^t$, we predict links based on the similarities between nodes with respect to attributes, the importance of which is weighted by $\mathbf{U}_{:,t}$. The predicted links correspond to node pairs with large similarity scores. The existing links in the original network serve as the ground-truth. We use $precision@k$ as the evaluation metric.

The results are presented in Figure 3.10. In general, the proposed method achieves better performance than baseline methods. Its interpretation performances are relatively stable across different network embedding methods. The edge weights in $\mathbf{A}$ contain more information than class labels, which could explain why the proposed method performs better than MTGL. The advantages of NDFS are not reflected via this experiment. The reason is that the attributes information are already modeled as links and fed into network embedding methods, so jointly considering embeddings and attributes does not bring additional benefits.

3.4.4 Case Study

We conduct a case study on 20NewsGroups network using LINE [16] to show the taxonomy induction result and the characteristics of extracted taxonomy concept (i.e., clusters). We present the inducted taxonomy with 7 leaf clusters and 20 leaf clusters in Figure 3.11a and Figure 3.11b, denoted by $\mathcal{T}^7$ and $\mathcal{T}^{20}$, respectively. The two figures are generated from the same taxonomy, but with different depths of division. Some clusters in $\mathcal{T}^{20}$ are obtained by splitting the larger clusters in $\mathcal{T}^7$. Clusters are indicated with different colors. The visualization of embedding vectors are obtained using t-SNE [115]. We select several clusters and present the keywords of the documents in each cluster in the dashed boxes. For each cluster $\mathcal{M}^t$, we first find its significant attributes m corresponding to the large positive entries $\mathbf{U}_{m,t}$, and then identify the keywords associated with each attribute m. Here the "attributes" are the latent factors extracted by LDA. Keywords of larger font size in boxes are ranked higher than those of smaller font size. Keywords that belong to the stop words or email artifacts (e.g., com, edu) are not shown. The bold text prior to the keywords, if available, represent the manually-identified news topic (named by the dataset provider) which the majority of the nodes in the cluster belong to. There are 20 such topics in the dataset.

We make the following observations from Figure 3.11a and 3.11b. First, nodes in each cluster extracted by our interpretation method locate closely to each other. In general, the cluster structures are consistent with the visualization results, which reflects the effectiveness of the global-view interpretation method. Second, the keywords of each cluster are consistent with the ground truth news topics, which validates the effectiveness of the local-view interpretation. Third, we can observe that some major clusters, such as $\mathcal{M}^6$ and $\mathcal{M}^{11}$, contain multiple topics, since they have not been throughly decomposed in shallow levels of the taxonomy. Different topics are separated as we continue splitting clusters towards deeper levels. For example, many topics are mixed in $\mathcal{M}^6$ which is one of the major clusters in $\mathcal{T}^7$. However, as we split $\mathcal{M}^6$ into $\mathcal{M}^{13}$, $\mathcal{M}^{17}$, $\mathcal{M}^{29}$ and $\mathcal{M}^{36}$, each sub-cluster has a coherent topic. Some keywords in these sub-clusters are inherited from $\mathcal{M}^6$, such as the {god, jesus} in $\mathcal{M}^{36}$, {game, team} in $\mathcal{M}^{29}$ and {government, gun} in $\mathcal{M}^{13}$, so that a larger and coarser concept is split into smaller but refined ones.

4. INTERPRETABLE MODELING VIA RESOLVING LATENT REPRESENTATIONS [1]

The local feature-based interpretation and representation interpretation are post-hoc methods that can only be applied to understand trained models. While these techniques have a broad range of applications, there are also several challenges that prevent us from achieving further model transparency. First, it is hard to guarantee the accuracy of post-hoc interpretation with respect to the actual working mechanism of the model. Second, post-hoc interpretation does not help gain control of the model during its training stage. We have to use other techniques to adjust the behavior of models if we find undesirable properties after applying post-hoc interpretation.

The above challenges can be tackled through designing models that are inherently interpretable. In this work, instead of trying to understand latent representations after building the model, we directly control the semantic meaning of latent dimensions during model establishment. Specifically, we choose network embedding as the domain scenario. We extend several existing network embedding models, including Deepwalk, PTE and GCN. In the proposed model, each node may have several embedding vectors, each of which corresponds to one facet/meaning of the dataset. We call this setting as *node polysemy*, as motivated by word polysemy in natural language processing, where each word owns several meanings. Besides obtaining the meanings of latent dimensions, the proposed new models could also achieve better performances in downstream tasks.

4.1 Polysemous Network Embedding

In this section, we introduce how to explicitly assign meanings to latent dimensions in network embedding, by using Deepwalk as the base model. Then, we design an optimization algorithm to train the polysemous embedding model. We will also discuss how to estimate the facet of a node in different contexts. Finally, we introduce how to combine embedding vectors of different facets towards downstream tasks such as classification and link prediction.

[1] Part of this chapter is reprinted with permission from "Is a single vector enough? exploring node polysemy for network embedding" by Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Pages 932–940, Copyright 2019 by Association for Computing Machinery.

4.1.1 Polysemous Deepwalk

In the setting of polysemous network embedding, for the Deepwalk model, each node v_i is associated with a target embedding matrix $\mathbf{U}_i \in \mathbb{R}^{K_i \times D}$ and a context embedding matrix $\mathbf{H}_i \in \mathbb{R}^{K_i \times D}$. Here K_i is the number of embedding vectors possessed by node v_i considering its different possible facets. The traditional Deepwalk model could be seen as a special case where $K_i = 1$. The embedding vector for facet k of node v_i is denoted as $\mathbf{U}_i^k \in \mathbb{R}^D$ or $\mathbf{H}_i^k \in \mathbb{R}^D$, and D is the dimension of each embedding vector. Different nodes may be associated with different number of embedding vectors, depending on the diversity of their characteristics in the network. In this work, for illustration purposes, we simply let all nodes have the same number of embedding vectors, so that $K_i = K$ and K is a predefined constant integer. In practice, the value of K could be estimated from data. For examples, K can be approximately set as the number of latent interest categories in recommender systems, or be estimated as the number of major topics in an academic network. We will discuss in later sections about how to assign facets to nodes with different probabilities.

Traditional Deepwalk: Deepwalk utilizes the skip-gram model [116] that is trained using the maximum likelihood estimation, where we try to find the model parameters that maximize the likelihood of obtained *observations*. Specifically, let θ be the parameters to be optimized and $\mathcal{O}$ be the set of all observations, the objective function to be maximized is:

$$\begin{aligned}
\mathcal{L}_{DW}(\theta) &= \sum_{o \in \mathcal{O}} \log\ p(o|\theta) = \sum_{o \in \mathcal{O}} \log\ p\big((\mathcal{N}(v_i), v_i)|\theta\big) \\
&= \sum_{o \in \mathcal{O}} \sum_{v_j \in \mathcal{N}(v_i)} \log\ p(v_j|v_i) \\
&= \sum_{o \in \mathcal{O}} \sum_{v_j \in \mathcal{N}(v_i)} \log\ \frac{\exp(< \mathbf{H}_j, \mathbf{U}_i >)}{\sum_v \exp(< \mathbf{H}_v, \mathbf{U}_i >)},
\end{aligned} \tag{4.1}$$

where each observation $o \in \mathcal{O}$ is defined as a tuple, $o = (\mathcal{N}(v_i), v_i)$, consisting of a central node v_i and its context $\mathcal{N}(v_i)$. Each node within the context is denoted as v_j so that $v_j \in \mathcal{N}(v_i)$. The model parameters, i.e., the embedding vectors of nodes, are used in computing $p(v_j|v_i)$ which is the conditional probability of having v_j as one of the contexts given v_i.

Consideration of Multiple Facets: In our settings where each node owns multiple facets, the activated facet of a node varies under different contexts. Also, the facet of a given context is determined by the combination of all facets of the nodes within the context. Suppose the distribution of node facets are known in advance, and we treat them as prior knowledge denoted as $\mathcal{P}$. Taking the additional information into account, the objective is reformulated as:

$$\mathcal{L}_{PolyDW}(\theta) = \sum_{o \in \mathcal{O}} \log\ p(o|\mathcal{P}, \theta) = \sum_{o \in \mathcal{O}} \log\ [\sum_{s(o)} p(o|s(o), \mathcal{P}, \theta) \cdot p(s(o)|\mathcal{P}, \theta)]. \tag{4.2}$$

Here $s(o)$ is defined as one case of activated facets of all nodes within o, so $s(o) = \{s(v|o)\ |v \in v_i \cup \mathcal{N}(v_i)\}$ where $s(v|o)$ is the activated facet of node v in the context of o. In a given observation o, suppose the activated facet of v_i is k_i and the facet of each $v_j \in \mathcal{N}(v_i)$ is k_j, the conditional probability $p(o|s(o), \mathcal{P}, \theta)$ is thus defined as:

$$p(o|s(o), \mathcal{P}, \theta) = \prod_{v_j \in \mathcal{N}(v_i)} p(v_j|v_i, s(o)), \tag{4.3}$$

and each product factor is calculated as

$$p(v_j|v_i, s(o)) = \frac{\exp(< \mathbf{H}_j^{k_j}, \mathbf{U}_i^{k_i} >)}{\sum_{v,k} \exp(< \mathbf{H}_v^{k}, \mathbf{U}_i^{k_i} >)}, \tag{4.4}$$

which is similar to the softmax function in traditional skip-gram models, except that general node embeddings are replaced by node facet embeddings. Here "$< \cdot, \cdot >$" denotes the inner product of two vectors. The denominator acts as normalization over all possible nodes and facets. For readability, we omit $\mathcal{P}$ after the expansion in Equation 4.3, as it is no longer applied in later steps.

Efficient Optimization: It is cumbersome to directly apply gradient descent to optimize the objective function in Equation 4.2 $\sim$ Equation 4.4, due to the summation term inside the logarithm function. Also, it becomes unclear of how to incorporate negative sampling [116] to approximate the normalization term for more efficient computation. Therefore, we further derive the objective

Algorithm 3: Polysemous Deepwalk

Input: Input network $\mathcal{G}$, number of nodes N, facet sampling rate R.
Output: Embedding matrix $\mathbf{U}$, context embedding matrix $\mathbf{H}$.

```
Initialize U and H ;
Estimate facet distribution p(v_i) for each node, so that P = {p(v_i)}, 1 ≤ i ≤ N ;
Obtain observations O via random walks and context window sliding ;
while not converged do
    for o ∈ O do
        Obtain a target-context tuple o = (N(v_i), v_i) ;
        for 1 ≤ r ≤ R do
            Sample a facet k_i = s(v_i|o) ~ p(v_i|o), defined in Eq. 4.7 ;
            Sample a facet k_j = s(v_j|o) ~ p(v_j|o), defined in Eq. 4.7, for each context node
              v_j ∈ N(v_i) ;
            Obtain negative samples ;
            Update U_i^{k_i}, H_j^{k_j} for v_j ∈ N(v_i), as well as facet embeddings of negative samples,
              using stochastic gradient descent ;
```

function as below:

$$\begin{aligned}
\mathcal{L}_{PolyDW}(\theta) &= \sum_{o\in\mathcal{O}} \log\,[\sum_{s(o)} p(o|s(o),\mathcal{P},\theta)\cdot p(s(o)|\mathcal{P},\theta)] \\
&\geq \sum_{o\in\mathcal{O}}\sum_{s(o)} p(s(o)|\mathcal{P},\theta)\cdot \log\ p(o|s(o),\mathcal{P},\theta) \\
&= \sum_{o\in\mathcal{O}}\sum_{s(o)} p(s(o)|\mathcal{P})\cdot [\sum_{v_j\in\mathcal{N}(v_i)} \log\ p(v_j|v_i,s(o))] \\
&= \mathcal{L}^*_{PolyDW}(\theta)
\end{aligned} \tag{4.5}$$

The intuition behind the above transformation is that, instead of maximizing the original objective function, we turn to maximize its lower bound [117], denoted as $\mathcal{L}^*_{PolyDW}(\theta)$, by applying Jensen's inequality. The final form of the objective function is similar to that of the skip-gram model, except the external summation over $s(o)$. As a result, we could adopt the same negative sampling strategy as in the traditional skip-gram model to approximate the normalization term in $p(v_j|v_i,s(o))$. Therefore, one major advantage of the proposed polysemous embedding model is that the training process can be easily implemented through minimal modification to the existing learning frameworks, by adding an additional sampling step of assigning activated facets to nodes in each observation o.

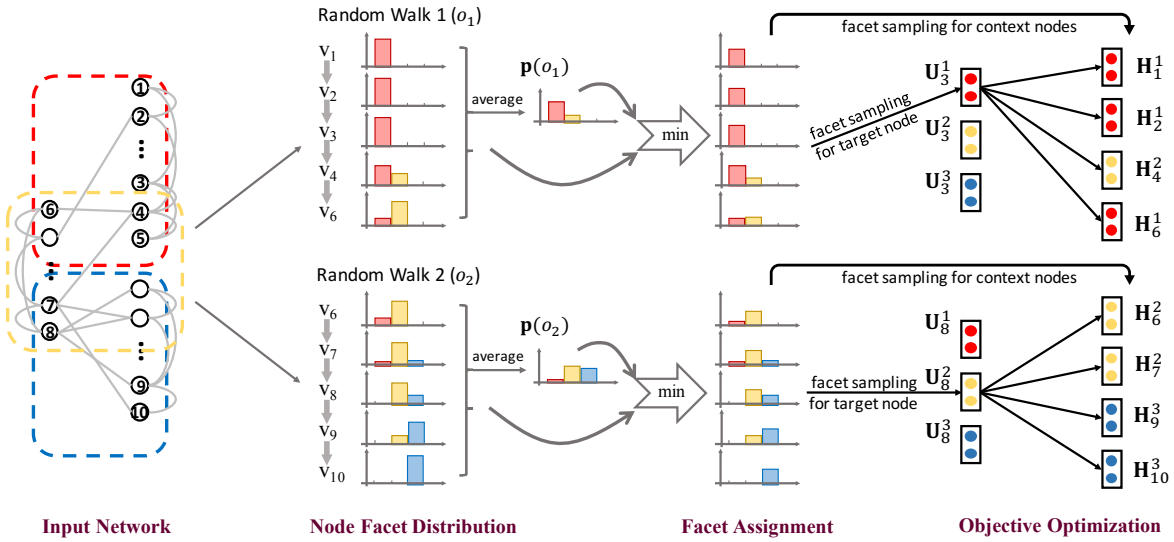


Figure 4.1: The overall training procedure of Polysemous Deepwalk. Each color refers to one facet (i.e., cluster in this case) in the input network. Each histogram visualizes a probability distribution. Note that the above network is only a toy example for illustration purposes, where numerical values do not reflect real results. Reprinted from [4].

Specifically, the summation over $\sum_{s(o)}$, following the distribution of $p(s(o)|\mathcal{P})$, is implemented through *facet sampling* separately for each node in o. The overall optimization algorithm is specified in Algorithm 3. After initialization and node-facet distribution estimation (will be introduced later), a number of random walks are sampled just as in traditional Deepwalk (Line 3). Then, for each observation o, several rounds of facet sampling are conducted on each node within o (the loop in Line 7). In each round, each node has one facet activated (Line $8 \sim 10$), so that the embedding vector corresponding to that facet will be updated using SGD (Line 11). The major additional computation cost, compared with traditional Deepwalk, comes from the increased size of training data in $\mathcal{O}$ by a factor of sampling rate R.

4.1.2 Node Facets Estimation and Assignment

The overall process of polysemous Deepwalk is illustrated in Figure 4.1. The "Objective Optimization" part has been introduced as above, while the other steps related with facet distribution and facet assignment will be discussed in this subsection.

Following the process in the figure, we first introduce how the prior knowledge $\mathcal{P}$ can be obtained, and how to determine the facet of nodes given a specific observation o. For now, we limit the discussion to undirected homogeneous plain networks, and provide one method to obtain the global distribution of node facets by only leveraging the network adjacency matrix. For networks with attribute information or heterogeneous information networks, we may resort to other strategies and we leave it to future work. Let $\mathbf{A}$ be the symmetric adjacency matrix of the network, we perform community discovery on the network [118, 119]:

$$\min_{\mathbf{P}\geq 0} \|\mathbf{A} - \mathbf{P}\cdot\mathbf{P}^T\|_F^2 + \alpha\|\mathbf{P}\|_F^2, \tag{4.6}$$

where $\mathbf{P}$ can be solved using gradient search algorithms. The probability $p(k|v)$, that node v is assigned with the k-th facet, can be calculated as $p(k|v) = \frac{\mathbf{P}_{v,k}}{\sum_{c=1,...,K}\mathbf{P}_{v,c}}$. We define the facet distribution of a node as $\mathbf{p}(v) = [p(1|v), ..., p(K|v)]$. We treat $\mathbf{p}(v_i), 1 \leq i \leq N$ as the prior knowledge $\mathcal{P}$, as it encodes the global understanding of the network states. Applying attribute information could achieve better estimation, but it is beyond the discussion in this work.

After obtaining the prior knowledge $\mathcal{P}$, we are able to estimate the overall facet of each observation o as well as the facet of nodes within o. Given an observation $o = (\mathcal{N}(v_i), v_i)$, a straightforward way to define its facet distribution $\mathbf{p}(o)$ is by averaging the facet distributions of nodes inside the observation, i.e., $\mathbf{p}(o) = \big(\mathbf{p}(v_i) + \sum_{v_j\in\mathcal{N}(v_i)}\mathbf{p}(v_j)\big)/(|\mathcal{N}(v_i)|+1)$. Considering that the activated facet of a node depends on the specific context where it is observed, given o, the node's facet $s(v|o)$ is sampled according to the distribution $\mathbf{p}(v|o)$, which is defined heuristically as below

$$\mathbf{p}(v|o) = \min(\mathbf{p}(v), \mathbf{p}(o)), \tag{4.7}$$

where we include a $\min(\cdot,\cdot)$ operator because it is undesirable to assign a node v_i with facet k if $p(k|v_i) \approx 0$, even when the k-th entry in $\mathbf{p}(o)$ is large. To make it a valid probability distribution, $\mathbf{p}(v|o)$ is further normalized to sum to 1.

Till now, we have introduced the whole training process of polysemous Deepwalk as shown in

Figure 4.1. Given the input network, we first estimate node facet distributions as prior knowledge $\mathcal{P}$. Then, random walks are performed to construct node-context observations $\mathcal{O}$. After that, within each walk sample o, each node is assigned with an activated facet. Finally, node embeddings of correspondent facets are updated through optimization.

4.1.3 Aggregation of Multiple Embeddings for Inference

We are able to obtain multiple embedding vectors for each node after training the polysemous model. Then the question arises that, during inference, how to collectively consider different embedding vectors for subsequent tasks. Here we discuss two major network analysis tasks including classification and link prediction.

Node Classification: In node classification, for each node, our strategy is to combine multiple vectors $\{\mathbf{U}_i^k\}_{k=1,...,K}$ into a joint vector $\tilde{\mathbf{U}}_i$. Specifically, some options include: $\tilde{\mathbf{U}}_i = \mathbf{U}_i^1 \oplus \mathbf{U}_i^2 \oplus ... \oplus \mathbf{U}_i^K$, where we directly concatenate embeddings of different facets, or

$$\tilde{\mathbf{U}}_i = \big(p(1|v_i) \cdot \mathbf{U}_i^1\big) \oplus \big(p(2|v_i) \cdot \mathbf{U}_i^2\big) \oplus ... \oplus \big(p(K|v_i) \cdot \mathbf{U}_i^K\big), \tag{4.8}$$

where we first scale each embedding vector with the probability of belonging to the corresponding facet and then concatenate these scaled vectors. Here $\oplus$ denotes the concatenation operation. The resultant embedding vectors $\{\tilde{\mathbf{U}}_i\}_{i=1,...,N}$ can be directly used in node classification. We adopt Equation 4.8 in experiments.

Link Prediction: For link prediction or network reconstruction tasks, a higher similarity score between the representations of two nodes indicates greater possibility that a link exist between the nodes. We can define the similarity between two nodes v_i and v_j as:

$$similarity(v_i, v_j) = \sum_{k=1}^{K} \sum_{k'=1}^{K} p(k|v_i)\, p(k'|v_j) < \mathbf{U}_i^k, \mathbf{U}_j^{k'} >, \tag{4.9}$$

where different facet pairs of embedding vectors contribute to the overall similarity computation, weighted by the probability that the node belongs to the corresponding facet.

4.1.4 Discussion

The proposed work may be related to disentangled representation learning [120], since different representation dimensions are sensitive to varying factors behind the data. Traditional disentangled representation learning methods in computer vision are based on variational autoencoders and encourage different latent dimensions to be mutually independent. Different from the traditional methods where each single dimension encodes one facet in data, our method is more flexible where dimensions are grouped and several dimensions could encode one facet together. Moreover, it helps improving the interpretability of representation learning [121], because representation dimensions are separated according to node facets that could be associated with concrete meanings or characteristics of real-world networked objects.

4.2 Models Extended by Polysemous Embedding

The methodology of polysemous embedding can also be applied to extend other fundamental single-embedding models. In this section, we elaborate two scenarios as examples. First, we show how polysemous embedding can be used in heterogeneous networks where links exist between nodes of different types. Second, we show how polysemous embedding can be combined with graph neural networks, where the embedding look-up tables in Deepwalk is replaced by feedforward modules.

4.2.1 Polysemous PTE for Heterogeneous Networks

We show how to consider node polysemy when modeling heterogeneous networks. We choose PTE [62] as the base model to be extended, as it is a fundamental model for tackling the problem. To simplify the illustration, we only consider bipartite networks during model development. The discussion can be applied for more complex networks, since the objective function of PTE could be extended as the sum of several terms considering multiple bipartite or homogeneous networks.

Given a network containing two types of nodes $\mathcal{V}_A$ and $\mathcal{V}_B$, as well as a set of links denoted as $\mathcal{E}$, each observation o is defined as a link $o = (u_j, v_i) \in \mathcal{E}$ where $v_i \in \mathcal{V}_A, u_j \in \mathcal{V}_B$. The set of all observations is thus defined as $\mathcal{O} = \mathcal{E}$. Similar to the derivation in Section 4.1.1, the objective

function is formulated as:

$$\begin{aligned}\mathcal{L}_{polyPTE}(\theta) &= \sum_{o\in\mathcal{E}} \log\,[\sum_{s(o)} p(o|s(o),\mathcal{P},\theta)) \cdot p(s(o)|\mathcal{P},\theta)] \\ &\geq \sum_{o\in\mathcal{E}} \sum_{s(o)} p(s(o)|\mathcal{P},\theta) \cdot \log\ p(o|s(o),\mathcal{P},\theta) \\ &= \sum_{o\in\mathcal{E}} \sum_{s(o)} p(s(o)|\mathcal{P}) \cdot \log\ p(u_j|v_i,s(o)),\end{aligned} \tag{4.10}$$

where

$$p(u_j|v_i,s(o)) = \frac{\exp(<\mathbf{H}_j^{k_j},\mathbf{U}_i^{k_i}>)}{\sum_{v,k}\exp(<\mathbf{H}_v^{k},\mathbf{U}_i^{k_i}>)}. \tag{4.11}$$

According to the original PTE model, the matrix $\mathbf{U}$ contains the embedding vectors of nodes in $\mathcal{V}_A$, and $\mathbf{H}$ contains the embedding vectors of nodes in $\mathcal{V}_B$. The resultant lower bound objective is denoted as $\mathcal{L}^*_{polyPTE}$, which is to be optimized.

4.2.1.1 Node Facet Assignment

The strategy of determining heterogeneous node facet distribution is similar to that of the previous section. Given the asymmetric adjacency matrix $\mathbf{A}$, we first solve

$$\min_{\mathbf{P}\geq 0,\mathbf{Q}\geq 0} \|\mathbf{A}-\mathbf{P}\cdot\mathbf{Q}^T\|_F^2 + \alpha(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2), \tag{4.12}$$

where we set $\alpha = 0.05$ in experiments. The resultant factor matrix $\mathbf{P}$ contains the association intensity between $\mathcal{V}_A$ and facets, while $\mathbf{Q}$ contains the facet association information for nodes in $\mathcal{V}_B$. Then, we normalize $\mathbf{P}$ and $\mathbf{Q}$ to obtain the probabilities of belonging to different facets for each node $v_i \in \mathcal{V}_A$ and $u_j \in \mathcal{V}_B$ [119]. If we denote the facet distribution as $\mathbf{p}(v_i)$ and $\mathbf{p}(u_j)$ respectively for the two types of nodes, then the facet distribution of an observation (i.e., an edge) is computed as $\mathbf{p}(o) = \big(\mathbf{p}(v_i) + \mathbf{p}(u_j)\big)/2$. Different from polysemous Deepwalk, here we simply let $\mathbf{p}(v|o) = \mathbf{p}(o)$ be applied for node-facet sampling, because the window size in PTE equals to 1 which is much smaller than Deepwalk.

4.2.1.2 *Engage Multiple Embeddings for Inference*

The way of jointly considering a node's multiple embedding vectors, for downstream tasks, is similar to what is introduced in Section 4.1.3. The only difference is that the measurement of similarity between two different types of nodes is adjusted as:

$$similarity(v_i, u_j) = \sum_{k=1}^{K} \sum_{k'=1}^{K} p(k|v_i)\, p(k'|u_j) < \mathbf{U}_i^k, \mathbf{H}_j^{k'} > . \tag{4.13}$$

4.2.2 Polysemous Embedding with GCN

Instead of using indexing matrices to store embeddings, polysemous embedding can be applied to models of other architectures. In this subsection, we consider GCN [66] as an example and show how it could be extended as PolyGCN to consider node polysemy. Different from PolyDeepwalk and PolyPTE that keep embedding look-up tables to store embedding vectors, PolyGCN uses a feedforward network module to generate embedding of each node by gathering information from neighborhoods.

The fundamental step of embedding generation in GCN is to aggregate information from a node's local neighborhood. Let $\mathbf{u}_d(i)$ denote the intermediate representation of node v_i on layer of depth d, then the forward propagation is

$$\mathbf{u}_d(i) \leftarrow \sigma\Big(\mathbf{W}_d \cdot \text{MEAN}(\mathbf{u}_{d-1}(i) \cup \{\mathbf{u}_{d-1}(j), \forall v_j \in \mathcal{N}(v_i)\})\Big), \tag{4.14}$$

where $\mathbf{W}_d$ is the weight matrix on layer d, MEAN refers to the aggregator based on the pre-processed adjacency matrix, and $\mathcal{N}(v_i)$ is the local neighborhood of v_i. The final embedding output for v_i is defined as $\mathbf{U}_i = \mathbf{u}_{d_{max}}(i)$, where d_{max} denotes the depth of the final layer. After forward propagation, GCN can be trained in an unsupervised manner similar to Deepwalk or PTE.

We here propose a feasible approach to incorporate node polysemy into GCN-based models, where each facet of embeddings corresponds to one GCN model, while the internal architecture of each GCN is unchanged. Specifically, within a certain facet k, we limit the local neighborhood of

	$\lvert\mathcal{V}_A\rvert$	$\lvert\mathcal{V}_B\rvert$	$\lvert\mathcal{E}\rvert$
BlogCatalog	5,196	–	171,743
Flickr	7,575	–	239,738
MovieLens	6,040	3,952	1,000,209
Pinterest	55,187	9,916	1,500,809

Table 4.1: Statistics of datasets. Reprinted from [4].

the node as other nodes with the same facet k activated, so that

$$\mathbf{u}_d^k(i) \leftarrow \sigma\left(\mathbf{W}_d^k \cdot \text{MEAN}(\mathbf{u}_{d-1}^k(i) \cup \{\mathbf{u}_{d-1}^k(j), \forall v_j \in \mathcal{N}^k(v_i)\})\right), \tag{4.15}$$

where $\mathcal{N}^k(v_i)$ denotes the local neighborhood under facet k. The final embedding output for facet k of v_i is $\mathbf{U}_i^k = \mathbf{u}_{d_{max}}^k(i)$. The main question to be answered is how to construct the local neighborhood for a specific facet k of node v. Here we still consider bipartite networks as the scenario, so for each facet there are two GCNs respectively for generating embeddings for each type of nodes. Following the similar strategy as in Section 4.2.1, the observed links are decomposed into interaction results of different facet pairs. We further relax the problem so that embeddings of different facets are mutually independent. Specifically, the original observation matrix $\mathbf{A} = \sum_k \mathbf{A}^k$, where $\mathbf{A}^k \geq 0$ and $\mathbf{A}^k(i,j)$ is proportional to $\mathbf{P}(i,k) \cdot \mathbf{Q}(j,k)$. $v_j \in \mathcal{N}^k(v_i)$ if both v_i and v_j connect to the same node of the other type under facet k. For training each GCN pair, we adopt the similar unsupervised training strategies as proposed in [65], where nodes are encouraged to have similar embeddings of facet k if they are connected according to the corresponding observation matrix.

4.3 Experiments

Experiments are conducted on several real-world datasets. We try to answer several questions through experiments. First, how effective is the proposed polysemous embedding method compared with single-vector embedding counterparts? Second, is considering node polysemy beneficial for network embedding when evaluated on different downstream tasks? Third, how will polysemous embedding models react to the changes of hyperparameters?

4.3.1 Experimental Settings

We first introduce the applied datasets as below. The detailed information is shown in Table 4.1.

- **BlogCatalog**: A social network dataset built based on connections between online bloggers. The original dataset contains both link and attribute information, while we only keep the links in our experiments. Each node is also associated with a label determined from some predefined interest categories of bloggers.

- **Flickr**: A network dataset constructed from interactions between users on a multimedia hosting website. Each link corresponds to a following relation between users. The groups that users joined are regarded as labels.

- **MovieLens**: A movie rating dataset widely used for evaluating collaborative filtering models. In our experiments, we treat it as a heterogeneous network where links exist between users and items (i.e., movies). We transform rating scores into implicit data, so that each entry is either 0 or 1 indicating whether the user rated the movie [122]. In this way, conducting recommendation on this dataset can also be regarded as performing link prediction.

- **Pinterest**: An implicit feedback data originally constructed for image recommendation. Users and images are regarded as nodes. Each link is a pin on an image initiated by a user. Each user has at least 20 links. Similarly, link prediction can be seen as recommending images to users.

The networks in BlogCatalog and Flickr are homogeneous, and we use them in classification and link prediction. The networks in MovieLens and Pinterest are heterogeneous, and we will apply them in link prediction tasks which can also be seen as doing recommendation. The baseline methods for comparison are as below.

- **Deepwalk** (**DW**) [17], **PTE** [62], **GCN** [67]: Some commonly used network embedding models that map each node to a single vector. We include them as baseline methods to analyze whether their polysemous embedding counterparts achieve better performance. Here GCN is only applied in heterogeneous link prediction.

- **NMF** [118, 123]: A traditional model that learns latent factors from data. We include NMF as one of the baseline methods because we applied it to estimate the global facets contained in data. The number of latent factors is the same as the number of facets in the polysemous model.

- **MSSG** [124]: A multi-sense word embedding model original developed for natural language processing. Senses of words determined by clustering their average context representations. Each word has only one embedding vector when it is regarded as context. The model is adapted for network analysis tasks.

- **MNE** [125]: A model that factorizes the network proximity matrix into several group of embedding matrices, and adds a diversity constraint to force different matrices focus on different aspects of nodes.

4.3.2 Node Classification

Node classification is a commonly applied task for evaluating network embedding outcomes. In this experiment, we compare the performance of the proposed polysemous models compared with baseline models. We use Deepwalk as the base single-embedding model. The proposed model is named as PolyDeepwalk. The BlogCatalog and Flickr datasets are applied in this experiment.

Unless specifically stated in each task, the default model hyperparameters are set as follows. For BlogCatalog dataset, the number of walks generated per node is 110, the length of walk is 11, the context size is 8, the number of facets is 6 considered for PolyDeepwalk, the embedding of dimension is 210 for each node in Deepwalk and $210/6 = 35$ for each node's facet in PolyDeepwalk, the number of negative samples is 5 for Deepwalk and 10 for PolyDeepwalk. For Flickr dataset, the parameter settings are similar, except that the number of facets is 5 for PolyDeepwalk, the embedding dimension is 180 for each node in Deepwalk and $180/5 = 36$ for each node's facet in PolyDeepwalk, and the number of negative samples is 15 for PolyDeepwalk. It is worth noting that the dimension of polysemous embedding is divided by the number of facets, so that after concatenation, the resultant embedding has the same length as the single-embedding counterpart. We use Support Vector Machine as the classification model, where 80% of data samples are used as

training data. The performance comparison is illustrated in Figure 4.2 and Figure 4.3. Note that we do not plot the performance of NMF as its F1 scores are much lower than other models. Some work includes node attributes as additional information source when using NMF related models for node classification [58, 126], but this is beyond our experiment setting. From the figures, some observations are summarized as below:

- In general, PolyDeepwalk achieves better performances than other polysemous embedding models. Also, by varying the dimensionality, we can observe that polysemous embedding models are less likely to be affected, while the classification performance of Deepwalk gradually decreases as dimensionality increases.

- Deepwalk and PolyDeepwalk have different responses to the change of context size. The former achieves better performances as context size increases, while the latter shows the opposite trend. A possible reason is that, as the context window enlarges in PolyDeepwalk, random walks from each node are more likely to reach other network regions of different facets. As a result, the facet distributions are over smoothed, so it becomes more difficult to decide which facet a context window belongs to. It could be one of the challenges to be tackled in future work.

Besides making comparisons to baseline models, we also analyze the sensitivity of PolyDeepwalk to some of the key parameters when modeling node polysemy. The experiment results are shown in Figure 4.4. Some observations are made as below:

- Increasing the number of facets has positive influence on the classification performance of the proposed model. It is also worth noting that we keep the total embedding dimension (i.e., $K \times D_{\text{PolyDeepwalk}}$) to be fixed, where $D_{\text{PolyDeepwalk}}$ actually decreases as K increases, so that the downstream classification task will not be affected by the feature size.

- In this part of experiment, changing the facet sampling rate for each observation (i.e., context window) does not significantly perturb the results. The possible reason is that many random walks have been generated from each node, so the facet sampling is also implicitly performed

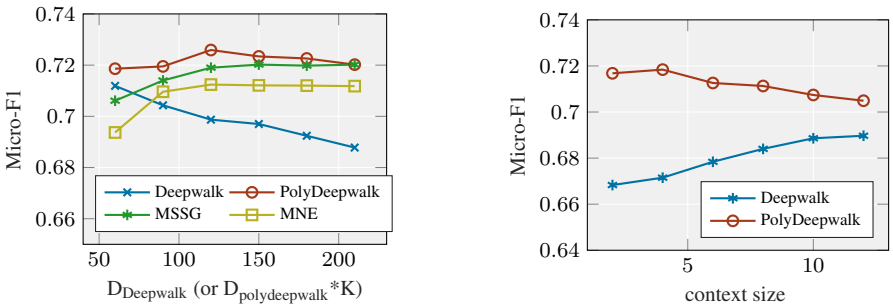


Figure 4.2: Node classification evaluation on BlogCatalog dataset. Reprinted from [4].

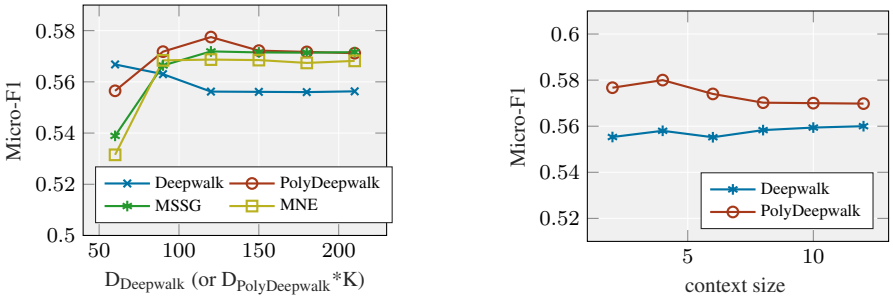


Figure 4.3: Node classification evaluation on Flickr dataset. Reprinted from [4].

for each random walk. The training samples are already adequate even when we reduce the facet sampling rate for each context window.

4.3.3 Link Prediction in Homogeneous Networks

Besides node classification, we also include link prediction as one of the tasks for evaluating the embedding results. We randomly select one link for each node as test data, and use the remaining network as training data. The default settings of hyperparameters are similar to those applied in the last experiment. The main difference lies on the embedding dimensionality, where we let the embedding dimension to be $D_{\text{Deepwalk}} = D_{\text{PolyDeepwalk}}$. In other words, the embedding space of PolyDeepwalk is the same as that of Deepwalk. The default dimension value, unless otherwise stated, is chosen as 150. The performance of link prediction is summarized in Table 4.2, from which we could observe that:

- There is no significant advantage for PolyDeepwalk on BlogCatalog dataset compared with

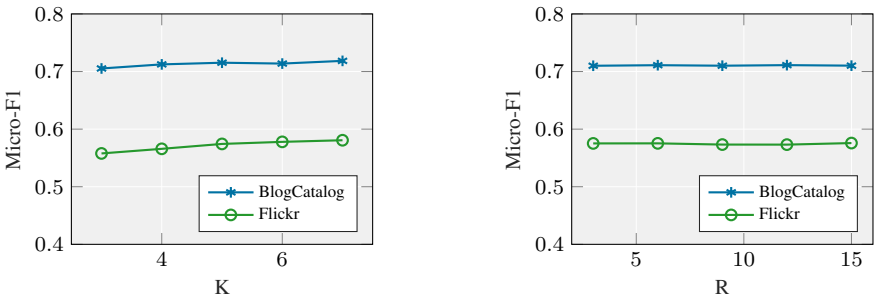


Figure 4.4: Parameter analysis for node classification. Reprinted from [4].

Deepwalk. Although PolyDeepwalk achieves better performance in terms of AUC score, Deepwalk scores higher when evaluated using $HR@k$ and k is small. However, it implicitly indicates that Deepwalk is gradually surpassed by PolyDeepwalk when k increases, which means PolyDeepwalk is better at recovering links that do not match the major connection patterns of a node.

- PolyDeepwalk performs better than NMF, which indicates that the effectiveness of polysemous embedding is not purely inherited from the matrix factorization step which provides prior knowledge of node facet distribution.

In addition, we analyze the sensitivity of PolyDeepwalk by varying the values of certain hyperparameters, including dimensionality, the number of facets (K), and the facet sampling rate (R) for each context window (Figure 4.5). Some phenomenons are observed as below:

- Network embedding models are sensitive to embedding dimension in link prediction tasks, as we can observe a clear growth of AUC score when embedding dimension increases.

- Increasing the number of facets K appropriately could boost link prediction performance. However, increasing K will also augment the embedding storage and inference computation cost. We should better keep an appropriate K value, in order to balance the performance and efficiency.

- Similar to what we have discussed in node classification, increasing the facet sampling rate does not bring much benefits. Therefore, when the number of walks per node is large enough, we can keep the face sampling rate to be low in order to reduce computation costs.

	BlogCatalog					Flickr				
	HR@10	**HR@50**	**HR@100**	**HR@200**	**AUC**	**HR@10**	**HR@50**	**HR@100**	**HR@200**	**AUC**
PolyDW	0.234	0.493	0.615	0.737	0.957	0.210	0.395	0.487	0.590	0.928
DW	0.253	0.512	0.639	0.764	0.950	0.195	0.348	0.430	0.530	0.912
NMF	0.041	0.132	0.196	0.280	0.801	0.048	0.148	0.226	0.335	0.852
MSSG	0.102	0.272	0.376	0.505	0.910	0.071	0.152	0.207	0.268	0.811
MNE	0.208	0.466	0.571	0.684	0.939	0.175	0.318	0.416	0.523	0.901

Table 4.2: Performance of homogeneous link prediction. Reprinted from [4].

	MovieLens					Pinterest				
	HR@10	**HR@50**	**HR@100**	**HR@200**	**AUC**	**HR@10**	**HR@50**	**HR@100**	**HR@200**	**AUC**
PolyPTE	0.067	0.210	0.334	0.491	0.892	0.062	0.204	0.318	0.460	0.919
PTE	0.040	0.143	0.227	0.336	0.832	0.007	0.030	0.052	0.088	0.703
PolyGCN	0.050	0.158	0.250	0.389	0.863	0.057	0.198	0.251	0.455	0.912
GCN	0.039	0.130	0.204	0.313	0.813	0.051	0.187	0.262	0.430	0.902
NMF	0.051	0.165	0.265	0.404	0.865	0.005	0.021	0.040	0.075	0.654
MSSG	0.060	0.181	0.288	0.427	0.868	0.039	0.128	0.199	0.304	0.838
MNE	0.056	0.195	0.301	0.459	0.880	0.043	0.159	0.262	0.433	0.894

Table 4.3: Performance of heterogeneous link prediction. Reprinted from [4].

4.3.4 Link Prediction in Heterogeneous Networks

In some application scenarios such as recommender systems, each link stretches across two types of nodes. Here we construct two bipartite networks from two recommendation datasets: MovieLens and Pinterest. Models such as PTE and unsupervised GCN can handle such scenario, and we extend them as PolyPTE and PolyGCN to be evaluated.

The default values of hyperparameters are as follows. For both datasets, the number of facets K is 5, the embedding dimension is set as 30, and the facet sampling rate R for each observation equals K^2. The negative sampling rate is set as 30 for PolyPTE. Different from the last experiment where $D_{\text{PolyDeepwalk}} = D_{\text{Deepwalk}}$, the embedding dimension in this experiment is set as D/K. Performances of heterogeneous link prediction are shown in Table 4.3, where we can observe that:

- Polysemous embedding models achieves better performances than their single-embedding counterparts. It thus suggests that considering node polysemy is beneficial for modeling the associa-

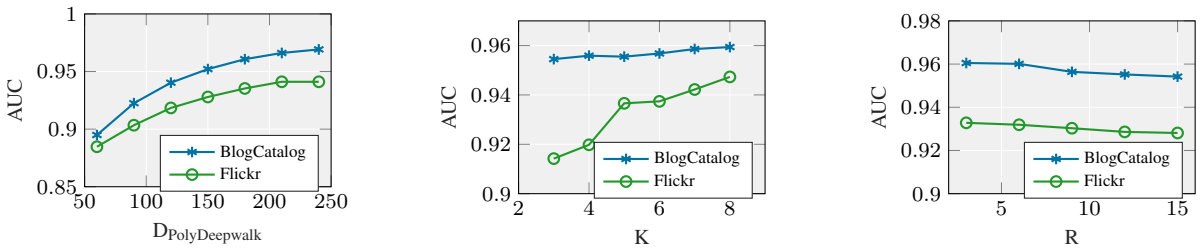


Figure 4.5: Parameter analysis for link prediction on homogeneous networks. Reprinted from [4].

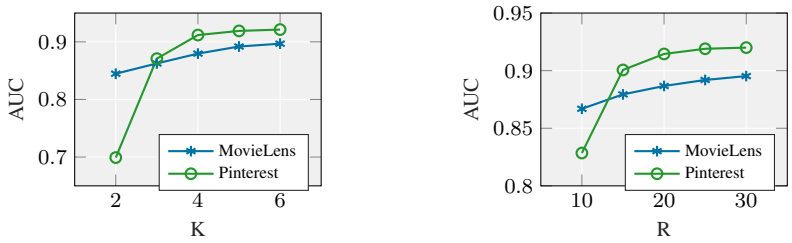


Figure 4.6: Parameter analysis for PolyPTE. Reprinted from [4].

tion relation between different types of node entities.

- Polysemous embedding models in general perform better than NMF, which demonstrates that the effectiveness of polysemous models comes beyond the prior knowledge obtained from NMF.
- PolyGCN is not as good as PolyPTE in performance boosting. A possible reason is that we do not consider inter-facet relations in PolyGCN. However, GCN-based models still have much room for improvement, especially when attribute information is available.

Similar to previous experiment tasks, here we also analyze the sensitivity of PolyPTE to model hyperparameters, as shown in Figure 4.6 for MovieLens and Pinterest. Besides some similar observations as in the last experiment, PolyPTE is sensitive to the facet sampling rate, especially when its value is low. A possible reason for such contrasting observations between PolyDeepwalk and PolyPTE could be the different forms of observation samples $o \in \mathcal{O}$. PolyDeepwalk samples plenty of random walks from each node (i.e., $o = \{\mathcal{N}(v), v\}$), while PolyPTE uses edge based sampling strategy (i.e., $o = (v_i, v_j)$). Therefore, if the facet sampling rate is low, PolyPTE could not fully consider all the possible correlation patterns between different facets of users and items.

5. ROBUSTIFYING MODELS BY UTILIZING INTERPRETATION [1]

Existing work on model interpretability mainly focuses on obtaining interpretation from machine learning models and developing interpretable components. However, it has not been well studied on how to exploit the obtained interpretation.

In this work, we explore the relation between interpretation and adversaries in order to improve model robustness. Specifically, I propose that interpretation is helpful in discovering the vulnerabilities in models, which is crucial in security-related applications. I choose spammer detection in the web security domain as the application scenario, where intelligent spammers (i.e., adversaries) can adaptively change their behaviors to avoid being detected by the deployed models. I developed an adversary-resistant model by utilizing interpretation to find model vulnerabilities and prepare the model in advance against the potential threats from malicious accounts.

5.1 Problem Statement

We first introduce the notations and application scenarios in this work. Let $\{\mathbf{X} \in \mathbb{R}^{N\times M}, \mathbf{y} \in \mathbb{R}^{N}\}$ be the data about user instances without considering adversaries, where N denotes the number of users and M is the feature dimension. The vector $\mathbf{x}_n = \mathbf{X}[n, :]$ $(1 \leq n \leq N)$ represents instance n in the feature space, $y_n = \mathbf{y}[n]$ is the label, and x_b denotes the b-th feature. In this work, we consider *social spammer* detection [127, 128, 129] as the scenario, but the methodology can be easily generalized to detection tasks against other malevolent entities. We set $y_n = 1$ for spammers and $y_n = 0$ for normal users. Then our goal is to build a spammer classifier $f(\mathbf{x}) : \mathbb{R}^M \rightarrow \{0, 1\}$, which is robust to future adversaries that may evolve (i.e., change their features) to evade the detection of f. We use existing spammer samples to predict the probable perturbation of adversaries. Let $\mathbf{x}^*$ be the adversary sample in feature space after behavior shifts of the original $\mathbf{x}$, where $\Delta\mathbf{x} = \mathbf{x}^* - \mathbf{x}$ denotes the perturbation in the feature space corresponding to the evasive actions

[1] Part of this chapter is reprinted with permission from "Adversarial detection with model interpretation" by Ninghao Liu, Hongxia Yang, and Xia Hu, Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Pages 1803–1811, Copyright 2018 by Association for Computing Machinery.

taken by the adversary. An adversary attempts to make the detector classify it as a normal account. We use $\mathbf{X}^* \in \mathbb{R}^{N' \times M}$ to denote the data matrix for adversary accounts, and N' is the number of adversaries considered. It is worth noting that, in this paper, "adversaries" and "spammers" both refer to the malicious accounts in systems, but adversaries are a specific subset of spammers who actively attempt to evade the inspection of detectors. In practice, adversaries can either poison the training data before a classifier is established [130, 131] or try to avoid being detected by the deployed classifier, while in this paper we only focus on the latter scenario.

We use adversarial training [79] as the fundamental defensive technique to improve the robustness of spammer classifiers against adversaries. The classifier is trained with a mixture of original and adversarial data instances. The overall loss function $\tilde{L}$ is formulated as below:

$$\tilde{L} = \alpha \cdot L(f, \mathbf{X}, \mathbf{y}) + (1 - \alpha) \cdot \sum_{\mathbf{x}^* \in \mathbf{X}^*} l(f, \mathbf{x}^*, 1), \tag{5.1}$$

where L and l measure dataset-level and instance-level classification errors, respectively. The parameter α controls the tradeoff between original and adversarial instances. We set $\alpha = 0.5$ in our experiments. The labels in the second term are fixed as 1 since they refer to spammers. In deep neural networks, it has been shown that $\mathbf{X}^*$ acts as the regularization term to increase model generality [76]. From another perspective, in our problem, $\mathbf{X}^*$ contains additional information of malicious instances after evolution, who are likely to reduce the effectiveness of the existing classifier. Unlike previous adversarial classification models that target certain types of classifiers [73, 75, 132, 133], the above formulation is independent of the type of the classifier f to be deployed.

5.2 Interpretation-Based Attack Strategy

The adversarial training of detectors requires speculated adversarial instances as part of the training data, which however are not directly available. To solve this problem, we develop a framework of interpretation-based adversary strategy to approximate the possible adversarial actions of spammers. The framework is illustrated in Figure 5.1. Given a classifier f, a special subset of spot-

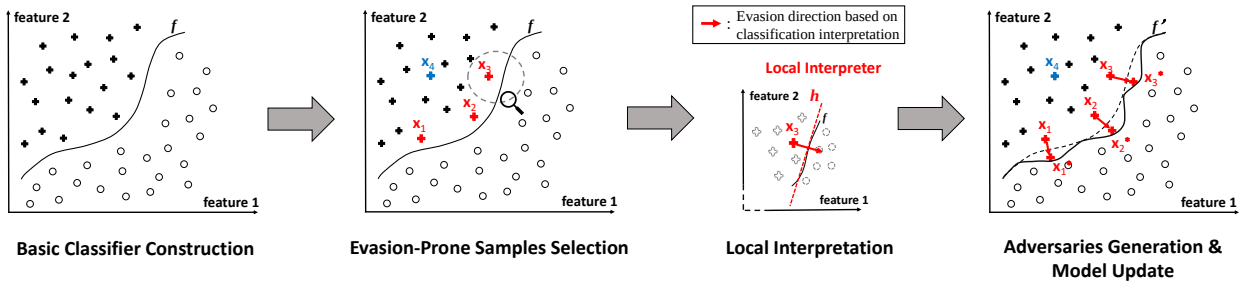


Figure 5.1: The framework of interpretation-based attack and defense based on evasion-prone samples. The dashed instances in the third image represent probing samples to explore the decision regions of the target classifier. Reprinted from [5].

ted malicious instances are chosen as the seeds (red points in the figure), and a local interpreter is then built to explain why a seed $\mathbf{x}$ is regarded as malicious by f. Adversarial samples $\mathbf{x}^*$ are then generated by perturbing each seed towards its evasion direction determined by the interpreter. The perturbation cost is constrained by the capability of adversaries. Finally, adversarial training with both original and adversarial data is used to obtain the new detector robust to adversaries.

5.2.1 Evasion-Prone Samples Selection

Spammer instances have different chances to evade detection, depending on the decision confidence assigned by the classifier. A classifier becomes vulnerable if many spammer instances fall into its low-confidence decision regions, since some small perturbation of these instances may result in misclassification of the detector. We want to pay more attention to the *evasion-prone* (EP) samples that are more likely to shift across the decision boundary. From the spammers' perspective, it requires less effort for them to wash the maliciousness off the instances, thus making it feasible for controlling a large number of accounts. From the classifiers' perspective, evasion-prone accounts can provide more information about how to improve the current model against the adversarial actions of spammers. In this paper, EP samples are used as the *seeds* for generating adversarial samples. An example can be found in Figure 5.1, where $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$ are evasion-prone samples, but not for $\mathbf{x}_4$.

We use the uncertainty score u_n [134] to measure the classification confidence for a given in-

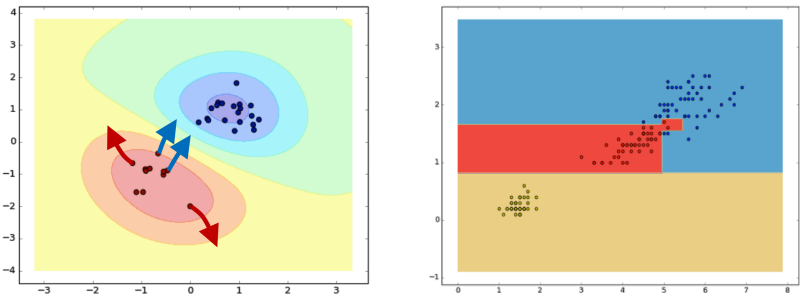


Figure 5.2: Left: Gradient descent based evasions targeting an SVM classifier [6]. Right: An example of decision regions of a three-class decsion tree classifier. Reprinted from [5].

stance $\mathbf{x}_n$ assigned by f. The concrete form of u_n varies according to the specific type of classifiers. For example, $u_n = 1 - P(y_n = 1|\mathbf{x}_n)$ for probabilistic models or neural networks whose output range is constrained, and $P(y_n = 1|\mathbf{x}_n)$ could also be the spammer class probability predicted by decision trees and ensemble models [135]. For linear models, $u_n = -(\mathbf{w} \cdot \mathbf{x}_n + b)$, where $\mathbf{w}$ and b are model parameters. In this work, we choose instances with high u_n scores as seeds for generating adversarial samples.

There are two widely adopted assumptions about whether adversaries have full knowledge of ML models. In some cybersecurity problems, ML models on the server side are assumed to be confidential [136, 83, 137]. In this case, attackers build their local substitute model after making enough queries to the server, generate adversarial samples on the local model, and rely on the transferability [76] of the adversarial samples to have them still effective on the original server models. In this paper, however, we aim to solve a classifier-oriented problem, so the full knowledge of the classifier is assumed to be known [73, 138, 78]. In this setting, the generated adversaries can target the most critical vulnerabilities of the classifier and initiate more threatening attacks. Note that our work can also be easily applied in black-box attacks by simply approximating the original model with the attacker-side substitute.

5.2.2 Local Interpretation Based Attack

For each evasion-prone seed, we need to find out its most sensitive perturbation direction for crafting the adversarial sample. Most existing methods perturb seeds along the direction of local

gradients of the loss function [6, 79, 139]. There are two problems for this class of methods. First, broader conditions of classification regions cannot be perceived by the local gradient, so gradient-based perturbation may be misled into unsupported regions. For example, on the left side of Figure 5.2, instances in red and blue regions will be recognized as spammers and normal users, respectively. The two samples moving along the red arrows will not be correctly directed towards blue regions to evade detection. Second, for some classifiers such as decision trees and ensemble methods, we cannot directly compute the gradient of loss function since a continuous prediction function is not available. As shown on the right side of Figure 5.2, the outputs of a tree-based model are discrete decisions, where gradients are not directly obtainable.

5.2.2.1 Local Approximation as Interpretation

We now introduce a new adversary samples crafting method based on the interpretation of ML models. Formally, given a seed instance $\mathbf{x}^s$ and the global classifier f, we define an explainable model $h \in H$ to locally approximate the prediction mechanism of f around $\mathbf{x}^s$. Here H denotes the class of explainable models, such as decision trees and linear models, or local vector flows constructed by Parzen window estimators [140]. The output of $h(\cdot) : \mathbb{R}^M \to \mathbb{R}$ measures the malicious score of the input instance. Let $l(f, h, \mathbf{x}^s)$ measure how close h is in approximating f in the space around $\mathbf{x}^s$, then the explanatory model h can be obtained by solving $\operatorname{argmin}_{h \in H} l(f, h, \mathbf{x}^s) + \beta \cdot c(h)$, where $c(h)$ measures the complexity of h. If h is chosen from linear classifiers, for example, then we can choose $c(h) = \|\mathbf{w}\|_1$, where $\mathbf{w}$ denotes the weight vector of linear models. In this paper, we define

$$l(f, h, \mathbf{x}^s) = \sum_{\mathbf{x}' \in \mathcal{X}'} \exp\left(\frac{-\|\mathbf{x}^s - \mathbf{x}'\|_2^2}{\eta^2}\right) (f(\mathbf{x}') - h(\mathbf{x}'))^2, \tag{5.2}$$

which is a weighted sum of approximation errors over a set of instances $\mathbf{x}'$ sampled around the seed $\mathbf{x}^s$, and η is the parameter controlling weight decay (parameter settings are discussed in experiments) [11]. The evasion-prone seeds introduced before facilitate the interpretation process, because they are close to the classification boundary so that it is easier when sampling $\mathbf{x}'$ to get both positive and negative labels for classification problems. The existence of negative samples

prevents perturbation from going to unsupported regions shown in Figure 5.2. After obtaining the local explainable model h, we rely on it to provide directions about how to perturb the seed $\mathbf{x}^s$.

5.2.2.2 Attack Strategy

With the local explanatory model h that measures the maliciousness of a given instance, the adversarial sample $\mathbf{x}^*$ corresponding to the seed $\mathbf{x}^s$ can be obtained by solving:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \ h(\mathbf{x}), \quad s.t. \ \mathbf{x} \in \Omega(\mathbf{x}^s)\,, \tag{5.3}$$

where $\Omega(\mathbf{x}^s)$ restricts the disturbance range around $\mathbf{x}^s$. The output $h(\mathbf{x}^*)$ is minimized so that the malicious score of the adversarial sample is maximally suppressed, which means we assume adversaries will try to stay as safe as possible by spending all of their budget. This strategy is different from the one applied in [76, 78], where adversaries try to minimize the perturbation as long as evasions are achieved. It is mainly because: (1) the former strategy considers the practical situation where adversaries have limited budget but want to achieve long-term survivals; (2) adversarial spammers assigned with low malicious scores are more threatening to the detector, so they are more helpful for model adjustment. The motivations of our strategy are that, on one hand, interpretation-based strategy has a broader understanding of f than gradient-based methods, so it can effectively lead perturbation into regions of lower maliciousness scores. On the other hand, the locality of h ensures its proximity to the global model f around $\mathbf{x}^s$, so that the adversarial samples targeting h are likely to be transferred to f [137].

5.2.2.3 Evasion Constraint

The constraint is defined as $\Omega(\mathbf{x}^s) = \{\mathbf{x} : d(\mathbf{x}, \mathbf{x}^s) \leq \zeta\}$. We will use l_2 and l_1 norms as two distance metrics for $d()$.

- l_2 norm constraint: We first use the commonly applied l_2 norm $d(\mathbf{x}, \mathbf{x}^s) = \|\mathbf{x} - \mathbf{x}^s\|_2$ to constrain attacks. Here we do not differentiate among the varying costs in perturbing different features. We consider this constraint in order to compare with other state-of-the-art attacking methods in

terms of effectively choosing the perturbation direction. For the l_2 norm constraint, we will use LASSO as the local interpretation model, i.e., h is chosen from linear models and $c(h) = \|\mathbf{w}\|_1$.

- l_1 norm constraint: We use l_1 norm constraint to model more practical attacking scenarios, where $\Omega(\mathbf{x}^s) = \{\mathbf{x} : \|(\mathbf{x} - \mathbf{x}^s) \circ \mathbf{e}\|_1 \leq \zeta\}$ and $\circ$ denotes element-wise multiplication. Here we assume each adversary is assigned with a universal cost budget ζ. And $e_b \in \mathbf{e}$ denotes the effort cost for one unit change of feature x_b. Adversary workers are constrained by their capability, i.e., the maximum total effort they would like to spend in manipulating each instance. The cost of changing the value of x_b can be related to the feature's robustness [141, 142]. A larger e_b suggests it is effort-consuming to change x_b for reversing the classification result. For l_1 norm constraint, we will try two local interpretation models: LASSO and Vector Flow model [140]. Since LASSO is a linear predictor, for each seed and its local interpretation vector $\mathbf{w}$, only the feature b with the largest $\frac{|w_b|}{e_b}$ will be manipulated. For Vector Flow model, it is constructed with a number of Parzen windows to the samples in decision regions. The problem to solve is formulated as:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \frac{\sum\limits_{i,y_i=1} k_\sigma(\mathbf{x} - \mathbf{x}'_i) - \sum\limits_{i,y_i=0} k_\sigma(\mathbf{x} - \mathbf{x}'_i)}{\sum\limits_{i} k_\sigma(\mathbf{x} - \mathbf{x}'_i)} \tag{5.4}$$
$$s.t.\ \|(\mathbf{x} - \mathbf{x}^s) \circ \mathbf{e}\|_1 \leq \zeta,$$

where $k_\sigma(\cdot)$ is the Gaussian kernel, and $\mathbf{x}'_i$ denotes the sample for building the local interpretation model. The only hyper-parameter σ can be determined through validation, where some samples are chosen to build Parzen windows and some others are generated for validation. We adopt the iterative gradient descent algorithm [6] to solve for $\mathbf{x}^*$ in Equation 5.4.

5.2.3 Adversary Costs Estimation

In practice, different features have varying costs to be manipulated. The costs refer to the profit loss of attackers or the efforts paid for initiating adversarial actions. Such costs originate from several aspects. First, one malicious worker in real world usually controls a large number

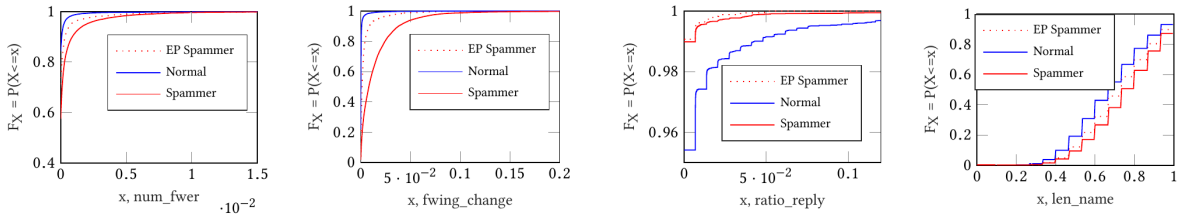


Figure 5.3: CDF of example features for overall non-spammers, lazy spammers and EP spammers in the Twitter dataset. Reprinted from [5].

of spammer accounts [143, 144]. It is harder to mimic certain aspects of normal users' behaviors, such as having diverse content in post streams or creating a lot of reviews recognized as useful by other customers. Second, spammers have different goals from legitimate users, in terms that they try to spread malicious or false content to deceive others and gain profit. For example, spammers seldom interact with other users online and they tend to include many URLs in their posts. Third, some designed features in traditional spammer detection are simply shallow artifacts extracted from datasets rather than intrinsic properties or behavior representations of accounts. Spammers can easily manipulate these features without much effort.

To estimate feature robustness, a straightforward way is to find the closed-form formula for each feature and explicitly compute the difficulty of changing its value [73, 142], or to unify the cost to a constant value [74]. These approaches, however, can only be applied to a limited number of straightforwardly designed features. Therefore, we design an empirical measure by relying on evasive spammers themselves to indicate the difficulty of controlling different aspects of their behaviors. We estimate an adversary's cost of manipulating feature x_b as:

$$e_b = \begin{cases} 1/|\overline{x_b^{ep}} - \overline{x_b^+}|, & \text{if } sign(\overline{x_b^{ep}} - \overline{x_b^+}) = sign(\overline{x_b^-} - \overline{x_b^+}) \\ e_{MAX}, & \text{Otherwise} \end{cases}, \tag{5.5}$$

where $\overline{x_b^-}$ denotes the average value of x_b for normal instances, $\overline{x_b^{ep}}$ is averaged over evasion-prone samples, and $\overline{x_b^+}$ averages x_b for spammers without the evasion-prone samples. The intuition of this equation is that, if evasion-prone samples have significantly shifted their x_b values to make them

	Twitter			YelpReview		
	Prec	Recall	F1	Prec	Recall	F1
LR	0.910 (0.867, 0.869)	0.889 (0.914, 0.915)	0.899 (0.890, 0.892)	0.794 (0.788, 0.776)	0.854 (0.890, 0.912)	0.823 (0.836, 0.838)
SVM	0.936 (0.911, 0.907)	0.928 (0.946, 0.944)	0.932 (0.929, 0.925)	0.905 (0.893, 0.887)	0.882 (0.890, 0.898)	0.893 (0.892, 0.893)
RF	**0.952 (0.952, 0.952)**	**0.962 (0.968, 0.965)**	**0.957 (0.960, 0.959)**	**0.988 (0.989, 0.985)**	**0.951 (0.961, 0.964)**	**0.969 (0.975, 0.974)**
NN	0.894 (0.879, 0.890)	0.958 (0.965, 0.968)	0.925 (0.920, 0.927)	0.937 (0.858, 0.905)	0.884 (0.919, 0.950)	0.910 (0.887, 0.927)
ESM	0.944 (0.927, 0.925)	0.943 (0.956, 0.961)	0.944 (0.941, 0.943)	0.970 (0.945, 0.959)	0.940 (0.956, 0.965)	0.955 (0.951, 0.962)

Table 5.1: Spammer detection performance of the proposed framework. Each triple represents: performance *without* adversarial training, performance *with* l_2 adversarial training, and performance *with* l_1 adversarial training. Reprinted from [5].

less malicious, then it indicates the cost of manipulating x_b is small, and vice versa. In addition, if the change direction of x_b^{ep} deviates from that of normal users, then we assume spammers are not willing to manipulate x_b and we set e_b with a large value e_{MAX}. In our experiments, we simply set $e_{MAX} = \infty$. More complex scenarios such as the correlation between features are beyond our scope in this study. We use evasion-prone samples here because they possess relatively strong incentives to evade the detection and have already taken some actions, so they provide information of how spammers may evolve.

Visualizations of the shift of EP instances are depicted in Figure 5.3, where the cumulative distribution function (CDF) of some example features are plotted. We can observe that, for *num_fwer* (number of followers), *fwing_change* (change speed of followings) and *len_name* (length of username), EP spammers are more similar to normal users as their CDF curves are closer than the lazy spammers. Especially, we find that $|\overline{x_b^{ep}} - \overline{x_b^+}|/|\overline{x_b^-} - \overline{x_b^+}| \approx 85\%$ for $x_b =$ *fwing_change*, so *fwing_change* may not be a robust feature with respect to potential attacks. However, for *ratio_reply*, the plot shows that the distribution difference becomes more remarkable for EP spammers, so it is not preferred to be manipulated by spammers. The feature manipulation costs are taken into consideration in our attack strategies under l_1 norm constraint. The effectiveness of the attacks, defenses and the roles of different features will be further discussed in our experiments.

5.3 Experiments

In this section, we evaluate the performance of the proposed method on several real-world datasets. First, we will introduce the experimental settings. Second, we will measure the effectiveness of attacks generated by the proposed method compared with other baseline attacking methods.

Third, we will measure the performance of adversarial training using the proposed method compared with the baseline defending methods.

5.3.1 Experimental Settings

Here we introduce the datasets extracted from several web platforms applied in our experiments, the basic classifiers as targets of adversaries, and the baseline methods for adversarial attacks and defenses.

Datasets: We use two public datasets for evaluation, including the Twitter dataset [145] and the YelpReview dataset [146]. The Twitter dataset, collected via social honeypot techniques, consists of the profile information and post streams of content polluters and legitimate users. The YelpReview dataset consists of reviewers' information and review posts about restaurants on Yelp. The labels of Twitter users and Yelp posts are available. We set Yelp reviewers with more than 10% spam posts as spammers. After preprocessing, for Twitter dataset, we have $41,499$ users and each user instance has 28 features, and for YelpReview dataset we have $16,941$ reviewers and each reviewer has 16 features.

Target Models: Spammer detection using traditional classification models has been extensively studied [128, 127, 145, 147]. Following previous work, to cover diverse types of models, we apply Logistic Regression (LR), Support Vector Machines (SVM), Random Forest (RF), Neural Network (NN) and Ensemble method (ESM) to detect spammers in our datasets. The performances of different models are shown in Table 5.1 (outside parentheses). The parameters of the models are determined via 5-fold cross-validation. YelpReview dataset is more complicated than Twitter, so the resultant classifiers for YelpReview are more complex. For examples, the weights of regularization terms in LR and SVM are set smaller for YelpReview, and the number of layers and the size of each layer are also larger in NN. These trained classifiers will be targeted by various attacking strategies, and will be improved by defensive methods.

Baseline Methods: The methods applied in our experiments can be categorized into two classes, i.e., attacking methods and defensive methods, as introduced below.

- Gradient Descent Attacking method (GDA) [6]: An attacking method moving samples along the

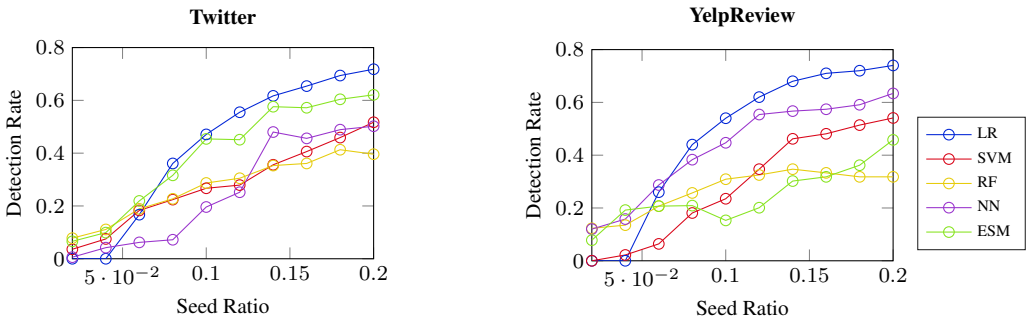


Figure 5.4: Attack effectiveness evaluation with different percentage of evasion-prone samples. Reprinted from [5].

direction of gradient vectors. The gradients or subgradients of the loss function are needed. The well-known Fast Gradient Sign method [79] can be seen as an special case of GDA applied on image data, where the input space is discrete.

- DeepFool [148]: A more effective attacking method by applying gradient-based attacks multiple times within the perturbation constraint.

- Defensive Distillation (DD) [84]: A defensive method by transfering the knowledge learned by a complex model f to a compact model f'. The soft labels generated from complex models f are used to train the new models [149]. It has been shown that, by making f' have the same model structure with f, the distilled model f' is more robust to adversarial samples than f.

However, it is hard to directly apply gradient-based methods (e.g., GDA and DeepFool) to attack tree-based models or ensemble models. Motivated by the black-box attacking strategy introduced in [150], we first approximate RF and ESM classifiers with a neural network as the substitute and then apply GDA and DeepFool for attacking.

5.3.2 Effectiveness of the Attacking Strategy

We evaluate the effectiveness of the proposed method by checking whether the crafted adversarial samples could successfully evade the detection of target classifiers. For local interpretation, we set $d_{pos-neg}/\sqrt{M}$ as the variance of multivariate Gaussian distributions for sampling $\mathbf{x}'$ around

Attack Success Rate	Method	LR	SVM	RF	NN	ESM
Twitter	GDA	**0.814**	0.961	0.702	0.860	0.716
	DeepFool	**0.814**	**0.968**	0.686	**0.893**	0.702
	Prop	0.803	0.958	**0.925**	0.878	**0.863**
YelpReview	GDA	**1.000**	0.964	0.769	0.902	0.798
	DeepFool	**1.000**	**0.970**	0.748	**0.947**	0.761
	Prop	0.996	0.961	**0.972**	0.905	**0.909**

Table 5.2: l_2 attack effectiveness evaluation on different types of classifiers. Reprinted from [5].

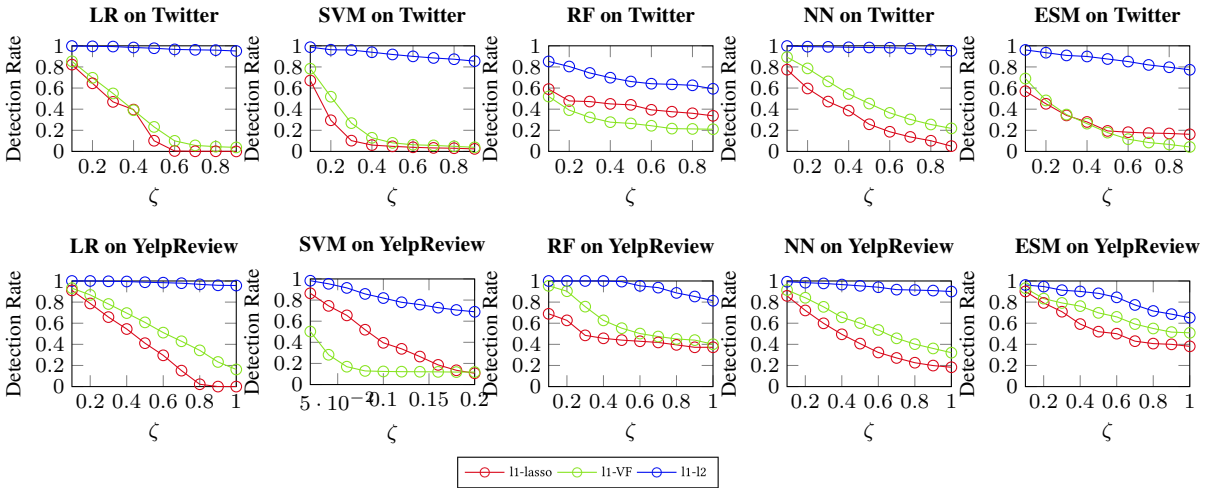


Figure 5.5: L_1 attack effectiveness evaluation with different perturbation distances on different types of classifiers. Reprinted from [5].

each seed, where $d_{pos-neg}$ is the seed's distance to the closest non-spammer instance and M is the number of features. The weight decay η is set as $+\infty$ in all experiments, so we treat all samples $\mathbf{x}'$ equally. The baseline attacking methods are implemented for comparison.

5.3.2.1 Effect of evasion-prone samples selection

To show the effect of applying evasion-prone samples, we measure the effectiveness of attacks with and without using evasion-prone samples as the seeds. The results averaged on five runs are shown in Figure 5.4. The *x* axis represents the portion of top spammers used as seeds for generating adversarial samples, and the *y* axis shows the detection rates of the target classifier. Data instances of high uncertainty scores are regarded as evasion-prone. Instances are ranked according to the

uncertainty score, where the top ones have higher scores. For better visualization, the adversarial perturbation distance is fixed as $0.1 \times d_{pos-neg}^{avg}$ for Twitter data and $0.2 \times d_{pos-neg}^{avg}$ for YelpReview data (except $0.04 \times d_{pos-neg}^{avg}$ in SVM), where $d_{pos-neg}^{avg}$ is the average of distance from seeds to their closest non-spammer neighbors. The figures show that, as we include more instances whose labels are more certain to the classifier, the success rate of evasion by the adversarial samples decreases (i.e., the detection accuracy of the classifier increases). If we randomly choose seeds for adversaries crafting, the evasion rate (not shown in figures) is usually less than 50%. Therefore, carefully choosing the seeds for generating adversarial samples will increase the effectiveness of attacks, thus providing more information about the weakness of the classifier.

5.3.2.2 Attack Effectiveness Under the l_2 Constraint

In this part, we explore how effectively the interpretation-based attack strategy under l_2 norm constraint would impair the performance of classifiers. Here feature manipulation costs are not considered. We will compare the proposed method with baseline methods on different classifiers. We choose top 4% evasion-prone samples as seeds for SVM and RF, 6% for ESM, 9% for NN and LR, which are applied to all attacking methods. The perturbation distance is fixed at $0.15 \times d_{pos-neg}^{avg}$. The results are shown in Table 5.2. In general, the proposed method significantly outperforms the baseline methods on RF and ESM models. The probable reason is that gradient-based methods cannot directly attack the original classifiers but the substitute models, which reduces their effectiveness. On LR, SVM and NN models where gradients are directly obtainable, the proposed method does not perform as well as gradient-based baseline methods, but the gap is small. DeepFool performs better than GDA on these models since DeepFool is an iterative algorithm. The overall evaluation thus shows the adaptability of the proposed method.

5.3.2.3 Attack Effectiveness Under the l_1 Constraint

We evaluate the attack performance of the proposed method with different local interpretation models under the l_1 norm constraint. Here feature manipulation costs are considered. The results are shown in Figure 5.5. Here the "*l1-l2*" method first generates adversarial samples under l_2

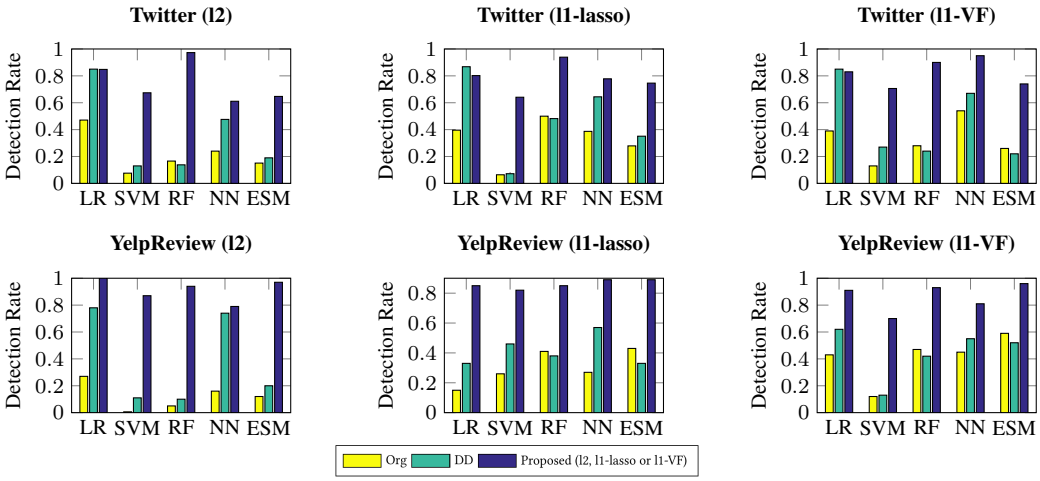


Figure 5.6: Defense effectiveness evaluation of proposed methods compared to Defensive Distillation. Reprinted from [5].

constraint, and then normalize the samples with respect to the costs specified by ζ. It is not a surprise that *l1-lasso* and *l1-VF* significantly outperform *l1-l2*, since the former ones select features of small costs to manipulate while the latter wastes effort in perturbing robust features which are of little reward. In general, the attack effectiveness of *l1-lasso* is greater than that of *l1-VF* as *l1-lasso* achieves lower detection rates on more classifiers. Though not shown in the figures, *l1-lasso* is more "concentrated" than *l1-VF* in terms that it is more focused on manipulating only a small number of features, and these features have low cost e_b. Let p_b denote the probability that feature b is perturbed by adversaries and $\mathbf{p} = [p_1, ..., p_B]$, $entropy(\mathbf{p}_{lasso}) = 0.11$ and 1.3 respectively in Twitter and YelpReview data, while $entropy(\mathbf{p}_{VF}) = 1.28$ and 1.87 for the two datasets. After inspecting the adversarial samples, we found that for Twitter data, nearly all adversaries choose to perturb the $fwing\_change$ feature. This is because its e_b is small (i.e., the feature is easy to be manipulated) and the gradient with respect to this feature is large (i.e., classifiers heavily rely on this feature to make decisions). In practice, slowing down the speed of following more users will not significantly affect the malevolent activities of spammers, and this feature tends to be controlled by adversaries. For YelpReview data, the manipulated features are relatively diverse across different samples. Some examples include *speedReviews* and *reviewCount*. It indicates that

spammers can suppress their activities to a safety zone, which is easier than changing other features such as *usefulCount* and *scoreDeviation* as these features are not fully under spammers' control.

5.3.3 Effectiveness of the Defensive Strategy

To evaluate the effectiveness of defense against adversaries, we will check whether the adversarial samples can be captured by the new classifiers after applying defensive strategies, and whether the classification results on the original data are not negatively affected by the shift of decision boundary. We use Defensive Distillation as the baseline method.

We first re-evaluate the performance of new models on the old test data, to check whether the prediction results are affected after adversarial training. The results are shown in Table 5.1. The performances before and after using adversarial training are shown inside and outside parentheses, respectively. Compared with the original classifiers, the new classifiers have higher Recall and lower Precision, because the adversarial samples used in training expand the region where an instance is classified as a spammer. The F1 scores do not significantly change. RF has the best detection performance and shows good compatibility to adversarial training. In addition, the increases in Recall and F1 score using l_1 adversarial samples are greater than using l_2 samples. The possible reason is that the consideration of feature manipulation costs leads the attacks towards the distribution of crafty spammer instances missed by the old classifiers.

We then compare the proposed method to Defensive Distillation concerning the ability to capture adversarial samples. Both l_2 and l_1 norms are involved. The results are shown in Figure 5.6. The perturbation distance in l_2 attack is $0.1 \times d^{avg}_{pos\_neg}$ for Twitter and $0.3 \times d^{avg}_{pos\_neg}$ for YelpReview, while the perturbation cost in l_1 attack is 0.4 for Twitter and 0.7 for YelpReview, except that the perturbation in SVM is further multiplied by 0.2 for visualization. In general, the capture rate of adversarial samples increases after applying defensive methods. The proposed method has better overall performance than Defensive Distillation. DD has good performance in NN and LR (could be seen as a one-layer NN), while it does not perform well in RF and ensemble methods. The proposed method is well adapted to a wide range of classification models, so it is helpful to proactively predict the evolution of malevolent entities and take it into consideration to train classifiers.

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this thesis, we develop a series of interpretation techniques towards improving model transparency of increasing levels, from unraveling feature-output relations, to understanding internal data representations inside models, developing inherently interpretable models, as well as utilizing interpretation towards building more robust models against adversaries.

First, we extend traditional feature-level interpretation to resolve local context of target predictions. Specifically, we apply the new method for model-agnostic outlier interpretation. We define the interpretation of an outlier from three aspects including the abnormal attributes, outlierness score and the outlier's context. For each outlier, we divide its neighbor data instances into several clusters. Interpretation is distilled and aggregated from the weight coefficients of a series of SVM classifiers between the outlier and each of the clusters. Prior knowledge in different applications can be incorporated with interpretation results to refine the outlier detection result.

Then, to complement feature-level interpretation that does not check the internal information of models, we propose a post-hoc interpretation framework to understand representations after feeding data into models. We formulate the problem as taxonomy induction from representation vectors. Specifically, the backbone of the taxonomy is constructed as a tree by performing hierarchical clustering all the representation vectors. We adopt spectral clustering to deal with the non-linearity in manifold. The characteristics of the clusters in taxonomy, i.e., the important attributes, are identified through multitask feature selection with tree-based regularization. Quantitative evaluations and case studies on real-world datasets demonstrate the effectiveness of the proposed framework.

However, post-hoc interpretation methods does not help gain control of model establishment. As an exploration of building interpretable models, we propose a polysemous network embedding method for considering multiple facets of nodes in network data. Each facet of a node owns an embedding segment, so that we could control the meaning of latent dimensions. The proposed

method is flexible as it can be used to extend various existing network embedding models. Before training embeddings, a clustering step is conducted and we assume each cluster corresponds to one facet. During the training process, different embedding segments are updated depending on which facet is activated in training samples. We also discuss how to combine multiple embedding segments during classification and link prediction. Experiments on real-world datasets comprehensively evaluate when we benefit from considering node polysemy compared with single-segment embedding baseline methods.

Finally, after obtaining interpretation, we propose an adversarial training based method to improve model robustness against adversaries. We consider malicious account detection on social media as the application scenario. The detection is modeled as a binary classification problem. We assume that malicious users will evolve in the future to avoid been detected by classifiers, which leads to the adversarial competition between malicious accounts and classifiers. We simulate probable adversarial attacks by utilizing the local interpretation of classifiers. The adversarial samples unveil the weakness of existing classifiers. The adversarial samples are added to normal data samples as data augmentation. Experiments show that models trained with the augmented data will be more robust to evolution of malicious accounts. Such an adversarial simulation method is better than traditional gradient-based methods as it can be applied to any type of classifiers such as tree-based models, where gradient is not directly available.

6.2 Future Work

My current research on interpretable machine learning poses a wealth of fascinating and challenging research questions that I plan to address in the short-term (first 3-5 years). My long-term goal is to create a human-machine ecosystem with responsible artificial intelligence and the ability to conduct causality learning.

6.2.1 Short-Term Plan

Interpretation Evaluation. Research on interpretable machine learning is growing rapidly in recent years, but not without growing pains. One such pain is the challenge of how to evaluate

the faithfulness (i.e., correctness) of interpretation, because there is no ground truth of "correct explanation". Some of my preliminary work [151, 152] tackle this problem via perturbation sensitivity analysis, but it is not a general solution. I will follow two principles in metric design. First, I will distinguish interpretation faithfulness from other properties such as intuitiveness, so human judgment should be excluded from the process. Second, I will propose rigorous quantitative definitions, so that an explicit goal is set for evaluation. In light of these, I plan to investigate two directions: (1) manually insert interpretation via backdoor techniques [153] as ground-truth and perform standard quantitative evaluations. (2) leverage causal inference as the unified framework to define interpretation.

Graph-Based Interpretable Model Design. Deep models are challenging to interpret, not only due to the large number of parameters and intricate feature interactions, but also because they operate on low-level features, rather than high-level, discrete, human-comprehensible concepts. To tackle this challenge, I will investigate using graph data as the foundation for interpretable model design. The investigation will be in two folds: (1) how to automatically construct an effective graph (i.e., nodes, edges, attributes) for a given application task? (2) how to jump out of the paradigm of end-to-end training, which is universal in the deep learning era, to accomplish knowledge extraction and model parameter training simultaneously? This research direction would have broad application scenarios, including visual understanding, natural language comprehension, interpretable recommender systems, and multi-modal learning.

User-Friendly Interface for Human-Computer Interaction. Most existing efforts on interpretable machine learning are devoted to developing more accurate interpretation methods, while the human experience aspect is usually overlooked. For end-users, friendly interpretation can help them make better and faster decisions, promote user experience, and gain trust to the system. For domain experts without machine learning background, an intuitive interface helps integrate them into the system improvement loop. To improve user-friendliness, I plan to borrow the experiences from the human-computer interaction (HCI) domain to design a desirable interpretation format. To better involve domain experts for human-in-the-loop learning, I plan to develop an interactive

interface, where models would utilize domain knowledge from experts for improvement.

6.2.2 Long-Term Plan

Responsible AI. Before being widely adopted in daily applications and production as replacement of human labor and decision-makers, it is necessary to guarantee that AI systems could take responsibility in terms of security assurance, and compliance to the law as well as code of ethics. However, recent studies have unveiled several vulnerable aspects of state-of-the-art AI models. For example, the security of deep models could be challenged by adversarial attacks and backdoor attacks. Also, it has been found that deep models sometimes capture unwanted bias (e.g., race, age, and gender) mainly due to its purely data-driven training paradigm. To tackle these issues, I will investigate using model interpretability to improve AI system transparency and strengthen regulation on models. I have done some preliminary work on adversarial defense [154], backdoor detection [153] and NLP model credibility promotion [155]. I will continue to investigate: (1) how to develop a comprehensive diagnosis tool with interpretation and visualization interface to identify model vulnerability? (2) how to effectively regularize a model by enforcing its interpretation to align with human-specified stipulation? (3) how to get rid of the end-to-end learning/inference paradigm of deep models, and use a more controllable strategy to learn more robust systems?

Bridging the Gap between Deep Learning and Reasoning. Will the popularity of one product drive the sales of another product? Will the traffic congestion of one region in a city affect the traffic flows of other regions? The answers to these questions require causal reasoning among entities, which goes beyond the conventional machine learning tasks that mainly focus on learning correlation. The ability to learn causality is considered as a significant component of human-level intelligence and serves as the foundation of AI. Existing deep learning models cannot well handle causal reasoning because they process data in continuous representation space, which is opaque and hard to interfere with human knowledge. In this research, I will explore several aspects of how to conduct reasoning with interpretable machine learning. First, can we transform data into knowledge base such as graphs and use desirable neural architectures for effective and interpretable inference? Second, can we revise adversarial attacks into meaningful perturbation for counterfac-

tual analysis in causal inference? I have done some preliminary work on graph-based interpretable modeling [152] and connecting adversarial learning with interpretation [154].

Promoting Interdisciplinary Research. I had very successful interdisciplinary collaboration experiences with researchers from both academia and industry, which not only broaden my visions but also inspired novel research ideas, as well as enabled the broad impact of my research. For example, with staffs of the Phoenix Veterans Affairs Health Care System, we studied how to apply machine learning to identifying impaired glucose tolerance participants who developed rapid carotid plaque progression. Model interpretation is a crucial building block in healthcare and medication-related domains, because high-stakes decisions are usually involved and knowledge from domain experts is abundant. I have also worked with engineers from Alibaba DAMO Academy to develop practical systems against fraudsters in online shopping websites. Specifically, we consider an adversarial context where the competition exists between fraudsters and system maintainers. Model interpretation can not only provide information to maintainers about why an account is regarded as malicious to avoid false positives, but it also indicates the possible actions taken by fraudsters to avoid being captured to provide direction of model improvement. Given the demand for developing transparent AI systems in healthcare, intelligent cities and cyber-security, in the future, I will seek opportunities for collaboration with industry in these domains.

REFERENCES

[1] N. Liu, D. Shin, and X. Hu, "Contextual outlier interpretation," *arXiv preprint arXiv:1711.10589*, 2017.

[2] N. Liu, X. Huang, J. Li, and X. Hu, "On interpretation of network embedding via taxonomy induction," KDD, 2018.

[3] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[4] N. Liu, Q. Tan, Y. Li, H. Yang, J. Zhou, and X. Hu, "Is a single vector enough? exploring node polysemy for network embedding," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[5] N. Liu, H. Yang, and X. Hu, "Adversarial detection with model interpretation," in *SIGKDD Conference on Knowledge Discovery and Data Mining*, 2018.

[6] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2013.

[7] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Conference on Computer Vision and Pattern Recognition*, 2016.

[8] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *ICCV*, 2017.

[9] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[10] M. Du, N. Liu, F. Yang, S. Ji, and X. Hu, "On attribution of recurrent neural network predictions via additive decomposition," in *International World Wide Web Conference*, pp. 383–393, 2019.

[11] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *SIGKDD Conference on Knowledge Discovery and Data Mining*, 2016.

[12] Z. Che, S. Purushotham, R. Khemani, and Y. Liu, "Distilling knowledge from deep networks with applications to healthcare domain," *arXiv preprint arXiv:1512.03542*, 2015.

[13] F. Wang, W. Zuo, L. Lin, D. Zhang, and L. Zhang, "Joint learning of single-image and cross-image representations for person re-identification," in *Conference on Computer Vision and Pattern Recognition*, 2016.

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

[15] M. Chen, "Efficient vector representation for documents through corruption," *arXiv preprint arXiv:1707.02377*, 2017.

[16] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *International World Wide Web Conference*, 2015.

[17] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2014.

[18] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Conference on Neural Information Processing Systems*, 2008.

[19] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Conference on Computer Vision and Pattern Recognition*, 2015.

[20] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, 2009.

[21] T. Chen, L.-A. Tang, Y. Sun, Z. Chen, and K. Zhang, "Entity embedding-based anomaly detection for heterogeneous categorical events," *arXiv preprint arXiv:1608.07502*, 2016.

[22] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[23] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *International World Wide Web Conference*, pp. 173–182, 2017.

[24] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *NMI*, 2019.

[25] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *DSP*, 2018.

[26] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *arXiv preprint arXiv:1808.00033*, 2018.

[27] Q. Zhang and S.-C. Zhu, "Visual interpretability for deep learning: a survey," *Frontiers of Information Technology Electronic Engineering*, 2018.

[28] Q. Zhang, R. Cao, F. Shi, Y. N. Wu, and S.-C. Zhu, "Interpreting cnn knowledge via an explanatory graph," in *AAAI Conference on Artificial Intelligence*, 2018.

[29] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[30] R. C. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," in *ICCV*, 2017.

[31] C. Guan, X. Wang, Q. Zhang, R. Chen, D. He, and X. Xie, "Towards a deep and unified understanding of deep neural models in nlp," in *Conference on Neural Information Processing Systems*, 2019.

[32] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: learning basic visual concepts with a constrained variational framework," in *ICLR*, 2017.

[33] J. Wang, J. Oh, H. Wang, and J. Wiens, "Learning credible models," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.

[34] M. Tsang, H. Liu, S. Purushotham, P. Murali, and Y. Liu, "Neural interaction transparency (nit): disentangling learned interactions for improved interpretability," in *Conference on Neural Information Processing Systems*, 2018.

[35] M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM sigmod record*, 2000.

[36] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," in *ACM Sigmod Record*, vol. 30, pp. 37–46, ACM, 2001.

[37] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2002.

[38] X. Song, M. Wu, C. Jermaine, and S. Ranka, "Conditional anomaly detection," *IEEE Transactions on Knowledge and Data Engineering*, 2007.

[39] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han, "On community outliers and their efficient detection in information networks," *SIGKDD Conference on Knowledge Discovery and Data Mining*, 2010.

[40] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-based outliers: algorithms and applications," *The VLDB Journal*, 2000.

[41] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *ACM SIGMOD Record*, 2000.

[42] S. D. Bay and M. Schwabacher, "Mining distance-based outliers in near linear time with randomization and a simple pruning rule," in *SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 29–38, ACM, 2003.

[43] F. Angiulli and F. Fassetti, "Dolphin: An efficient algorithm for mining distance-based outliers in very large datasets," *ACM Transactions on Knowledge Discovery from Data*, 2009.

[44] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data*, 2012.

[45] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[46] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," *Pattern Recognition Letters*, vol. 24, no. 9, pp. 1641–1650, 2003.

[47] H. Tong and C.-Y. Lin, "Non-negative residual matrix factorization with application to graph anomaly detection.," *SIAM International Conference on Data Mining*, 2011.

[48] P. Filzmoser, R. Maronna, and M. Werner, "Outlier identification in high dimensions," *CSDA*, 2008.

[49] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Transactions on Knowledge Discovery from Data*, 2009.

[50] B. Perozzi and L. Akoglu, "Scalable anomaly ranking of attributed neighborhoods," *SIAM International Conference on Data Mining*, 2016.

[51] A. Zimek, M. Gaudet, R. J. Campello, and J. Sander, "Subsampling for efficient and effective unsupervised outlier detection ensembles," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 428–436, ACM, 2013.

[52] J. Liang and S. Parthasarathy, "Robust contextual outlier detection: Where context meets sparsity," in *Conference on Information and Knowledge Management*, 2016.

[53] A. Lazarevic and V. Kumar, "Feature bagging for outlier detection," in *SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 157–166, ACM, 2005.

[54] B. Perozzi, L. Akoglu, P. Iglesias Sánchez, and E. Müller, "Focused clustering and outlier detection in large attributed graphs," *SIGKDD Conference on Knowledge Discovery and Data Mining*, 2014.

[55] N. Liu, X. Huang, and X. Hu, "Accelerated local anomaly detection via resolving attributed networks," in *International Joint Conferences on Artificial Intelligence*, 2017.

[56] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, 2013.

[57] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016.

[58] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *ACM International Conference on Web Search and Data Mining*, ACM, 2017.

[59] L. Liao, X. He, H. Zhang, and T.-S. Chua, "Attributed social network embedding," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[60] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding.," in *AAAI Conference on Artificial Intelligence*, 2017.

[61] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM.

[62] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2015.

[63] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec," in *ACM International Conference on Web Search and Data Mining*, ACM, 2018.

[64] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[65] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Conference on Neural Information Processing Systems*, 2017.

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

[67] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.

[68] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang, "Dynamic network embedding by modeling triadic closure process," 2018.

[69] X. Huang, Q. Song, J. Li, and X. Hu, "Exploring expert cognition for attributed network embedding," in *ACM International Conference on Web Search and Data Mining*, 2018.

[70] M. Grbovic and H. Cheng, "Real-time personalization using embeddings for search ranking at airbnb," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.

[71] H. Wang, C. Zhou, J. Wu, W. Dang, X. Zhu, and J. Wang, "Deep structure learning for fraud detection," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.

[72] D. Wang, M. Jiang, Q. Zeng, Z. Eberhart, and N. V. Chawla, "Multi-type itemset embedding for learning behavior success," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.

[73] N. Dalvi, P. Domingos, S. Sanghai, D. Verma, *et al.*, "Adversarial classification," in *SIGKDD Conference on Knowledge Discovery and Data Mining*, 2004.

[74] M. Brückner and T. Scheffer, "Stackelberg games for adversarial prediction problems," in *SIGKDD Conference on Knowledge Discovery and Data Mining*, 2011.

[75] M. Brückner and T. Scheffer, "Nash equilibria of static prediction games," in *Advances in neural information processing systems*, 2009.

[76] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[77] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[78] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Security and Privacy (SP), 2017 IEEE Symposium on*, IEEE, 2017.

[79] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *stat*, vol. 1050, p. 20, 2015.

[80] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Conference on Computer Vision and Pattern Recognition*, 2016.

[81] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[82] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387, IEEE, 2016.

[83] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[84] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Security and Privacy (SP), 2016 IEEE Symposium on*, IEEE, 2016.

[85] X. Cao and N. Z. Gong, "Mitigating evasion attacks to deep neural networks via region-based classification," *arXiv preprint arXiv:1709.05583*, 2017.

[86] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[87] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[88] N. Shah, "Flock: Combating astroturfing on livestreaming platforms," in *International World Wide Web Conference*, 2017.

[89] M. Jiang, A. Beutel, P. Cui, B. Hooi, S. Yang, and C. Faloutsos, "A general suspiciousness metric for dense blocks in multimodal data," in *IEEE International Conference on Data Mining*, pp. 781–786, IEEE, 2015.

[90] J. Ma and S. Perkins, "Online novelty detection on temporal sequences," in *SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 613–618, ACM, 2003.

[91] E. M. Knorr and R. T. Ng, "Finding intensional knowledge of distance-based outliers," in *International Conference on Very Large Data Bases*, 1999.

[92] B. Micenková, R. T. Ng, X.-H. Dang, and I. Assent, "Explaining outliers by subspace separability," in *IEEE International Conference on Data Mining*, 2013.

[93] L. Duan, G. Tang, J. Pei, J. Bailey, G. Dong, A. Campbell, and C. Tang, "Mining contrast subspaces," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2014.

[94] H. He and E. A. Garcia, "Learning from imbalanced data," *TKDE*, 2009.

[95] R. Tibshirani and G. Walther, "Cluster validation by prediction strength," *Journal of Computational and Graphical Statistics*, 2005.

[96] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, "1-norm support vector machines," *Conference on Neural Information Processing Systems*, vol. 16, no. 1, pp. 49–56, 2004.

[97] C. Yang, R. C. Harkreader, and G. Gu, "Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers," in *International Workshop on RAID*, 2011.

[98] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly, "Detecting spam web pages through content analysis," in *International World Wide Web Conference*, 2006.

[99] F. Keller, E. Muller, and K. Bohm, "Hics: high contrast subspaces for density-based outlier ranking," in *IEEE International Conference on Data Engineering*, IEEE, 2012.

[100] A. Asuncion and D. Newman, "UCI machine learning repository," 2007.

[101] N. X. Vinh, J. Chan, S. Romano, J. Bailey, C. Leckie, K. Ramamohanarao, and J. Pei, "Discovering outlying aspects in large datasets," *Data Mining and Knowledge Discovery*, 2016.

[102] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, 2007.

[103] C. Ding, X. He, and H. D. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering," in *SIAM International Conference on Data Mining*, SIAM, 2005.

[104] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural computation*, 2007.

[105] C.-J. Hsieh and I. S. Dhillon, "Fast coordinate descent methods with variable selection for non-negative matrix factorization," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2011.

[106] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, 2000.

[107] D. Kuang, S. Yun, and H. Park, "Symnmf: nonnegative low-rank approximation of a similarity matrix for graph clustering," *Journal of Global Optimization*, 2015.

[108] S. Kim and E. P. Xing, "Tree-guided group lasso for multi-task regression with structured sparsity," in *Conference on Neural Information Processing Systems*, 2010.

[109] A. Argyriou, T. Evgeniou, and M. Pontil, "Convex multi-task feature learning," *Machine Learning*, 2008.

[110] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[111] D. Kuang and H. Park, "Fast rank-2 nonnegative matrix factorization for hierarchical document clustering," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2013.

[112] Z. Li, Y. Yang, J. Liu, X. Zhou, H. Lu, *et al.*, "Unsupervised feature selection using nonnegative spectral analysis.," in *AAAI Conference on Artificial Intelligence*, 2012.

[113] D. M. Christopher, R. Prabhakar, and S. Hinrich, "Introduction to information retrieval," 2008.

[114] S. Zhu, K. Yu, Y. Chi, and Y. Gong, "Combining content and link for classification using matrix factorization," in *Special Interest Group on Information Retrieval*, 2007.

[115] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, 2008.

[116] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Conference on Neural Information Processing Systems*, 2013.

[117] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[118] D. Kuang, C. Ding, and H. Park, "Symmetric nonnegative matrix factorization for graph clustering," in *SIAM International Conference on Data Mining*, 2012.

[119] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Special Interest Group on Information Retrieval*, 2003.

[120] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *ICLR*, 2017.

[121] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *arXiv preprint arXiv:1808.00033*, 2018.

[122] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Special Interest Group on Information Retrieval*, 2016.

[123] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Conference on Neural Information Processing Systems*, 2001.

[124] A. Neelakantan, J. Shankar, A. Passos, and A. McCallum, "Efficient non-parametric estimation of multiple embeddings per word in vector space," in *EMNLP*, 2014.

[125] L. Yang, Y. Guo, and X. Cao, "Multi-facet network embedding: Beyond the general solution of detection and representation," in *AAAI Conference on Artificial Intelligence*, 2018.

[126] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu, "Attributed network embedding for learning in a dynamic environment," in *Conference on Information and Knowledge Management*, 2017.

[127] C. Castillo, M. Mendoza, and B. Poblete, "Information credibility on twitter," in *International World Wide Web Conference*, 2011.

[128] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting spammers on twitter," in *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, vol. 6, p. 12, 2010.

[129] A. Mukherjee, B. Liu, and N. Glance, "Spotting fake reviewer groups in consumer reviews," in *International World Wide Web Conference*, 2012.

[130] G. Wang, T. Wang, H. Zheng, and B. Y. Zhao, "Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers.," in *Usenix Security*, 2014.

[131] S. Alfeld, X. Zhu, and P. Barford, "Data poisoning attacks against autoregressive models.," in *AAAI Conference on Artificial Intelligence*, 2016.

[132] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Asian Conference on Machine Learning*, 2011.

[133] Y. Zhou, M. Kantarcioglu, B. Thuraisingham, and B. Xi, "Adversarial support vector machine learning," in *SIGKDD Conference on Knowledge Discovery and Data Mining*, 2012.

[134] B. Settles, "Active learning literature survey," *University of Wisconsin, Madison*, vol. 52, no. 55-66, p. 11, 2010.

[135] C. Bucilua, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *SIGKDD Conference on Knowledge Discovery and Data Mining*, 2006.

[136] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers," in *Proceedings of the 2016 Network and Distributed Systems Symposium*, 2016.

[137] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *USENIX Security*, 2016.

[138] S. Alfeld, X. Zhu, and P. Barford, "Explicit defense actions against test-set attacks.," in *AAAI Conference on Artificial Intelligence*, 2017.

[139] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[140] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. MÃžller, "How to explain individual classification decisions," *Journal of Machine Learning Research*, 2010.

[141] F. Nie, H. Huang, X. Cai, and C. H. Ding, "Efficient and robust feature selection via joint l2,1-norms minimization," in *Advances in neural information processing systems*, 2010.

[142] C. Yang, R. C. Harkreader, and G. Gu, "Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers," in *Recent Advances in Intrusion Detection*, pp. 318–337, Springer, 2011.

[143] S. Ghosh, B. Viswanath, F. Kooti, N. K. Sharma, G. Korlam, F. Benevenuto, N. Ganguly, and K. P. Gummadi, "Understanding and combating link farming in the twitter social network," in *International World Wide Web Conference*, 2012.

[144] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, "Copycatch: stopping group attacks by spotting lockstep behavior in social networks," in *International World Wide Web Conference*, 2013.

[145] K. Lee, B. D. Eoff, and J. Caverlee, "Seven months with the devils: A long-term study of content polluters on twitter.," in *International AAAI Conference on Web and Social Media*, 2011.

[146] A. Mukherjee, V. Venkataraman, B. Liu, and N. Glance, "Fake review detection: Classification and analysis of real and pseudo reviews," *Technical Report UIC-CS-2013–03, University of Illinois at Chicago, Tech. Rep.*, 2013.

[147] F. Li, M. Huang, Y. Yang, and X. Zhu, "Learning to identify review spam," in *International Joint Conference on Artificial Intelligence*, vol. 22, p. 2488, 2011.

[148] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[149] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *stat*, vol. 1050, p. 9, 2015.

[150] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against deep learning systems using adversarial examples," *arXiv preprint arXiv:1602.02697*, 2016.

[151] N. Liu, X. Huang, J. Li, and X. Hu, "On interpretation of network embedding via taxonomy induction," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1812–1820, 2018.

[152] N. Liu, Y. Ge, L. Li, X. Hu, R. Chen, and S.-H. Choi, "Explainable recommender systems via resolving learning representations," in *Conference on Information and Knowledge Management*, 2020.

[153] R. Tang, M. Du, N. Liu, F. Yang, and X. Hu, "An embarrassingly simple approach for trojan attack in deep neural networks," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 218–228, 2020.

[154] N. Liu, H. Yang, and X. Hu, "Adversarial detection with model interpretation," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.

[155] M. Du, N. Liu, F. Yang, and X. Hu, "Learning credible deep neural networks with rationale regularization," in *IEEE International Conference on Data Mining (ICDM)*, pp. 150–159, 2019.