Fakultät für Informatik
Technische Universität München

Machine Learning on Graphs in the Presence of Noise and Adversaries

Aleksandar Bojchevski

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzende:
Prof. Dr. Laura Leal-Taixé

Prüfende der Dissertation:
1. Prof. Dr. Stephan Günnemann
2. Prof. Dr. Stefanie Jegelka,
   Massachusetts Institute of Technology

Die Dissertation wurde am 01.10.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 24.11.2020 angenommen.

Abstract

From protein interactions to social networks, complex systems of interlinked entities are endemic in a connected world and graphs are a powerful abstraction for capturing their structure. Accordingly, we have a rich literature of machine learning techniques for graph data to solve problems ranging from fraud detection to cancer classification. Since in reality data is unreliable understanding the robustness of these techniques to noise and adversaries is critical. The contributions of this thesis deepen our understanding of robustness for three types of models: unsupervised, generative, and semi-supervised. First, we derive a noise-resilient variant of the classical spectral embedding, and we introduce Gaussian embeddings that represent nodes as distributions to capture uncertainty. Then we study the sensitivity of node embeddings to graph poisoning. Next, we develop a generative model that explicitly accounts for anomalies and detects clusters obfuscated by noise. Finally, we derive provable robustness guarantees. We propose the first certificate w.r.t. structure perturbations for a large class of PageRank-based models, and we derive a general certificate for discrete data applicable to any graph classifier.

Zusammenfassung

Von Protein-Interaktionen bis hin zu sozialen Netzerken, komplexe Systeme miteinander verbundener Einheiten sind allgegenwärtig in einer vernetzten Welt. Graphen können dabei als wirkmächtige Abstraktion dienen, um deren Strukturen zu erfassen. Dementsprechend gibt es eine umfassende Literatur zu Techniken im Bereich maschinelles Lernen mit Graphen, um Probleme wie Betrugserkennung oder Krebsklassifizierung zu lösen. Weil Daten in der Realität nicht verlässlich sind, ist es grundlegend wichtig, die Belastbarkeit dieser Techniken gegenüber Störungen und Angreifern zu verstehen. Die vorliegende Dissertation vertieft unser Verständnis von Robustheit für drei Modelltypen: unüberwachtes, generatives, und teilweise überwachtes Lernen. Zuerst wird eine Variante des klassischen Spektralen Einbettens entwickelt, die resilient ist gegenüber Rauschen in den Daten. Dabei werden Formen Gauß'scher Einbettung vorgestellt, die Knoten als Wahrscheinlichkeitsverteilung repräsentieren, um Unsicherheit zu erfassen. Danach wird die Sensitivität von Knoteneinbettungen gegenüber sogenannter Vergiftungs-Angriffe (engl. *poisoning attacks*) hin untersucht. Daraufhin wird ein generatives Modell entwickelt, das explizit Anomalien berücksichtigt und Cluster entdeckt, die durch Rauschen in den Daten verborgen sind. Schließlich werden mathematisch beweisbare Robustheitsgarantien abgeleitet und das erste Zertifikat mit Blick auf Strukturstörungen für eine große Klasse PageRank-basierter Modelle entwickelt. Zudem wird ein allgemeines Zertifikat für diskrete Daten abgeleitet, das auf jeden Graph-Klassifikator anwendbar ist.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my *Doktorvater* Prof. Stephan Günnemann for his continuing, unwavering support and encouragement. I could not have asked for a better supervisor and mentor. You are always incredibly generous with your time, and you always gave me invaluable advice. I am extremely lucky to have the opportunity to work with you, and I have learned so much from you. I will always cherish our research discussions as the favorite part of my PhD studies. Thank you.

I would like to thank Prof. Stefanie Jegelka and Prof. Laura Leal-Taixé for their time, for joining my dissertation committee, and for the insightful discussion during my defense.

All the exceptionally talented people at the DAML group have my profound thanks. You are the integral part that made TUM such an exciting and thriving work environment for me. Special thanks to Oleksandr Shchur. Your detailed and insightful feedback invariably improved my work, and your clarity of thought helped my own thinking. My sincere thanks to all my co-authors, especially Daniel Zügner and Johannes Klicpera. I admire your dedication and rigor, and I learned a lot from our collaborations.

I want to thank Jan, Yihan, Tom, Hakan, Lisa, Akshay, Maximillian, René, Mohamed, Dmytro, Thomas, and all of the other students that worked with me over the years. You have taught me so much, and without you my work would not be possible.

Thanks to all the wonderful people at the chair for database systems for their hospitality and guidance. Special thanks to my office mate Dimitri Vorona for the thought-provoking discussions, and to Dr. Juan Miguel Cejuela and Dr. Bryan Perozzi for their mentorship.

I thank my friends and family for their support on this journey as a first-generation graduate student. I am particularly grateful to Chris. You continue to make me a better scientist and a better person. I must also thank my dog for the emotional support.

Finally, to my parents: *Фала ви за вашата безгранична љубов и поддршка. Без сѐ што жртвувавте за мене не би бил денес овде, и не би го постигнал ова.*

Contents

Part I

Introduction

1 Introduction

In the early 1900s the German public was fascinated by the extraordinary mathematical ability of a horse named Clever Hans (der Kluge Hans). Hans was able to perform arithmetic and other intellectual tasks, and would provide the answer by tapping with his hoof the correct number of times.[1] Upon closer investigation however, it was discovered that the horse was giving the right answers by observing the minute reactions and the body language of the people around him. The eponymous Clever Hans effect, which has been broadly observed from sniffing dogs to reinforcement learning agents, can be summarized as the problem of learners solving a task by relying on spurious correlations.

With the increasing number of machine learning models deployed in safety-critical environments and decision-making contexts that involve humans, it is crucial to detect and mitigate similar effects in our models in order to ensure they are reliable. Accordingly, in this thesis we study the robustness of graph-based models to noise and adversaries.

1.1 Noise and Adversaries

While we can build models that show seemingly human-level performance (on narrowly defined tasks in laboratory conditions) as soon as the distribution of inputs differs even slightly from the one used during training[2] the behavior of most models is unpredictable. More importantly, adversaries can take advantage of this lack of robustness and easily craft *adversarial examples* – deliberate perturbations of the data designed to achieve a specific and often malicious goal. Even in scenarios where adversaries are not present we should examine the performance of our models in the worst case, i.e. treat nature as an adversary, since in reality data can be noisy, incomplete, anomalous, or simply different from what we have previously observed. Ideally, we would like our models to fail gracefully and provide guarantees about (the range of) their validity.

The sensitivity to noise, small adversarial perturbations, and more generally distribution shift between the training and test environment, has been conclusively demonstrated for many machine learning approaches from classic models like SVMs [15] to modern deep learning models [16]. However, research mostly has focused on scenarios assuming identically and independently distributed (i.i.d.) data such as images or tabular data. In contrast, in this thesis we focus on models for *graph* data where the i.i.d. assumption is explicitly violated which mandates a different set of techniques to study and improve their robustness. Despite receiving less attention relative to e.g. images studying the robustness of graph data, as we argue in the next section, is of utmost importance.

[1]If his owner Von Osten would ask Hans, “If the eighth day of the month comes on a Tuesday, what is the date of the following Friday?” Hans would answer by tapping his hoof eleven times.

[2]For example, data generated during a pandemic can be significantly different from normal.

1.2 Machine Learning on Graphs

Neural connections in the brain, social networks, interactions between proteins, and the structure of the Web seem to be entirely disparate phenomena at a first glance. Yet, if we abstract them as *graphs* – simply a set of nodes connected by a set of edges – we can view and study them through a common lens. This powerful abstraction dates back to 1736 when L. Euler used it to solve the famous problem of the seven bridges of Königsberg, laying the foundations of graph theory. Beyond a shared abstraction these systems of interconnected entities tend to exhibit similar structural patterns such as homophily or small-world effects which we can leverage to build broadly applicable models. Indeed, graph-based machine learning models are used across many domains for various high impact applications such as: drug discovery [17], fraud detection [18], categorization of scientific papers [19], traffic forecasting [20], and breast cancer classification [21].

Generally we can distinguish between two different scenarios: (i) we (partially) observe a single (large) graph, e.g. a social network, and aim to make inferences about individual nodes and the relations between them; (ii) we observe many (small) graphs, e.g. molecules, and aim to reason about each graph as a whole. In this thesis we mostly focus on the first scenario. We study the robustness of models for (semi-supervised) node classification, link prediction, (unsupervised) node representation learning, and clustering.

As an example consider a social network where each node represents a person and the edges encode friendship relations. Often we additionally observe a set of attributes for each node, such as the age of the person or their education. We refer to such data as attributed graphs.[3] One task might be to classify individual nodes in the graph, for example as either real user or bots, or to infer which edges are missing in the observed graph in order to recommend friends. If we are interested in the community structure of the network the task is to partition the set of nodes into (overlapping) clusters such that similar nodes are placed in the same cluster. For all these tasks we take into account both the network structure and the node attributes. For example, we may consider two people similar if they have many friends in common or if they have similar attributes. This similarity is (implicitly) leveraged to solve the downstream tasks.

1.2.1 Graph Neural Networks

One of the most effective ways to tackle the aforementioned tasks is with Graph Neural Networks (GNNs) [19, 22, 23]. They have emerged as a fundamental building block in many machine learning models, alongside CNNs and RNNs. Their inductive bias is relational – the hidden representation of a node is (recursively) computed based on its neighbors. Many GNN models fit into the message-passing framework proposed by Gilmer et al. [24]. At each layer, input transformation, e.g. a linear projection plus a non-linearity, is followed by an aggregation among neighbors, e.g. averaging. Increasing the number of layers increases the receptive field. The non-linear coupling between the node representations makes GNNs simultaneously powerful and difficult to analyze.

[3] We use the following terms interchangeably: graphs or networks, nodes or vertices, edges or links, and attributes or features.

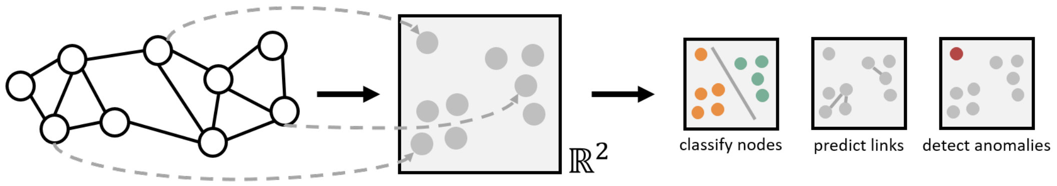


Figure 1.1: Illustration of graph representation learning. Each node is mapped to a low-dimensional vector. Similar nodes are close to each other. The embeddings are used to solve various downstream tasks such as: node classification, link prediction, and anomaly detection.

Beyond uses where the structure of the graph is explicitly given (e.g. social networks) GNNs show impressive performance for general object-oriented perception and reasoning [25–27]. However, in the rush to apply them to new and exciting domains questions about their robustness are routinely overlooked, and their reliability is poorly understood. We remedy this in Part IV where we study the robustness of GNNs to noise and adversaries.

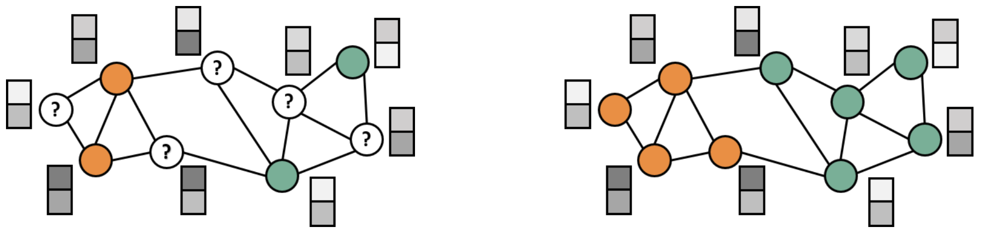
1.2.2 Unsupervised Representation Learning

The traditional approach for extracting features from graph data is based on user-defined heuristics such as degree statistics, motifs, and graph kernels. In contrast, the goal of network representation learning is to learn the features, also called representations or *embeddings*, directly from data, removing the need for manual feature engineering. For example, we can learn to map every node in the graph to a (low-dimensional) vector such that similar nodes are close to each other in the embedding space. The embeddings should preserve relevant information from the graph structure and the node features.

A big benefit of this approach is that we can use the same embeddings to solve many different downstream tasks (see illustration in Fig. 1.1), often achieving state-of-the-results and significantly improving upon traditional techniques [28]. Moreover, by operating in the embedding space we can employ existing learning techniques (e.g. classifiers) for vector data and bypass the difficulty of incorporating the graph structure. In this thesis, we are concerned with learning node embeddings that are robust to noise. We also show that popular node embedding techniques are vulnerable to adversarial examples.

1.2.3 Generative Models

Generative models for graphs have a longstanding history, with many applications including missing data imputation, simulation, and data augmentation. The goal is to capture the generative process behind the data and uncover latent factors. For example, we often observe community structure. That is, we can partition the set of nodes into communities such that the nodes within the same community have similar properties and are more likely to form edges compared to nodes that belong to different communities. With a generative model we can detect these latent communities and capture other common patterns such as power-law degree distribution or sparsity. Since real graphs



(a) Partially labeled attributed graph **(b)** Labels inferred by a model (e.g. a GNN)

Figure 1.2: Illustration of the semi-supervised node-classification task. Given a subset of labeled nodes the goal is to predict the class labels for the remaining nodes shown in different colors.

can be noisy and corrupted we design a generative model which additionally accounts for (partial) anomalies. We leverage this model to perform robust graph clustering.

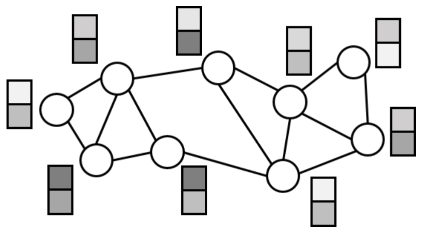
1.2.4 Semi-Supervised Learning

Given a subset of labelled nodes, e.g. a subset of nodes which are known to be either real users or bots in a social network, we aim to predict the class labels (real or bot) for the remaining nodes. This is the semi-supervised node classification task, also known as collective classification [29], which we illustrate in Fig. 1.2. Here we are in a transductive learning setting which means that both labeled and unlabeled nodes influence both training and inference. We take advantage of correlations between the label of a node and its own features, as well as correlations with the features and the (observed and unobserved) labels of the nodes in its neighborhood. A major benefit compared to standard supervised learning is that we can obtain excellent performance using very few labels. This is appealing since labels can be labor-intensive and expensive to obtain. In this thesis we focus on deriving robustness *certificates* – provable guarantees that no perturbation regarding a specific threat model will change the prediction of an instance.

1.3 Graphs and Robustness

A priori two hypotheses for how the relational nature of the data impacts robustness are plausible. On one hand, it might improve robustness since predictions are computed jointly rather than for each node in isolation. For example, even if the features of a given node are corrupted we might still be able to classify it correctly by taking advantage of the graph structure and the features of the neighboring nodes. On the other hand, the relational nature might cause cascading failures – perturbations in one part of the graph can propagate to the rest of the graph. Both effects manifest in practice. In Chapter 6 we show that integrating the graph structure and the attributes jointly improves the robustness of graph clustering, while in Chapter 5 we show that we can adversarially perturb a small part of the graph to significantly damage the quality of the embeddings.

The unique aspect of the graph domain is that noisy or adversarial perturbations can manifest in many different ways. Attackers can perturb the node features or modify the


(a) Clean (latent) graph

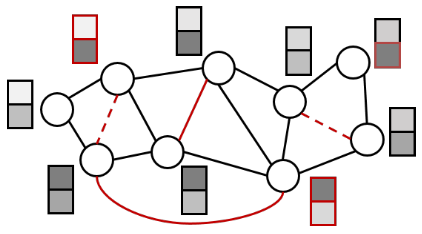

(b) Perturbed (observed) graph

Figure 1.3: Illustration of possible perturbations (marked in red) in a graph domain. Solid edges are spurious additions, dashed edges are spurious removals. Node attributes can also be corrupted.

graph structure, i.e. insert or remove edges (see Fig. 1.3). More importantly, perturbing even a single node could potentially influence the predictions for all other nodes in the graph. This means that even if the attacker does not have access to a certain target node they want to manipulate (direct attack) they can perturb other nodes under their control in order to achieve their goal (indirect attack) [30]. This is in contrast to non-relational data, where e.g. the predictions for a given image only depend on the pixel values of that image, and are independent of perturbations to other images.

Similarly, a few noisy edges could have a significant impact on all predictions. This is important to consider since often the process that gives rise to the graph is itself noisy. For example in a protein-protein interaction network edges encode associations between proteins which are derived from several (noisy) sources such as experimental data, text mining of scientific articles, and genomic context [31]. Moreover, whether an edge is inserted in the graph depends on an (arbitrary) threshold which reflects the likelihood that an association exist between two proteins. This holds even in cases where the data is “naturally” represented as a graph. For example in a social network an edge between two nodes could mean they have explicitly indicated their friendship, or that they have interacted a sufficient number of times (e.g. exchanged at least 10 messages), or some other arbitrary rule. In other words, we should not assume that the explicitly given structure of an observed graph is necessarily the ground-truth. Therefore, models for graph data should be robust to spurious edges and other noise.

Going back to the example of real users and bots, the attacker has a strong incentive to avoid detection so their fake accounts are not suspended. They attempt to find an adversarial perturbation of the graph which fools the model into classifying bots as real users. One naive way to achieve this would be to connect the fake accounts to a large number of real users and/or spend a lot of effort to make sure that the account details (i.e. node features) are realistic. This is inefficient and expensive. Instead, the attacker can specify a perturbation budget, e.g. at most B edge flips and feature corruptions in total, and search for a perturbed graph within their budget (by solving a constrained optimization problem) that fools the model. Any additional knowledge can be beneficial, e.g. they can encode the fact that for a two-layer GNN perturbing the neighbors which are three or more hops away from a target node will not have any effect.

1.4 Contributions and Outline

The contributions of this thesis deepen our understanding of the robustness of graph-based models along three complementary axes: representation learning, generative models, and semi-supervised learning. The overarching research question that we tackle is:

> What is the impact of noise and adversarial perturbations on machine learning models for graph data, and how can we improve and certify their robustness?

In this first part we introduce the main topic and in Chapter 2 we cover some necessary background and theoretical fundamentals. The main matter is organized in three parts.

- In Part II we tackle the robustness of graph representation learning. First, in Chapter 3 we derive RSC – a noise-resilient variant of the classical spectral embedding. We propose to decompose the (similarity) graph into a latent clean graph and sparse corruptions. We jointly learn the decomposition and the embedding, steered by the underlying clustering. Next, in Chapter 4 we introduce Gaussian embeddings that represent nodes as distributions to capture uncertainty. To learn the embeddings we adopt an unsupervised personalized ranking formulation w.r.t. node distances. Our method Graph2Gauss is inductive and generalizes to unseen nodes. Finally, in Chapter 5 we expose the sensitivity of node embeddings to graph poisoning. We derive efficient adversarial perturbations for a family of methods based on random walks, and we show that they are transferable to other models.

- In Part III, Chapter 6 we develop PAICAN – a probabilistic generative model that explicitly accounts for (partial) anomalies and detects clusters obfuscated by noise. The key observation is that even if one source of information is corrupted (e.g. the attributes for a given node are anomalous), by taking advantage of the other source (e.g. the network structure) we can still derive meaningful cluster assignments.

- In Part IV we study the vulnerability of semi-supervised models to adversarial examples. Specifically, in Chapter 7 we propose the first method for certifiable robustness to graph perturbations for a large class of models that includes label propagation and graph neural networks. By exploiting connections to PageRank and Markov decision processes our certificates can be efficiently (and for many threat models exactly) computed. In Chapter 8 we introduce a general method for certifying the robustness of machine leaning models for discrete data based on the randomized smoothing framework. Our certificate account for sparsity in the input which, as our findings show, is often essential for obtaining non-trivial guarantees.

We highlight that at the end of each chapter we add a *Retrospective* section where we discuss some limitations of the work and how we might overcome them, noteworthy follow up research, and other aspects which in hindsight turn out to be interesting or relevant.

Finally, in Part V we give concluding remarks and we point out open questions for future research. We also discuss how this work fits more broadly in the area of trustworthy machine learning and its potential (positive and negative) impact.

Table 1.1: List of own publications on which this thesis was based including the respective chapter. Each publication is associated with a project page: https://www.daml.in.tum.de/[project]/.

Ch.	Ref.	Title	Conference	Project Page
3	[1]	Robust Spectral Clustering for Noisy Data	KDD 2017	/rsc/
4	[2]	Deep Gaussian Embedding of Graphs	ICLR 2018	/g2g/
5	[3]	Adversarial Attacks on Node Embeddings	ICML 2019	/embedding-attack/
6	[4]	Bayesian Robust Attributed Graph Clustering	AAAI 2019	/paican/
7	[5]	Certifiable Robustness to Graph Perturbations	NeurIPS 2019	/graph-cert/
8	[6]	Efficient Robustness Certificates for Discrete Data	ICML 2020	/sparse-smoothing/

1.5 Own Publications

Parts of this thesis contain material previously published in the publications referenced in Table 1.1. Each publication is associated with a project page where you can find additional material such as: the original publication, code, data, presentations slides, videos, posters, demonstrations, and examples.

We also present the full list of publications which were published during the PhD, including non-first author papers and first author papers not part of this thesis:

[1] Aleksandar Bojchevski, Yves Matkovic, and Stephan Günnemann. Robust spectral clustering for noisy data: Modeling sparse corruptions improves latent embeddings. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2017

[2] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations, ICLR*, 2018

[3] Aleksandar Bojchevski and Stephan Günnemann. Adversarial attacks on node embeddings via graph poisoning. In *International Conference on Machine Learning, ICML*, 2019

[4] Aleksandar Bojchevski and Stephan Günnemann. Bayesian robust attributed graph clustering: Joint learning of partial anomalies and group structure. In *Conference on Artificial Intelligence, AAAI*, 2018

[5] Aleksandar Bojchevski and Stephan Günnemann. Certifiable robustness to graph perturbations. In *Neural Information Processing Systems, NeurIPS*, 2019

[6] Aleksandar Bojchevski, Johannes Klicpera, and Stephan Günnemann. Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more. In *International Conference on Machine Learning, ICML*, 2020

[7] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. NetGAN: Generating graphs via random walks. In *International Conference on Machine Learning, ICML*, 2018

[8] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate PageRank. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2020

[9] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Martin Blais, Amol Kapoor, Michal Lukasik, and Stephan Günnemann. Is PageRank all you need for scalable graph neural networks? In *International Workshop on Mining and Learning with Graphs, MLG*, 2019

[10] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized PageRank. In *International Conference on Learning Representations, ICLR*, 2019

[11] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. In *Relational Representation Learning Workshop, R2L*, 2018

[12] Oleksandr Shchur, Aleksandar Bojchevski, Mohamed Farghal, Stephan Günnemann, and Yusuf Saber. Anomaly detection in car-booking graphs. In *International Conference on Data Mining Workshops, ICDM*, 2018

[13] Eugenio Angriman, Alexander van der Grinten, Aleksandar Bojchevski, Daniel Zügner, Stephan Günnemann, and Henning Meyerhenke. Group centrality maximization for large-scale graphs. In *Symposium on Algorithm Engineering and Experiments, ALENEX*, 2020

[14] Federico Monti, Oleksandr Shchur, Aleksandar Bojchevski, Or Litany, Stephan Günnemann, and Michael M Bronstein. Dual-primal graph convolutional networks. In *Graph Embedding and Mining Workshop, GEM*, 2019

2 Background

In this thesis we draw from spectral graph theory, probabilistic generative models, and adversarial machine learning. This chapter gives a concise summary of the key concepts and the notation we use, but it is not intended to be a representative overview. We discuss eigenvalue perturbation theory and topic-sensitive PageRank since they recur in several chapters. We also cover the necessary background on adversarial examples and threat models. For a comprehensive overview of spectral graph theory see Mahoney [32], and for Markov Decision Processes see Kallenberg [33].

2.1 Graphs

Let $G = (\mathcal{V}, \mathcal{E})$ be an (attributed) graph where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. We denote with $\boldsymbol{A} \in \{0,1\}^{N \times N}$ the binary adjacency matrix where $\boldsymbol{A}_{ij} = 1$ if there is an edge between node i and node j and $\boldsymbol{A}_{ij} = 0$ otherwise. $N = |\mathcal{V}|$ denotes the number of nodes. In some cases we allow weighted graphs $\boldsymbol{A} \in \mathbb{R}_{\geq 0}^{N \times N}$with positive edge weights. $\boldsymbol{D}$ typically denotes the diagonal matrix of node out-degrees with $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$. For attributed graphs, let $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ be the matrix of D-dimensional features associated with each node. Sometimes we denote attributed graphs with $G = (\boldsymbol{A}, \boldsymbol{X})$.

2.2 Topic-sensitive PageRank

The topic-sensitive (personalized) PageRank [34, 35] vector $\boldsymbol{\pi}_G(\boldsymbol{z})$ for a graph G and a given probability (topic) distribution over nodes $\boldsymbol{z}$ is defined as

$$\boldsymbol{\pi}_{G,\alpha}(\boldsymbol{z}) = (1-\alpha)(\boldsymbol{I} - \alpha \boldsymbol{A}\boldsymbol{D}^{-1})^{-1}\boldsymbol{z} \tag{2.1}$$

where $\boldsymbol{I}$ is the identity matrix and $\alpha \in (0,1)$ is the damping parameter, or equivalently $(1-\alpha)$ is the teleport parameter. Intuitively, $\boldsymbol{\pi}(\boldsymbol{z})_u$ represent the probability of random walker on the graph to land at node u when it follows edges uniformly at random with probability α and teleports back to the node v with probability $(1-\alpha)\boldsymbol{z}_v$.

In practice, to compute $\boldsymbol{\pi}$ instead of matrix inversion we solve the associated sparse linear system of equations $(\boldsymbol{I} - \alpha\boldsymbol{P}) \cdot \boldsymbol{\pi} = (1-\alpha)\boldsymbol{z}$ where $\boldsymbol{P} = \boldsymbol{A}\boldsymbol{D}^{-1}$ is the column-stochastic transition matrix. Alternatively, we compute $\boldsymbol{\pi}$ with power iteration:

$$\boldsymbol{r}^{(0)} = \boldsymbol{z} \qquad \boldsymbol{r}^{(k)} = \alpha\boldsymbol{P}\boldsymbol{r}^{(k-1)} + (1-\alpha)\boldsymbol{z} \tag{2.2}$$

where $\boldsymbol{r}^{(\infty)} = \boldsymbol{\pi}(\boldsymbol{z})$, although the method converges in just a few iterations. To see this note that we can rewrite $\boldsymbol{\pi} = (1-\alpha)\sum_{k=1}^{\infty} \alpha^k \boldsymbol{P}^k \boldsymbol{z}$. The coefficients α^k quickly go to zero and have only a small contribution to the sum. We have $\boldsymbol{\pi}(\boldsymbol{z})_u \geq 0$ and $\sum_u \boldsymbol{\pi}(\boldsymbol{z})_u = 1$.

For $\boldsymbol{z} = \boldsymbol{e}_v$, where $\boldsymbol{e}_v$ is the v-th canonical basis vector we get the personalized PageRank vector for node v. Here the random walker teleports only to node v with probability $1-\alpha$. We drop the index on G, α and $\boldsymbol{z}$ in $\boldsymbol{\pi}_{G,\alpha}(\boldsymbol{z})$ when they are clear from the context. We denote with $\boldsymbol{\Pi} = (1-\alpha)(\boldsymbol{I} - \alpha \boldsymbol{P})^{-1}$ the personalized PageRank matrix, where row $\boldsymbol{\Pi}_{v,:} = \boldsymbol{\pi}(\boldsymbol{e}_v)$ equals to the personalized PageRank vector of node v.

2.2.1 PageRank and Label Propagation

Given a subset $\mathcal{V}_L \subseteq \mathcal{V}$ of labelled nodes the goal of semi-supervised node classification is to predict for each node $v \in \mathcal{V} \setminus \mathcal{V}_L$ one class in $\mathcal{C} = \{1, \dots, C\}$. We denote with $y_v \in \mathcal{C}$ the label for node v. Label propagation is a classic method to achieve this task and there have been many variants proposed over the years [36–38].

The general idea is to find a label assignment matrix $\boldsymbol{F} \in \mathbb{R}^{N \times C}$ such that the training nodes are predicted correctly ($\arg\max_c \boldsymbol{F}_{ic} = y_i$) and the predicted labels change smoothly w.r.t. the graph. The problem can be solved in closed form (even though in practice one would use power iteration) and the solution for the standard Laplacian variant [38] is:

$$\boldsymbol{F} = (1-\alpha)\left(\boldsymbol{I} - \alpha \boldsymbol{D}^{-1}\boldsymbol{A}\right)^{-1} \boldsymbol{H} = \boldsymbol{\Pi}\boldsymbol{H} \tag{2.3}$$

where $\boldsymbol{H} \in \{0,1\}^{N \times C}$ is a matrix where the rows are one-hot vectors for the training nodes and zero vectors otherwise (see Sec. F.1 for more details).

We see that solution to the label propagation problem can be obtained by simply multiplying the personalized PageRank matrix $\boldsymbol{\Pi}$ with the training label matrix $\boldsymbol{H}$. That is, the initial beliefs are diffused using the PageRank scores. We can interpret Eq. 2.3 in two equivalent ways. One view is that we first compute the personalized PageRank vector $\boldsymbol{\pi}(\boldsymbol{e}_v)$ for a given node v (i.e. the teleport vector is $\boldsymbol{z} = \boldsymbol{e}_v$) and then we compute the dot product with all columns of $\boldsymbol{H}$. We have $\boldsymbol{\pi}(\boldsymbol{e}_v)^T \boldsymbol{H}_{:,c} = \sum_{u \in \mathcal{V}_l, y_u = c} \boldsymbol{\pi}(\boldsymbol{e}_v)_c$, i.e. we sum the scores that a random walker which teleports to v assigns to the labeled nodes from class c. In the second view we set the teleport vector $\boldsymbol{z} = \boldsymbol{H}_{:,c}$ (normalizing it to sum to 1) and compute C (one per class) topic-sensitive PageRank vectors $\boldsymbol{\pi}(\boldsymbol{H}_{:,c})$. The prediction for node v is $\arg\max_c \boldsymbol{\pi}(\boldsymbol{H}_{:,c})_v$, where $\boldsymbol{\pi}(\boldsymbol{H}_{:,c})_v$ is the score assigned to v by a random walker that teleports to the labeled nodes in class c.

2.3 Eigenvalue Perturbation Theory

We say $\boldsymbol{u} \in \mathbb{R}^d$ is the generalized eigenvector associated with the generalized eigenvalue $\lambda \in \mathbb{R}$ for a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{d \times d}$ and a symmetric positive-definite matrix $\boldsymbol{D} \in \mathbb{R}^{d \times d}$ if they satisfy the following systems of equations:

$$\boldsymbol{A}\boldsymbol{u} = \lambda \boldsymbol{D}\boldsymbol{u} \qquad \text{and} \qquad \boldsymbol{u}^T \boldsymbol{D}\boldsymbol{u} = 1 \tag{2.4}$$

In matrix form we have $\boldsymbol{A}\boldsymbol{U} = \boldsymbol{D}\boldsymbol{U}\boldsymbol{\Lambda}$ where the columns of $\boldsymbol{U} = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_d]$ are the eigenvectors and $\boldsymbol{\Lambda} = \text{diag}([\lambda_1, \dots, \lambda_d])$ is a diagonal matrix with the eigenvalues. We assume that the eigenvalues are sorted in an non-decreasing order i.e. $\lambda_1 \leq \dots \leq \lambda_d$. It holds $\boldsymbol{U}^T\boldsymbol{D}\boldsymbol{U} = \boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix. For $\boldsymbol{D} = \boldsymbol{I}$ we recover the standard eigenvalue problem $\boldsymbol{A}\boldsymbol{u} = \lambda\boldsymbol{u}$.

Given symmetric perturbation matrices $\Delta\boldsymbol{A}$ and $\Delta\boldsymbol{D}$ we are interested in the solutions of the perturbed generalized eigenvalue problem $(\boldsymbol{A} + \Delta\boldsymbol{A})\boldsymbol{u}' = \lambda'(\boldsymbol{D} + \Delta\boldsymbol{D})\boldsymbol{u}'$. We restate the following first-order perturbation results [39]:

$$\lambda' \approx \lambda + \Delta\lambda \qquad \text{with} \qquad \Delta\lambda = \boldsymbol{u}^T(\Delta\boldsymbol{A} - \lambda\Delta\boldsymbol{D})\boldsymbol{u} \tag{2.5}$$

$$\boldsymbol{u}' \approx \boldsymbol{u} + \Delta\boldsymbol{u} \qquad \text{with} \qquad \Delta\boldsymbol{u} = -(\boldsymbol{A} - \lambda\boldsymbol{D})^+(\Delta\boldsymbol{A} - \Delta\lambda\boldsymbol{D} - \lambda\Delta\boldsymbol{D})\boldsymbol{u} \tag{2.6}$$

where $(\cdot)^+$ is the pseudo-inverse. Moreover, we can bound the difference in the eigenvalues using the absolute Weyl theorem for generalized eigenvalue problems [40]. We have:

$$|\lambda_i' - \lambda_i| \leq \|(\boldsymbol{D} + \Delta\boldsymbol{D})^{-1}\|_2\|\Delta\boldsymbol{A} - \lambda_i\Delta\boldsymbol{D}\|_2 \tag{2.7}$$

$$|\lambda_i' - \lambda_i| \leq \|\boldsymbol{D}^{-1}\|_2\|\Delta\boldsymbol{A} - \lambda_i\Delta\boldsymbol{D}\|_2 \tag{2.8}$$

for all $1 \leq i \leq d$. For the standard eigenvalue problem ($\boldsymbol{D} = \boldsymbol{I}, \Delta\boldsymbol{D} = \boldsymbol{0}$) these bound reduce to $|\lambda_i' - \lambda_i| \leq \|\Delta\boldsymbol{A}\|_2$. Using the Wielandt-Hoffman theorem [41] we also have

$$\sum_{i=1}^{d}|\lambda_i' - \lambda_i|^2 = \|\Delta\boldsymbol{A}\|_2^2 \tag{2.9}$$

From these results we can see that for a small (in terms of the norm) perturbation the change in the eigenvalues is also small. This suggest that the first-order approximation is meaningful. While higher-order approximations can be easily derived, in practice we observe that the gain is not worth the increase in computational complexity.

For our purposes $\boldsymbol{A}$ is the adjacency matrix of an undirected graph and $\boldsymbol{D}$ is the associated diagonal degree matrix. We are interested in the change in the spectrum resulting from changing the graph structure. For example if we flip (add or remove) a single undirected edge (i, j) then we have that $\Delta\boldsymbol{A} = (1 - 2\boldsymbol{A}_{ij})(\boldsymbol{e}_i\boldsymbol{e}_j^T + \boldsymbol{e}_j\boldsymbol{e}_i^T)$ and $\Delta\boldsymbol{D} = (1 - 2\boldsymbol{A}_{ij})(\boldsymbol{e}_i\boldsymbol{e}_i^T + \boldsymbol{e}_j\boldsymbol{e}_j^T)$ where $\boldsymbol{e}_i = \boldsymbol{I}_i$ is the i-th canonical basis vector. That is $\Delta\boldsymbol{A}$ and $\Delta\boldsymbol{D}$ have only two non-zero entries. Note that in this case Eq. 2.5 and Eq. 2.6 can be significantly simplified (see Chapter 5).

Unfortunately, for any connected graph and for any possible flip the above bounds are not informative since $-1 \leq \lambda_i \leq 1$ and $\|\Delta\boldsymbol{A}\|_2 = \|\Delta\boldsymbol{D}\|_2 = 1$. To obtain an alternative bound we can transform the generalized eigenvalue problem from Eq. 2.4 into the equivalent standard eigenvalue problem $\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{u} = \lambda\boldsymbol{u}$. Now we can compute $\Delta P = (\boldsymbol{D} + \Delta\boldsymbol{D})^{-1}(\boldsymbol{A} + \Delta\boldsymbol{A}) - \boldsymbol{D}^{-1}\boldsymbol{A}$ and evaluate the above bounds. Unlike before, now the bounds depend on the specific edge flip (and whether we add or remove an edge). For example, removing the edge between the two nodes with highest degree on Cora-ML yields a bound of ≤ 0.01361, while for the edge between two nodes with degree 2 the bound is ≤ 0.70711. This matches our intuition that changes to low-degree nodes have a much higher impact on the spectrum compared to changes to high-degree nodes.

The bounds are tight but still conservative, for the example of removing the edge between high-degree nodes using the exact λ' we have $\max_i|\lambda_i' - \lambda_i| \leq 0.000141$. Nonetheless, as we show in Chapter 3 and Chapter 5 the first-order estimates for $\Delta\lambda$ and $\Delta\boldsymbol{u}$ still yield a good approximation in practice for optimizing functions of the spectrum.

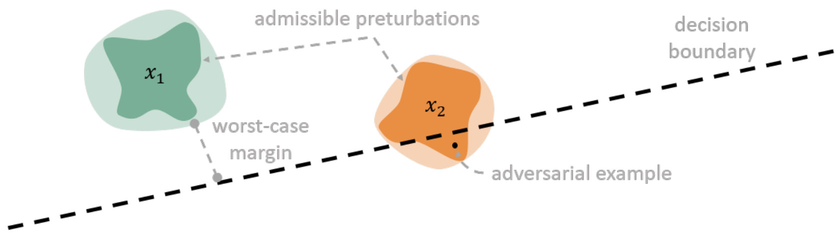


Figure 2.1: Illustration of adversarial examples. For the input $\boldsymbol{x}_1$ there is no other input in the set of admissible perturbations that crosses the decision boundary, i.e. the worst-case margin is positive. If we can *verify* that the worst-case margin (or a lower bound, illustrated with a lighter shade) is positive we say that $\boldsymbol{x}_1$ is certifiably robust. The input $\boldsymbol{x}_2$ is not robust since we can find a perturbed input within the admissible set that crosses the decision boundary.

2.4 Adversarial Examples

Adversarial examples explicitly demonstrate a lack of robustness. They can pose a serious threat for practitioners since attackers can leverage them to achieve malicious goals, e.g. bypass a bot detection system. As researchers, on the other hand, we can use them to better understand and mitigate the limitations of our models.

Given an admissible set of perturbations $\mathcal{B}$ encoding a threat model, e.g. all graphs which are reachable by changing a small number of edges in the observed graph, the goal is to find a perturbed input which causes the model to produce a given output. For example, the attacker looks for a perturbed graph such that a target node is misclassified:

$$\text{find} \qquad G' \in \mathcal{B} \qquad \text{such that} \qquad \arg\max_y f(G', t)_y \neq \arg\max_y f(G, t)_y \tag{2.10}$$

where $f(G, t)_y$ returns the probability that the target node t is classified as class y, and G is the unperturbed (clean) graph. Alternatively, the attacker's goal might be to decrease the overall quality of the learned embeddings, or to artificially increase the ranking of a target website (e.g. using link spam farms [42]).

Interestingly, defenders also benefit from adversarial examples. To date, the most successful defense strategies that can withstand strong attacks are based on adversarial training [43]. The idea is simple: the defender augments the training data with adversarial examples so the model can learn to correctly classified them. While such heuristic defenses work well in practice, there are no guarantees (which might be necessary in safety-critical applications) that some future attack will not be able to break them.

Robustness certificates are one way to avoid this cat-and-mouse game between the defender and attacker. A certificate is a provable guarantee that there exists no perturbation given a specific threat model that will lead to a certain outcome (e.g. misclassification). We compute the worse-case margin $m^*(t)$ for a target node t and a perturbation set $\mathcal{B}$:

$$m^*(t) = \min_{G' \in \mathcal{B}} m(G', t, y^*) = \min_{G' \in \mathcal{B}} \left(f(G', t)_{y^*} - \max_{y \neq y^*} f(G', t)_y \right) \tag{2.11}$$

where $y^* = \arg\max_y f(G, t)_y$ is the unperturbed prediction. If the worst-case $m^*(t) > 0$ is positive there is no $G' \in \mathcal{B}$ which changes the prediction and node t is certifiably robust.

See Fig. 2.1 for an illustration. Computing $m^*(t)$ is often intractable, especially when optimizing over a discrete and combinatorial graph domain, and when f is a complex non-convex function like a GNN. A typical approach is to relax the problem in Eq. 2.11 to a problem that is easier to solve (e.g. convex) which will give a lower bound on $m^*(t)$.

If the lower bound is positive we still have a valid certificate. If the lower bound is negative we cannot conclude that the node is necessarily not robust due to the relaxation. In this case, if can find a $G' \in \mathcal{B}$ for which the margin is negative we can certify non-robustness, e.g. if we also obtain the $\arg\min$ for the relaxed problem, or if we can find an adversarial example by other means. The gap between the ratio of certifiably robust and certifiably non-robust nodes can helps us reason about the tightness of the lower bound.

2.5 Threat Models

A key challenge when studying adversarial examples is specifying and optimizing for the right threat model. Generally, we can characterize the adversary based on their goals, knowledge, and capabilities. *Poisoning* attacks target the training data – the model is trained after the attack, while *evasion* attacks target the test data – the learned model is assumed fixed. We tackle both types of attacks in this thesis.

The poisoning setting is typically more challenging (and therefore less studied) since it often involves solving a difficult bi-level optimization problem. Note that for most models for i.i.d. data poisoning is only possible if the attacker can manipulate the labeled instances. However, for graph data since we are typically in a transductive learning setting both the labeled and unlabeled nodes influence the training and the learned model.

Depending on whether we assume that the attacker has knowledge about the model, and whether they have full access, query access, or no access to its parameters, its gradients, and the data, we can distinguish between white, gray and black-box attacks. While black-box attacks might be easier to carry out, studying the stronger (worse-case) white-box attacks is equally important in order to understand the intrinsic limitations of our models. This distinction might be less relevant in practice since we find that attacks are often transferable. That is, adversarial perturbations derived assuming one (surrogate) model can successfully damage other models previously unknown to the attacker. This holds true even if the attacker has only partial (or even no) access to the target data as evidenced by the existence of 'universal' adversarial perturbations [44].

For graph data the situation is more nuanced and we can distinguish between direct and indirect attacks since perturbing one node can potentially influence the predictions for all other nodes. For *direct* attacks we assume that the attacker has access to the target node and they can manipulate its edges and features. For *indirect* attacks we assume that the attacker controls some set of compromised nodes which they can leverage to indirectly influence a target node which is not under their control. We focus only on settings in which the size of the graph is fixed and the attacker can modify the edges between existing nodes and perturb the node features. Scenarios in which attackers can delete or add new nodes are interesting but are out of scope for this thesis.

In reality, perturbing the input is likely to incur some cost for the attacker. We model this with a global budget – the attacker can make at most B perturbations, i.e. add or delete at most B edges in total, or change at most B features. Furthermore, perturbing many edges for a single node might not be desirable since such a perturbation might be noticeable to defender. To ensure that the perturbation are unnoticeable we also model a local budget – we limit the number of perturbations locally per node. Typically we set the local budget relative to the node's clean degree.[1] More generally, the corruptions should be sparse compared to the clean input. For example, if a node has only 3 clean edges in its two-hop neighborhood and 30 adversarial edges, then it is reasonable and expected that a model will misclassify it since the (adversarial) noise dominates the (clean) signal.

[1]Other unnoticeability constraints such as making sure that the power-law exponent of the degree distribution is not significantly changed have also been explored [30].

Part II

Representation Learning

3 Robust Spectral Embedding

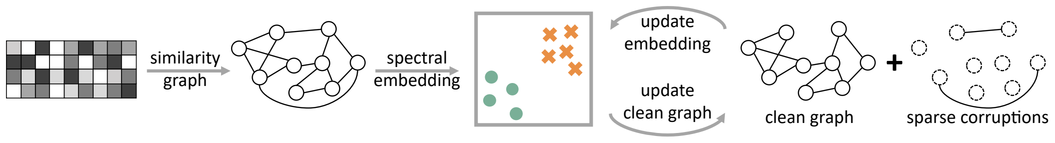


Figure 3.1: Overview of the proposed Robust Spectral Clustering (RSC) approach. Given a (similarity) graph we alternate between updating the spectral embedding based on the Laplacian of the clean graph, and updating the latent clean graph by removing sparse corruptions.

3.1 Introduction

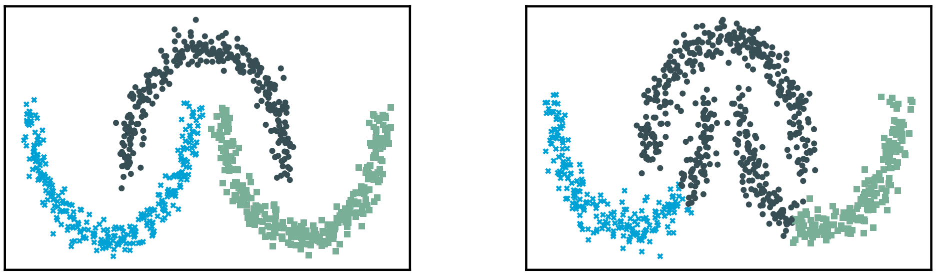
Clustering is a fundamental data mining task. Among the variety of methods that have been introduced in the literature [45], spectral clustering [46] is one of the most prominent and successful approaches. It has been successfully applied in many domains, ranging from computer vision [47] to bioinformatics [48, 49] and social network analysis [50].

Since spectral clustering relies on a similarity graph only (e.g. connecting each instance with its m nearest neighbors), it is applicable to almost any data type, with vector data being the most frequent case. It embeds the data into a vector space that is spanned by the k eigenvectors corresponding to the k smallest eigenvalues of the graph's (normalized) Laplacian matrix. By clustering in this space even complex structures living on a non-flat manifold, such as the half-moon data shown in Fig. 3.2, can be detected.

The spectral *embedding* is the key behind its success and can be derived from first principles, motivated by minimum-cut graph partitioning. Beyond clustering, we can use this embedding for graph visualization (spectral layout [51]), and more generally as a non-linear dimensionality reduction technique (Laplacian Eigenmaps [52]). In Chapter 4 we discuss an alternative approach where instead of embedding each node as a single point in some vector space, we represent it as a Gaussian distribution instead.

Even though spectral clustering is widely used in practice, especially in the sciences, one big issue is rarely addressed: it is highly sensitive to noisy input data. Fig. 3.2 illustrates this effect. While for the data on Fig. 3.2a spectral clustering perfectly recovers the ground-truth clusters, the scenario on Fig. 3.2b – with only slightly perturbed data – leads to a completely wrong clustering for any of the three established versions [46]. Spectral clustering fails in such noisy (and realistic) scenarios.

In this work, we introduce a principle to robustify spectral clustering. The core idea is that the observed similarity graph is not perfect but corrupted by errors. Thus, instead of operating on the original graph – or performing some, often arbitrary, data cleaning that



(a) Gaussian noise, $\sigma = 0.7$ **(b)** Gaussian noise, $\sigma = 0.9$

Figure 3.2: Spectral Clustering (SC) is sensitive to noisy input: (a) SC detects the correct clusters for the three moons data; (b) for only a slight perturbation SC fails. Our Robust Spectral Clustering (RSC) method is successful in both scenarios (not shown).

precedes the analysis – we assume the graph to be decomposed into two *latent* factors: the clean data and the corruptions. Assuming that the corruptions are sparse we *jointly* learn the latent corruptions and the latent spectral embedding using the clean data.

For tasks such as regression [53], PCA [54], and auto-regression [55, 56], such ideas have shown to significantly outperform non-robust techniques. Similarly, our Robust Spectral Clustering (RSC) method leads to clustering that is more robust to corruptions. On Fig. 3.2 our approach is able to detect the correct clustering structure (in both cases) despite the noise. More precisely, our work is based on a sparse latent decomposition of the graph with the aim to optimize the eigenspace of the graph's Laplacian. This is in strong contrast to, e.g., robust PCA where the decomposition is guided by the eigenspace of the data itself. In particular, different Laplacians affect the eigenspace differently and require different solutions. Our focus is not on finding the number of clusters automatically. Principles based e.g. on the largest eigenvalue gap [57] can similarly be applied to our work. We leave this aspect for future work. Overall, our contributions are:

- **Model**: We introduce a model for robust spectral clustering that handles noisy input data. Our principle is based on the idea of sparse latent decompositions. This is the first work exploiting this principle for spectral clustering, in particular tackling also the challenging case of normalized Laplacians.

- **Algorithms**: We provide algorithmic solutions for our model for all three established versions of spectral clustering using different Laplacian matrices. For our solutions we relate to principles such as eigenvalue perturbation and the multidimensional Knapsack problem. In each case, the complexity of the overall method is linear in the number of edges.

- **Experiments**: We conduct extensive experiments demonstrating the benefits of our method, with up to 15 percentage points improvement in accuracy on real-world data compared to standard spectral clustering. Moreover, we propose two novel measures – *local purity* and *global separation* – which enable us to evaluate the intrinsic quality of an embedding without relying on a specific clustering technique.

3.2 Preliminaries

We start with some basic definitions required in our work. Let $\boldsymbol{A}$ be a matrix, we denote with $\boldsymbol{a}_i$ the vector corresponding to the i-th row and with $\boldsymbol{A}_{ij}$ the value at position i, j. A similarity graph is represented by a symmetric adjacency matrix $\boldsymbol{A} \in (\mathbb{R}_{\geq 0})^{n \times n}$, with n being the number of instances. We denote the set of *undirected edges* as $\mathcal{E} = \{(i,j) \mid \boldsymbol{A}_{ij} > 0 \wedge i > j\}$. The set of edges incident to node i is given by $\mathcal{E}_i = \{(x,y) \in \mathcal{E} \mid x = i \vee y = i\}$. The vector representing the edges of $\boldsymbol{A}$ is written as $[\boldsymbol{A}_{ij}]_{(i,j)\in\mathcal{E}} = [\boldsymbol{A}_e]_{e\in\mathcal{E}}$.

We denote with $d_i = \sum_j \boldsymbol{A}_{ij}$ the degree of node i, and with $\boldsymbol{D}(\boldsymbol{A}) = \text{diag}(d_1, \ldots, d_n)$ the diagonal matrix representing all degrees. We denote with $\boldsymbol{I}$ the identity matrix, whose dimensionality becomes clear from the context. Furthermore, as required for spectral clustering, we introduce different notions of Laplacian matrices:

$$\boldsymbol{L}(\boldsymbol{A}) = \boldsymbol{D}(\boldsymbol{A}) - \boldsymbol{A} \tag{3.1}$$

$$\boldsymbol{L}_{\text{rw}}(\boldsymbol{A}) = \boldsymbol{D}(\boldsymbol{A})^{-1}\boldsymbol{L}(\boldsymbol{A}) = \boldsymbol{I} - \boldsymbol{D}(\boldsymbol{A})^{-1}\boldsymbol{A} \tag{3.2}$$

$$\boldsymbol{L}_{\text{sym}}(\boldsymbol{A}) = \boldsymbol{D}(\boldsymbol{A})^{-1/2}\boldsymbol{L}(\boldsymbol{A})\boldsymbol{D}(\boldsymbol{A})^{-1/2} = \boldsymbol{I} - \boldsymbol{D}(\boldsymbol{A})^{-1/2}\boldsymbol{A}\boldsymbol{D}(\boldsymbol{A})^{-1/2} \tag{3.3}$$

where $\boldsymbol{L}$ is the unnormalized Laplacian, $\boldsymbol{L}_{\text{rw}}$ is the normalized random walk Laplacian and $\boldsymbol{L}_{\text{sym}}$ is the normalized symmetric Laplacian.

3.2.1 Spectral Clustering

Spectral clustering can be briefly summarized in three steps (see [46] for details). First we construct the similarity graph $\boldsymbol{A}$. Different principles for the similarity graph construction exist. We focus on the symmetric x-nearest-neighbor graph, as it is recommended by [46] – any other construction can be used as well. Thus, the graph $\boldsymbol{A}$ is given by $\boldsymbol{A}_{ij} = 1$ if i is a x nearest neighbor of j or vice versa, and $\boldsymbol{A}_{ij} = 0$ else.

Depending on the considered Laplacian, the next step is to compute the following eigenvectors (corresponding to the k smallest eigenvalues):

- $\boldsymbol{L}(\boldsymbol{A})$: k first eigenvectors of $\boldsymbol{L}(\boldsymbol{A})$
- $\boldsymbol{L}_{\text{rw}}(\boldsymbol{A})$: k first generalized eigenvectors of $\boldsymbol{L}(\boldsymbol{A})\boldsymbol{u} = \lambda\boldsymbol{D}(\boldsymbol{A})\boldsymbol{u}$
- $\boldsymbol{L}_{\text{sym}}(\boldsymbol{A})$: k first eigenvectors of $\boldsymbol{L}_{\text{sym}}(\boldsymbol{A})$

This step stems from the fact that spectral clustering tries to obtain solutions that minimize the ratio-cut/normalized-cut in the similarity graph. As shown in [46], an approximation to the ratio-cut is obtained by the following trace minimization problem:

$$\min_{\boldsymbol{H}\in\mathbb{R}^{n\times k}} Tr(\boldsymbol{H}^T\boldsymbol{L}(\boldsymbol{A})\boldsymbol{H}) \text{ subject to } \boldsymbol{H}^T\boldsymbol{H} = \boldsymbol{I} \tag{3.4}$$

The solution being the k first eigenvectors of the Laplacian $\boldsymbol{L}(\boldsymbol{A})$ as stated above. Similar trace minimization problems can be formulated for the other Laplacians. We denote with $\boldsymbol{H} \in \mathbb{R}^{n\times k}$ the matrix storing the eigenvectors as columns.

The final step is to perform clustering on $\boldsymbol{H}$. The spectral embedding of each instance i is given by the i-th *row* of $\boldsymbol{H}$. To find the final clustering, the vectors $\boldsymbol{h}_i$ are (in case of $\boldsymbol{L}_{\text{sym}}(\boldsymbol{A})$ first normalized and then) clustered using, e.g., k-means.

3.3 Related Work

Multiple principles to improve spectral clustering have been introduced – focusing on different kinds of robustness. Surprisingly, many of the techniques [57–60] are based on fully connected similarity graphs – even though nearest-neighbor graphs are recommended [46]. First, using fully connected graphs highly increases the runtime – the considered matrices are no longer sparse – and, second, one has to select an appropriate scaling factor σ, required, e.g., for the Gaussian Kernel when constructing the graph [46]. Thus, many techniques [57–60] focus on robustness w.r.t. the scale σ.

Local similarity scaling [58] introduces a principle where the similarity is locally scaled per instance, i.e. the parameter σ changes per instance. By doing so, an improved similarity graph is obtained that better separates dense and sparse areas in the data space. Kong et al. [59] extended this principle by using a weighted local scaling. The methods work well on noise-free data, however, they are still sensitive to noisy inputs.

Laplacian smoothing [60] considers the problem of noisy data similar to our work, and they propose the principle of eigenvector smoothing. The initial Laplacian matrix is replaced by a smoothed version $\boldsymbol{M} = \sum_{i=2}^{n} \frac{1}{\gamma+\lambda_i} \boldsymbol{h}_i \cdot \boldsymbol{h}_i^T$ where $\boldsymbol{h}_i$ and λ_i are the eigenvectors/values of the original Laplacian matrix. Clustering is then performed on the eigenvectors of the matrix $\boldsymbol{M}$. A significant drawback is that a full eigenvalue decomposition is required.

Data warping [57] focuses on data where uniform noise has been added, not noisy data itself. They propose the principle of data warping. Intuitively, the data is transformed to a new space where noise points form their own cluster. Since they focus on fully connected graphs, noise can easily be detected by inspecting points with the lowest overall similarity. Since [60] and [57] are the most closely related works to our principle, we compare against them in our experiments.

Focusing on a different scenario, multiple works have considered noisy/irrelevant features. In [61] a global feature weighting is learned in a semi-supervised fashion, thus, leading to an improved similarity matrix. Zhu et al. [62] learn an affinity matrix based on random subspaces focusing on discriminative features. In [63], inspired by the idea of subspace clustering, feature weights are learned locally per cluster.

All the above techniques (except [63]) follow a two-step, *sequential* approach. They first construct an improved similarity graph/Laplacian and then apply standard spectral clustering. In contrast, our method *jointly* learns the similarity graph and the spectral embedding. Both steps repeatedly benefit from each other.

Besides the above works focusing on general spectral clustering, different extended formulations have been introduced: Li et al. [64] consider hypergraphs to improve robustness, Chang and Yeung [65] use path-based characteristics. None of the techniques jointly learns a similarity matrix and the spectral embedding. Not focusing on robustness w.r.t. noise, Wang et al. [66] compute a doubly stochastic matrix by imposing low-rank constraints on the graph's Laplacian. It is restricted to the unnormalized Laplacian and leads to dense graphs, making it impractical for large data. Moreover, works such as [67] consider the problem of finding anomalous subgraphs using spectral principles, again not focusing on the case of noise.

We note that the spectral analysis is not restricted to a graph's Laplacian as used in standard spectral clustering. The classical works of Davis-Kahan [68], for example, study the perturbation of a matrix $\boldsymbol{X}$ and the change of $\boldsymbol{X}$'s eigenspace. Following this line, [69] studies clustering based on the eigenspace of the adjacency matrix itself. In contrast, we focus on the change of the eigenspace of $\boldsymbol{L}$. In particular, we also consider the case of normalized Laplacians, which often lead to better results [46].[1]

3.4 Robust Spectral Clustering

Here we introduce the major principle behind our technique. For illustration purposes, we will start with spectral clustering based on the unnormalized Laplacian. The (more complex) principles for normalized Laplacians are described in Sec. 3.5.

Let $\boldsymbol{A} \in (\mathbb{R}_{\geq 0})^{n \times n}$ be the symmetric similarity graph extracted for the given data, with n being the number of instances in our data. Our main idea is that the similarity graph $\boldsymbol{A}$ is not perfect but might be corrupted (e.g. due to noisy input data). Any analysis performed on $\boldsymbol{A}$ might lead to misleading results. Therefore, we assume that the observed graph $\boldsymbol{A}$ is obtained by two *latent* factors: $\boldsymbol{A}^c$ representing the corruptions and $\boldsymbol{A}^g$ representing the "good" (clean) graph. More formally, we assume an additive decomposition[2] (see also Fig. 3.1)

$$\boldsymbol{A} = \boldsymbol{A}^g + \boldsymbol{A}^c \text{ with } \boldsymbol{A}^g, \boldsymbol{A}^c \in (\mathbb{R}_{\geq 0})^{n \times n}, \quad \text{both symmetric.} \tag{3.5}$$

Instead of performing the spectral clustering on the corrupted $\boldsymbol{A}$, our goal is to perform it on $\boldsymbol{A}^g$. The key question is how to find the matrices $\boldsymbol{A}^g$ and $\boldsymbol{A}^c$. In particular since clustering is an unsupervised learning task we do not know which entries in $\boldsymbol{A}$ might be wrong. To solve this challenge, we exploit two core ideas.

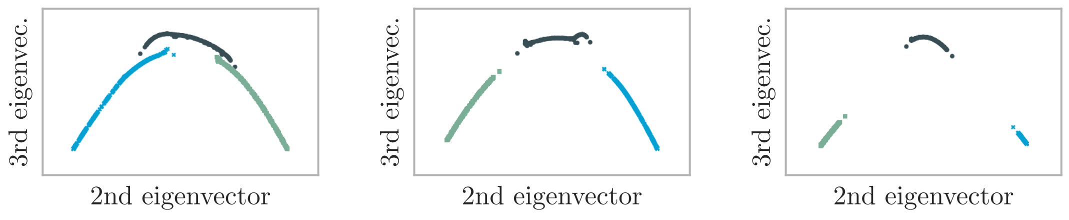
First, we assume that corruptions are relatively rare – if they were not rare, i.e. the majority of the data is corrupted, a reasonable clustering structure can not be expected. Technically, we assume the matrix $\boldsymbol{A}^c$ to be sparse. Let θ denote the maximal number of corruptions a user expects in the data. We require $\|\boldsymbol{A}^c\|_0 \leq 2 \cdot \theta$ where $\|\cdot\|_0$ denotes the element-wise L_0 pseudo-norm and we have $2 \cdot \theta$ due to symmetry of the graph.

While θ constrains the number of corruptions *globally*, it is likewise beneficial to enforce sparsity *locally* per node. This can be realized by the constraint $\|\boldsymbol{a}_i^g\|_0 \geq m$ for each node i, or equivalently $\|\boldsymbol{a}_i^c\|_0 \leq |\mathcal{E}_i| - m$. We chose the first version due to easier interpretability: each node in $\boldsymbol{A}^g$ will be connected to at least m other nodes. Note that θ and m control different effects. To ignore either global or local sparsity, one can simply set the parameter to its extreme value ($\theta = \frac{1}{2}\|\boldsymbol{A}\|_0$ or $m = 1$).

Second, the detection of the latent clean $\boldsymbol{A}^g$ (corrupted $\boldsymbol{A}^c$) is steered by the clustering process, i.e., we *jointly* perform the spectral clustering and the decomposition of $\boldsymbol{A}$. This is in contrast to a sequential process where first the matrix $\boldsymbol{A}^g$ is constructed and then

[1]Surprisingly, many advanced spectral works still consider only the easier case of unnormalized Laplacians. The baseline methods which we compare against [57, 60] handle normalized Laplacians.

[2]This general decomposition not only leads to good performance, as we show later, but also facilitates easy interpretation.



(a) SC, RSC with $\theta = 0$ **(b)** RSC with $\theta = 15$ **(c)** RSC with $\theta = 30$

Figure 3.3: Spectral embeddings for the data of Fig. 3.2b. RSC enhances the discrimination of the clusters as we increase θ. SC (or equivalently RSC with $\theta = 0$) cannot detect the clusters.

the clustering is performed. A strong advantage of a simultaneous detection is that we do not need to specify a separate – often arbitrary – objective for finding $\boldsymbol{A}^g$ since the process is complete determined by the underlying spectral clustering.

We exploit the equivalence of spectral clustering to trace minimization problems (see Sec. 3.2.1, Eq. 3.4). Intuitively, the value of the trace in Eq. 3.4 corresponds to an approximation of the ratio-cut in the graph $\boldsymbol{A}$. The smaller the value, the better the clustering. Thus, we aim to find the matrix $\boldsymbol{A}^g$ by minimizing the trace based on the Laplacian's eigenspace – subject to the sparsity constraints. Concretely, we have:

Problem 3.1. *Given the matrix $\boldsymbol{A}$, the number of clusters k, the sparsity threshold θ, and the minimal number of nearest neighbors m per node. Find $\boldsymbol{H}^* \in \mathbb{R}^{n\times k}$ and $\boldsymbol{A}^{g*} \in (\mathbb{R}_{\geq 0})^{n\times n}$ such that*

$$
\begin{aligned}
(\boldsymbol{H}^*, \boldsymbol{A}^{g*}) = \underset{\boldsymbol{H},\boldsymbol{A}^g}{\arg\min}\, &\mathrm{Tr}(\boldsymbol{H}^T \cdot \boldsymbol{L}(\boldsymbol{A}^g) \cdot \boldsymbol{H}) \\
\textit{subject to} \quad & \boldsymbol{H}^T\boldsymbol{H} = \boldsymbol{I}, \\
& \boldsymbol{A}^g = \boldsymbol{A}^{gT} \\
& \|\boldsymbol{A} - \boldsymbol{A}^g\|_0 \leq 2\cdot\theta \\
& \|\boldsymbol{a}_i^g\|_0 \geq m \quad \forall i \in \{1,\dots,n\}
\end{aligned}
\tag{3.6}
$$

The crucial difference between Eq. 3.4 and Problem 3.1 is that we now *jointly* optimize the spectral embedding $\boldsymbol{H}$ and the similarity graph $\boldsymbol{A}^g$. The Laplacian matrix $\boldsymbol{L}(\boldsymbol{A}^g)$ is *no longer constant* but adaptive. Fig. 3.3 shows the strong advantage of this joint learning. Here, different spectral embeddings $\boldsymbol{H}$ for the data in Fig. 3.2b are shown (second and third eigenvector since the first is constant). Fig. 3.3a shows the embedding using *standard* spectral clustering. Due to the noisy input, the three groups are very close to each other and each spread out. Clustering on this embedding merges multiple groups and, thus, leads to low quality. For real-world data these embeddings look even worse as we will see in the experimental section.

In contrast, Fig. 3.3b and Fig. 3.3c show the spectral embedding learned by our technique when removing just 15 or 30 corrupted edges, respectively. Evidently, the learned embeddings highlight the clustering structure more clearly. Thus, by simultaneously learning the embedding and the corruptions, we improve the clustering quality.

3.4.1 Algorithmic Solution

While our general objective is hard to optimize, in particular due to the $\|.\|_0$ constraints the problem becomes NP-hard in general, we propose a highly efficient block coordinate-descent (alternating) optimization scheme to approximate it. That is, given $\boldsymbol{H}$, we optimize for $\boldsymbol{A}^g/\boldsymbol{A}^c$; and given $\boldsymbol{A}^g/\boldsymbol{A}^c$ we optimize for $\boldsymbol{H}$ (see Algorithm 3.1). Of course, since $\boldsymbol{A}^c$ determines $\boldsymbol{A}^g$ and vice versa, it is sufficient to focus on the update of one, e.g., $\boldsymbol{A}^c$. It is worth pointing out that in many works, the $\|.\|_0$ norm is simply handled by relaxation to the $\|.\|_1$ norm. In our work, in contrast, we aim to preserve the *interpretability* of the $\|.\|_0$ norm. Thus, we derive a connection to the multidimensional Knapsack problem.

Update of $\boldsymbol{H}$. Given $\boldsymbol{A}^c$, the update of $\boldsymbol{H}$ is straightforward. Since $\boldsymbol{A}^g = \boldsymbol{A} - \boldsymbol{A}^c$, $\boldsymbol{L}(\boldsymbol{A}^g)$ is now constant. Therefore, finding $\boldsymbol{H}$ is a standard trace minimization problem (Eq. 3.6). The solution $\boldsymbol{H}^*$ is the k first eigenvectors of $\boldsymbol{L}(\boldsymbol{A}^g)$.

Update of $\boldsymbol{A}^c$. Clearly, since $\boldsymbol{A}^c$ needs to be non-negative, for all elements (i,j) with $\boldsymbol{A}_{ij} = 0$, it also holds $\boldsymbol{A}^c_{ij} = 0$. Thus, in the following, we only have to focus on the elements $\boldsymbol{A}^c_{ij}$ with $(i,j) \in \mathcal{E}$, i.e. the vector $[\boldsymbol{A}^c_e]_{e\in\mathcal{E}}$. We base our update on the following lemma:

Lemma 3.1. *Given $\boldsymbol{H}$, the solution for $\boldsymbol{A}^c$ minimizing Eq. 3.6 can be obtained by* maximizing

$$f_1([\boldsymbol{A}^c_e]_{e\in\mathcal{E}}) := \sum_{(i,j)\in\mathcal{E}} \boldsymbol{A}^c_{ij} \cdot \|\boldsymbol{h}_i - \boldsymbol{h}_j\|_2^2 \tag{3.7}$$

subject to the $\|.\|_0$ constraints and for each e: $\boldsymbol{A}^c_e \in \{0, \boldsymbol{A}_e\}$.

See Sec. B.1 for the proof. Exploiting Lemma 3.1, our problem can equivalently be treated as a set selection problem. Let $\mathcal{X} \subseteq \mathcal{E}$ and $[v^{\mathcal{X}}_e]_{e\in\mathcal{E}} = \boldsymbol{v}^{\mathcal{X}} \in \mathbb{R}^{|\mathcal{E}|}$ be the vector

$$v^{\mathcal{X}}_e = \begin{cases} \boldsymbol{A}_{ij} & \text{if } (i,j) = e \in \mathcal{X} \\ 0 & \text{else} \end{cases} \tag{3.8}$$

our goal is to find a set $\mathcal{X}^* \subseteq \mathcal{E}$ maximizing $f_1(\boldsymbol{v}^{\mathcal{X}^*})$ subject to the constraints. Accordingly, Problem 3.1 can be represented as (a special case of) a multidimensional Knapsack problem [70] operating on the set of edges $\mathcal{E}$.

Corollary 3.1. *Given $\boldsymbol{H}$. Let $\mathcal{X} = \{e \in \mathcal{E} \mid x_e = 1\}$ be the solution of the following multidimensional Knapsack problem: Find $x_e \in \{0,1\}$, $e \in \mathcal{E}$ such that $\sum_{e\in\mathcal{E}} x_e \cdot p_e$ is maximized subject to $\sum_{e\in\mathcal{E}} x_e \leq \theta$ and $\forall i = 1,\dots,n : \sum_{e\in\mathcal{E}_i} x_e \leq |\mathcal{E}_i| - m$ where*

$$p_e = p_{(i,j)} = \boldsymbol{A}_{ij} \cdot \|\boldsymbol{h}_i - \boldsymbol{h}_j\|_2^2 \tag{3.9}$$

The solution for $\boldsymbol{A}^c$ w.r.t. Eq. 3.6 corresponds to $\boldsymbol{v}^{\mathcal{X}}$.

This result matches the intuition for corrupted edges. The term p_e is high for instances whose embeddings are very dissimilar (i.e. they should not belong to the same cluster) but which are still connected by an edge.

Algorithm 3.1 Robust Spectral Clustering

```
Require: Similarity graph $\boldsymbol{A}$, number of clusters $k$, sparsity threshold $\theta$, minimum
    neighbors per node $m$
 1: $\boldsymbol{A}^g \leftarrow \boldsymbol{A}$
 2: while true do
 3:     Compute $\boldsymbol{H}^*$ and Trace based on the eigenspace of $\boldsymbol{L}_*(\boldsymbol{A}^g)$      ▷ update of $\boldsymbol{H}$
 4:     if Trace could not be lowered then break end if
 5:     $\mathcal{X} = \emptyset$
 6:     for each node $i$ do $\text{count}_i \leftarrow |\mathcal{E}_i| - m$ end for
 7:     Initialize priority queue PQ on tuples (score, edge)
 8:     for each edge $e \in \mathcal{E}$ do
 9:         if $p_e > 0$ then add tuple $(p_e, e)$ to PQ end if      ▷ Eq. 3.9 or Eq. 3.11
10:     end for
11:     while PQ not empty do
12:         Get first element from PQ $\rightarrow (., e_{\text{best}} = (i, j))$
13:         if $\text{count}_i > 0 \wedge \text{count}_j > 0$ then
14:             $\mathcal{X} \leftarrow \mathcal{X} \cup \{e_{\text{best}}\}$
15:             $\text{count}_i \mathrel{-}= 1; \text{count}_j \mathrel{-}= 1$
16:             if $|\mathcal{X}| = \theta$ then break end if
17:         end if
18:     end while
19:     Construct $\boldsymbol{A}^c$ according to $\boldsymbol{v}^{\mathcal{X}}$; $\boldsymbol{A}^g = \boldsymbol{A} - \boldsymbol{A}^c$      ▷ Update of $\boldsymbol{A}^c/\boldsymbol{A}^g$
20: end while
21: Apply $k$-means on (normalized) vectors $(\boldsymbol{h}_i)_{i=1,\ldots,n}$
22: return Clustering $\mathcal{C}_1, \ldots, \mathcal{C}_k$
```

While finding the optimal solution of a multidimensional Knapsack problem is intractable, multiple efficient and effective approximate solutions exist [70, 71]. We exploit these approaches for our final algorithm. Following the principle of [71], we first sort the edges $e \in \mathcal{E}$ based on their ratio $p_e/\sqrt{s_e}$. Here, s_e is the number of constraints the variable x_e participates in. Since in our special case, *each* x_e participates in exactly three constraints, $s_e = 3$, it is sufficient to sort the edges based on the value p_e. We then construct a solution by adding one edge after another to $\boldsymbol{A}^c$ as long as the constraints are not violated. This leads to the best possible worst-case bound of $1/\sqrt{n+1}$ [71].

Algorithm 3.1 (lines 5-19) shows the update of $\boldsymbol{A}^c/\boldsymbol{A}^g$. Note that we do not need to sort the full edge set, it is sufficient to iteratively obtain the best edges. Thus, a priority queue PQ (a heap) is used (line 7, 12). The local $\|.\|_0$ constraints can simply be ensured by recording how many edges per node can still be removed (line 6, 15). Thus, an edge can only be included in the result (line 14) if the incident nodes allow to do so (line 13).

The overall method for robust spectral clustering using unnormalized Laplacian iterates between the two update steps (lines 3-19). Note that in each iteration (line 8) considers all edges of the original graph. Thus, an edge marked as corrupted in a previous iteration might be evaluated as non-corrupted later. The algorithm terminates when the trace can not been improved further. In the last step (line 21), the k-means clustering on the improved $\boldsymbol{H}$ matrix is performed as usual.

Complexity. Using a heap, the update of $\boldsymbol{A}^c$ can be computed in time $\mathcal{O}(|\mathcal{E}|+\theta' \cdot \log|\mathcal{E}|)$, where $\theta' \leq |\mathcal{E}|$ is the number of iterations of the inner while loop. Using power iteration, the eigenvectors $\boldsymbol{H}$ can be computed in time linear in the number of edges. Thus, overall, linear runtime can be achieved, as also verified empirically. It is worth mentioning that all operations performed in the algorithm operate on sparse data. This includes the computation of the Laplacian, its eigenvectors, and the constructions of $\boldsymbol{A}^c$ and $\boldsymbol{A}^g$. Thus, even large datasets can easily be handled.

3.5 RSC: Normalized Laplacians

We now tackle the more complex cases of the two normalized Laplacians, which often lead to better clustering. For this, different algorithmic solutions are required.

3.5.1 Random Walk Laplacian

Spectral clustering based on $\boldsymbol{L}_{\text{rw}}$ corresponds to a *generalized* eigenvector problem using $\boldsymbol{L}$ [46]. Our problem definition becomes:

Problem 3.2. *Identical to Problem 3.1 but replacing the constraint $\boldsymbol{H}^T \cdot \boldsymbol{H} = \boldsymbol{I}$ with $\boldsymbol{H}^T \cdot \boldsymbol{D}(\boldsymbol{A}^g) \cdot \boldsymbol{H} = \boldsymbol{I}$.*

Again, our goal is to solve this problem via block-coordinate descent. While the update of $\boldsymbol{H}$ is clear, it corresponds to the first k generalized eigenvectors w.r.t. $\boldsymbol{L}(\boldsymbol{A}^g)$ and $\boldsymbol{D}(\boldsymbol{A}^g)$), using the same approach for $\boldsymbol{A}^c/\boldsymbol{A}^g$ as introduced in Sec. 3.4.1 turns out to be impractical. Since the constraint $\boldsymbol{H}^T \cdot \boldsymbol{D}(\boldsymbol{A}^g) \cdot \boldsymbol{H} = \boldsymbol{I}$ now also depends on $\boldsymbol{A}^g$, we get a highly restrictive constrained problem. As a solution, we propose a principle exploiting the idea of eigenvalue perturbation [39].

Using eigenvalue perturbation, we derive a matrix $\boldsymbol{A}^g$ aiming to minimize the sum of the k smallest *generalized* eigenvalues. Minimizing this sum is equivalent to minimizing the trace based on the normalized Laplacian's eigenspace. We obtain:

Lemma 3.2. *Given the eigenvector matrix $\boldsymbol{H}$ and the corresponding eigenvalues $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_k)$. An approximation of $\boldsymbol{A}^c$ minimizing the objective of Problem 3.2 can be obtained by* maximizing

$$f_2([\boldsymbol{A}^c_e]_{e\in\mathcal{E}}) = \sum_{(i,j)\in\mathcal{E}} \boldsymbol{A}^c_{ij} \underbrace{\left(\|\boldsymbol{h}_i - \boldsymbol{h}_j\|_2^2 - \|\sqrt{\boldsymbol{\lambda}} \circ \boldsymbol{h}_i\|_2^2 - \|\sqrt{\boldsymbol{\lambda}} \circ \boldsymbol{h}_j\|_2^2\right)}_{:=s_{(ij)}} \tag{3.10}$$

subject to the $\|.\|_0$ constraints and for each e: $\boldsymbol{A}^c_e \in \{0, \boldsymbol{A}_e\}$. Here, $\sqrt{.}$ denotes the element-wise square-root, and $\circ$ the Hadamard product.

Clearly, the solution of the unnormalized case (Eq. 3.7) and the normalized case (Eq. 3.10) are structurally very similar – and for solving it we can use the same principle as before (Algorithm 3.1), simply using different edge scores:

$$p_e = p_{(i,j)} = \boldsymbol{A}_{ij} \cdot s_{(ij)} \tag{3.11}$$

Accordingly, also the complexity for finding $\boldsymbol{A}^c$ remains unchanged. Note that only edges with positive score need to be added to the queue (line 8).

Advantages. Comparing Eq. 3.11 with Eq. 3.9 we see an additional "penalty" term which takes the norm/length of the vectors $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ into account. Thereby, instances whose embeddings are far away from the origin get a lower (or even negative) score. Intuitively, the clusters based on spectral embedding are separated by the origin. To see this, recall that in the case of two clusters the final clustering can be obtained by inspecting the sign of the one-dimensional Fiedler vector [46], i.e. the clusters are separated by 0. This is true in general, e.g. see Fig. 3.3 where the origin is in the center of the plots. Since instances that are far away from the origin can be clearly assigned to their cluster, marking their edges as corrupt might improve the clustering only slightly. In contrast, edges that are at the border between different clusters are more challenging, and exactly these edges are preferred by Eq. 3.11. Therefore, the additional penalty term has an overall effect of encouraging a better clustering.

3.5.2 Symmetric Laplacian

We now turn to the last case, spectral clustering using $\boldsymbol{L}_{\text{sym}}$.

Problem 3.3. *Identical to Problem 1 but replacing Eq. 3.6 with*

$$(\boldsymbol{H}^*, \boldsymbol{A}^{g*}) = \operatorname*{arg\,min}_{\boldsymbol{H},\boldsymbol{A}^g} \operatorname{Tr}(\boldsymbol{H}^T \cdot \boldsymbol{L}_{sym}(\boldsymbol{A}^g) \cdot \boldsymbol{H}) \tag{3.12}$$

Using alternating optimization, the matrix $\boldsymbol{H}$ can easily be updated when $\boldsymbol{A}^g$ is given. For updating the matrix $\boldsymbol{A}^g$ (or equivalently $\boldsymbol{A}^c$) we have the following result:

Lemma 3.3. *Given the eigenvector matrix $\boldsymbol{H}$. The matrix $\boldsymbol{A}^c$ minimizing Eq. 3.12 can be obtained by* maximizing

$$f_3([\boldsymbol{A}^c_e]_{e\in\mathcal{E}}) := \sum_{(i,j)\in\mathcal{E}} \frac{\boldsymbol{A}_{ij} - \boldsymbol{A}^c_{ij}}{\sqrt{d_i - d^c_i} \cdot \sqrt{d_j - d^c_j}} \cdot \boldsymbol{h}_i \cdot \boldsymbol{h}^T_j \tag{3.13}$$

subject to the $\|.\|_0$ *constraints and* $0 \leq \boldsymbol{A}^c_e \leq \boldsymbol{A}_e$*, where* $d^c_i = \sum_{e\in\mathcal{E}_i} \boldsymbol{A}^c_e$.

What is the crucial difference between Lemma 3.3 and Lemma 3.1/Lemma 3.2? For the previous solutions, the objective function has decomposed in independent terms. That is, when adding an edge to $\boldsymbol{A}^c$, i.e. changing $\boldsymbol{A}^c_{ij}$ from 0 to $\boldsymbol{A}_{ij}$, the scores of the other edges are not affected. In Lemma 3.3, the sum in f_3 does *not* decompose into independent terms. In particular, the terms d^c_i in the denominator lead to a coupling of multiple edges.

While, in principle, f_3 can be optimized via projected gradient ascent, each gradient step would require to iterate through all edges. Therefore, as an alternative, we propose a more efficient greedy approximation. Similar to before, we focus on the solutions $\boldsymbol{v}^{\mathcal{X}}$.

Starting with $\mathcal{X} = \emptyset$, we iteratively let this set grow following a steepest ascent strategy. That is, we add the edge e_{best} to $\mathcal{X}$ fulfilling

$$e_{\text{best}} = \arg\max_{e \in \mathcal{E}'} f_3(\boldsymbol{v}^{\mathcal{X} \cup \{e\}}) \tag{3.14}$$

where $\mathcal{E}'$ indicates the edges that could be added to $\mathcal{X}$ without violating the constraints. Naively computing Eq. 3.14 requires $|\mathcal{E}'| \cdot |\mathcal{E}|$ many steps – and since we perform multiple iterations to let $\mathcal{X}$ grow, it results in a runtime complexity of $\mathcal{O}(\theta \cdot |\mathcal{E}|^2)$; obviously not practical. In the following, we show how to compute this result more efficiently.

Let $\mathcal{X} \subseteq \mathcal{E}$, $d_i^{\mathcal{X}} := d_i - \sum_{e \in \mathcal{E}_i \cap \mathcal{X}} \boldsymbol{A}_e$, and $p_{ij} := \boldsymbol{A}_{ij} \cdot \boldsymbol{h}_i \cdot \boldsymbol{h}_j^T$. We define

$$s(i, w, \mathcal{X}) := \sum_{\substack{j \\ (i,j) \in \mathcal{E}_i \setminus \mathcal{X} \\ \vee (j,i) \in \mathcal{E}_i \setminus \mathcal{X}}} \left(\frac{1}{\sqrt{d_i^{\mathcal{X}} - w}\sqrt{d_j^{\mathcal{X}}}} - \frac{1}{\sqrt{d_i^{\mathcal{X}}}\sqrt{d_j^{\mathcal{X}}}} \right) p_{ij} \tag{3.15}$$

for each node i, and

$$\delta(e, \mathcal{X}) := \left(\frac{1}{\sqrt{d_i^{\mathcal{X}}}\sqrt{d_j^{\mathcal{X}}}} - \frac{1}{\sqrt{d_i^{\mathcal{X}} - \boldsymbol{A}_e}\sqrt{d_j^{\mathcal{X}}}} - \frac{1}{\sqrt{d_i^{\mathcal{X}}}\sqrt{d_j^{\mathcal{X}} - \boldsymbol{A}_e}} \right) p_{ij} \tag{3.16}$$

for each edge $e = (i, j)$, and $\Delta(e, \mathcal{X}) := s(i, a_e, \mathcal{X}) + s(j, a_e, \mathcal{X}) + \delta(e, \mathcal{X})$.

Corollary 3.2. *Given $\mathcal{X}$ and $\mathcal{E}' \subseteq \mathcal{E} \setminus \mathcal{X}$. It holds*

$$\arg\max_{e \in \mathcal{E}'} f_3(\boldsymbol{v}^{\mathcal{X} \cup \{e\}}) = \arg\max_{e \in \mathcal{E}'} \Delta(e, \mathcal{X})$$

By exploiting Corollary 3.2, we can find the best edge according to Eq. 3.14, by only considering the terms $\Delta(e, \mathcal{X})$. This term can be interpreted as the gain in f_3 when adding the edge e to the set $\mathcal{X}$. After computing the scores $s(i, w, \mathcal{X})$ for each *node*, $\Delta(e, \mathcal{X})$ can be evaluated in constant time per *edge*.

Moreover, let $e = (i, j)$, for each non-incident edge $(i', j') = e' \in \mathcal{E} \setminus (\mathcal{E}_i \cup \mathcal{E}_j)$ it obviously holds $s(i', w, \mathcal{X}) = s(i', w, \mathcal{X} \cup \{e\})$ and $\delta(e', \mathcal{X}) = \delta(e', \mathcal{X} \cup \{e\})$. Thus, assuming the edge $e_{\text{best}} = (i, j)$ has been identified and added to $\mathcal{X}$, for finding the next best edge, only the scores $s(i, ., .)$ and $s(j, ., .)$ need to be updated. We then evaluate δ for all edges incident to the nodes i and j. The remaining nodes and edges are *not* affected since their s, δ, and Δ values are unchanged.

Exploiting these results, we compute the set $\mathcal{X}$ similar to Algorithm 3.1 (lines 5 - 19). Initially, compute for each node i and unique edge weight $\boldsymbol{A}_{ij}$ the term $s(i, \boldsymbol{A}_{ij}, \mathcal{X})$. Then compute for each edge e the term $(\Delta(e, \mathcal{X}), e)$ and add it to the priority queue. These steps can be done in time $\mathcal{O}(\gamma \cdot |\mathcal{E}|)$, where γ is the number of unique edge weights per node. Every time the best element $e_{\text{best}} = (i, j)$ from the PQ is retrieved, we recompute $s(i, ., \mathcal{X})$ and $s(j, ., \mathcal{X})$, followed by a recomputation of $\delta(e, \mathcal{X})$ for all incident edges. Noticing that there are at most $2x$ many incident edges (x nearest-neighbor graph) these steps can be done in time $\mathcal{O}(\gamma \cdot x + x \cdot \log(|\mathcal{E}|))$.

Overall, this leads to a time complexity of $\mathcal{O}(\gamma \cdot |\mathcal{E}| + \theta \cdot (x \cdot \log(|\mathcal{E}|) + \gamma \cdot x))$. Note that the worst case (each edge has a unique weight) corresponds to $\gamma = x$. In this case we obtain $\mathcal{O}(x \cdot |\mathcal{E}| + \theta \cdot (x \cdot \log(|\mathcal{E}|) + x^2))$. For our case of spectral clustering using nearest-neighbor graphs, however, it holds $\gamma = 1$. In this case, we obtain an algorithm with complexity $\mathcal{O}(|\mathcal{E}| + \theta \cdot x \cdot \log(|\mathcal{E}|))$ Thus, being linear in the number of edges.

In summary, the principle for solving Eq. 3.12 is almost identical to Algorithm 3.1 with the additional overhead of re-evaluating the term $\Delta(e, \mathcal{X})$ for the edges incident to e_{best}.

3.6 Local Purity and Global Separation

Our main hypothesis is that jointly learning the embedding and the corruptions improves the embedding quality. How can we measure the quality? Instead of relying on an arbitrary technique for clustering the embedding we derive statistics that depend on the embedding itself and the ground-truth classes[3]. We argue that two properties should be fulfilled:

- **Local Purity**: In a good embedding the instances within every local neighborhood should mostly belong to the same class.
- **Global Separation**: In a good embedding it should be possible to distinguish between instances of different classes by inspecting the intra-class and inter-class distances only. That is, the classes should be easily separable.

In Fig. 3.4 we illustrate different embeddings which different quality according to the proposed metrics. A good embedding is both locally pure and globally separated (Fig. 3.4a). In Sec. 3.7 we show that the embedding produced by RSC satisfies both of these criteria.

3.6.1 Local Purity

Let $\boldsymbol{h}_i$ denote the embedding of instance i and $c_i \in \mathcal{C}$ its class according to the ground truth. Intuitively, we define the purity $\text{pur}_x(i)$ of the neighborhood around instance i given its x-nearest neighbors as the largest fraction of instances belonging to the same class. Formally, let $\mathcal{E}_x(i)$ denote the set of x nearest neighbors of i in the embedding space. Then the purity for node i is given by

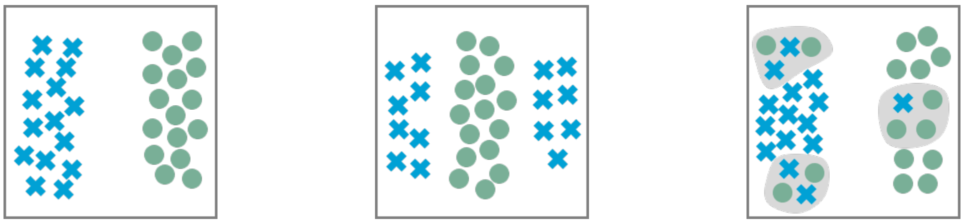
$$\text{pur}_x(i) = \frac{1}{x+1} \max_{c \in \mathcal{C}} \underbrace{\left|\{j \in (\mathcal{E}_x(i) \cup \{i\}) \mid c_j = c\}\right|}_{\text{frequency of class } c \text{ in } \mathcal{E}_x(i)} \tag{3.17}$$

i.e. equals the ratio of the majority class in the neighborhood of node i (including the node itself). The overall local purity (at scale x) is defined as the average over all instances:

$$\text{PUR}_x = \frac{1}{N} \sum_{i=1}^{N} \text{pur}_x(i) \tag{3.18}$$

In the best case $\text{PUR}_x = 1$ and each neighborhood contains instances of a single class only, and in the worst case $\text{PUR}_x = 1/|\mathcal{C}|$ the classes are uniform.

[3]We specifically use the term “class” to indicate the groups given by the ground truth, not the groups detected by an arbitrary clustering method applied on the embedding.



(a) Both pure and separated **(b)** Pure but not separated **(c)** Separated but not pure

Figure 3.4: Illustration of the proposed local purity and global separation metrics for the quality of an embedding. A good embedding is both locally pure and globally separated as in (a).

3.6.2 Global Separation

We formalize global separation by extending the idea of the Silhouette coefficient [72]. For each class c we compute the list of pairwise distances $P_{c,c}$ of all instances within the class, and the list of pairwise distances $P_{c,c'}$ between instances from class c and $c' \neq c$

$$P_{c,c'} = [\mathrm{dist}(\boldsymbol{h}_i, \boldsymbol{h}_j)]_{i \in \mathcal{C}_c, j \in \mathcal{C}_{c'}} \tag{3.19}$$

where $\mathcal{C}_c = \{i \mid c_i = c\}$ is the set of all instances from class c.

For each list we compute the mean over the x% smallest elements, denoted as $P_{c,c'}(x)$, i.e. we compute the truncated mean discarding the top $(1-x)$% of the elements. Following the Silhouette coefficient, we then compute the difference between the *within class* distances and the distance to the *closest other class* c^*

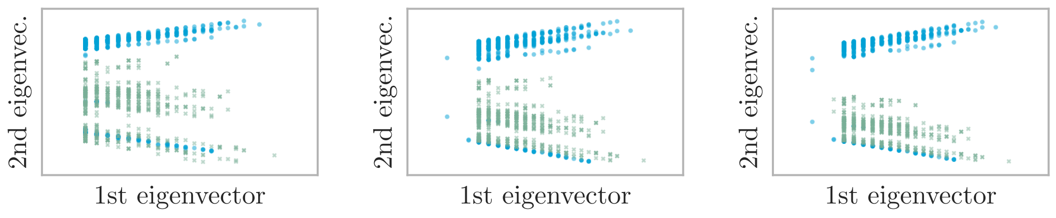
$$\mathrm{GS}_c(x) = \frac{P_{c,c^*}(x) - P_{c,c}(x)}{\max\{P_{c,c^*}(x), P_{c,c}(x)\}}, \qquad c^* = \arg\min_{c' \neq c} P_{c,c'}(x) \tag{3.20}$$

In the best case $\mathrm{GS}_c(x) = 1$, in the worst case -1. $\mathrm{GS}_c(x)$ can intuitively be seen as a robust extension of the Silhouette coefficient w.r.t. the ground-truth classes. For $x = 1$ it resembles the Silhouette coefficient w.r.t. class c. For $x < 1$ only partial distances are considered capturing that the embedding might not exactly represent the ground truth.

3.7 Experiments

Setup. We compare our method (RSC) against standard spectral clustering (SC), and the two related works AHK [60] and NRSC [57]. We denote with RSC-$\mathrm{L}_{\mathrm{xyz}}$ the different variants of our method using the corresponding Laplacian. For all techniques, we set the number of clusters k equal to the number of clusters in the data.

As default values we construct nearest-neighbor graphs with $q = 15$ neighbors, allowing half of the edges to be removed per node ($m = 0.5 \cdot q$). While Li et al. [57] use a principle for automatically setting their parameters the obtained performance was often below average, so we manually optimized their parameters to obtain better solutions. All experiments are averaged over several k-means runs for stability. We evaluate the clustering quality of the different approaches using normalized mutual information (NMI) with 1 being the best.



(a) SC, RSC with $\theta = 0$ **(b)** RSC with $\theta = 10$ **(c)** RSC with $\theta = 20$

Figure 3.5: Spectral embedding of the Banknote data based on $\boldsymbol{L}_{\text{sym}}$. The learned embedding increase the discrimination between the two clusters (denoted with different colors/markers).

Real-world data. We use three handwritten digits datasets: Pendigits [73] ($n = 7494$), USPS [74] ($n = 9298$), and two subsamples of the MNIST dataset [75] ($n = 20K$) because the baselines have complexity cubic in the number of instances. We also use the Banknote authentication ($n = 1372$), and Iris ($n = 150$) dataset [73]. See Sec. A.1 for details.

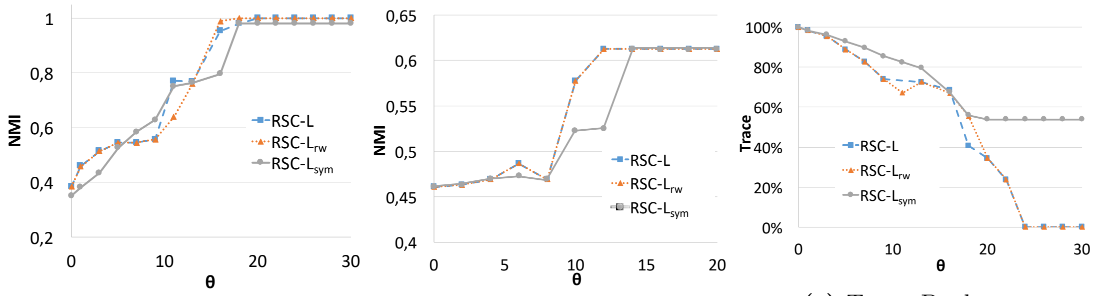
Synthetic data. Besides the moon data (Fig. 3.2) which is perturbed with Gaussian noise using different variance σ, we also generate synthetic similarity graphs based on the planted partitions model [76]. Given the clusters we randomly connect each node to p percent of the other nodes in its cluster. Additionally, we add a fraction of noisy edges to the graph. By default we generate data with 1000 instances, $p = 0.3$ and 20 clusters.

3.7.1 Evaluating the Spectral Embedding

We start by analyzing the spectral embeddings obtained by RSC. In Fig. 3.3 we illustrated the spectral embeddings for the data of Fig. 3.2b. Standard spectral clustering fails on this data, since the embedding does not separate the clusters. In contrast, applying our technique, we obtain the embeddings as shown in Fig. 3.3c where the three clusters are clearly separated which means that we can perfectly remove the ground-truth clusters.

A similar behavior can be observed for real world data. Fig. 3.5 shows the spectral embedding of the Banknote data regarding $\boldsymbol{L}_{\text{sym}}$ (the other Laplacians show similar results). On Fig. 3.5a we see the original SC embedding. The points do not show a clear separation in two groups. In the middle and right plot, we applied RSC with $\theta = 10$ and $\theta = 20$, respectively. As shown, the separation between the points clearly increases. The embedding is optimized leading to a higher clustering accuracy. As we will see later, for the Banknote data, the NMI score increases from 0.46 for $\theta = 0$ to 0.61 for $\theta = 20$.

Sparsity threshold. As indicated in Fig. 3.5, increasing the sparsity threshold might lead to a clearer separation. We now analyze this aspect in more detail. Fig. 3.6a analyzes the Two Moons dataset with Gaussian noise of $\sigma = 0.1$. We vary θ for all three techniques. $\theta = 0$ corresponds to original spectral clustering using the corresponding Laplacian. Clearly, its quality is low. As shown, for all techniques we observe an increase in the clustering quality until a stable point is reached. Fig. 3.6b shows the same behavior for the Banknote data. The removal of corrupted edges improves the clustering results. All three variants are able to reach the highest NMI of 0.61.



(a) NMI, Two Moons **(b)** NMI, Banknote **(c)** Trace, Banknote

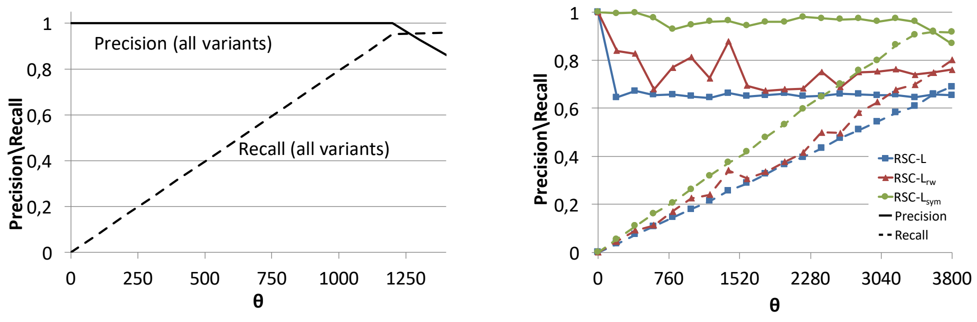
Figure 3.6: Increasing the sparsity threshold θ improves the clustering quality (increases the NMI score) and decreases the value of the trace (relative to the trace obtained by SC, $\theta = 0$).

Remark: Using the variant based on $\boldsymbol{L}$, at some point, the quality will surely drop again. When all corrupted edges have been removed, one will start to remove "good" edges. The reason is that the terms in Eq. 3.7 are always non-negative. In contrast, using $\boldsymbol{L}_{rw}/\boldsymbol{L}_{\text{sym}}$ (Eq. 3.7, Corollary 3.2, Δ), edges connecting points within the same cluster will often obtain negative scores. Those edges will *never* be included in the matrix $\boldsymbol{A}^c$, independent of θ. Thus, in general, the later two versions are more robust regarding θ.

According to our problem description (e.g. Problem 3.1) we aim to minimize the trace. In Fig. 3.6c we illustrate the value of the trace for the setting of Fig. 3.6b. Since the trace between the different Laplacians can not be meaningfully compared in absolute values, we plot it relative to the trace obtained by standard spectral clustering. For all of our approaches the trace can successfully be lowered by a significant amount, which confirms the effectiveness of our approach. Our algorithms often need only around 10 iterations to converge to these good results.

3.7.2 Detecting Corrupted Edges

Next, we analyze how well our approach is able to spot corrupted edges. For this, we artificially added corrupted edges to the similarity graph based on the planted partition



(a) 10% noise **(b)** 20% noise

Figure 3.7: RSC achieves high precision and recall. Corrupted edges are successfully detected.

model. We investigate two different settings: in one case 10% of all edges in the graph are corrupted; in the other even 20% of all edges. Knowing the corrupted edges, we measure the precision $p = |A \cap B|/|B|$ and recall $r = |A \cap B|/|A|$, where A denotes the corrupted edges, and B the edges removed by our technique.

Fig. 3.7 shows the results when increasing the number of removed edges (θ). For the 10% noise case (Fig. 3.7a), we observe a very high precision which stays at the optimal value until $\theta = 1200$ – only the corrupted edges are removed. Note that the absolute number of corrupted edges in the data is 1261. Likewise, the recall is continuously increasing until ≈ 0.96 – only a few corrupted edges could not be detected. The scenario with 20% noise (3605 corrupted edges in total) is more challenging. While $\boldsymbol{L}_{\text{sym}}$ obtains a result very close to optimal, $\boldsymbol{L}_{\text{rw}}$ and $\boldsymbol{L}$ perform slightly worse. For these variants also some 'good' edges get removed. Note that the curves do not need to be monotonic. Due to the joint optimization, different edges can be removed for each parameter setting. Overall, for realistic noise levels all techniques perform well – with $\boldsymbol{L}_{\text{sym}}$ often being the best.

3.7.3 Evaluating Robustness

Next we study how increasing the noise affects the clustering quality. Specifically, we randomly add Gaussian noise to the Two Moons data with variance σ increased from 0 to 0.14. To highlight the variation in the clustering quality we average the results over ten seeds for each noise parameter. Fig. 3.8 shows the results. The lines represents the mean NMI, while the error bands represent the 95% confidence interval. Note that for standard SC we report the best result among all three Laplacian for each dataset individually. Thus, standard spectral clustering gets an additional strong benefit. Clearly, spectral clustering is not robust and rapidly decreases in quality. $\boldsymbol{L}_{\text{sym}}$ performs best and all of our approaches clearly outperform the baseline.

Next, we analyze the robustness of the methods by artificially adding noise to real data. To ensure that the cluster detection is indeed getting more difficult, we specifically add corrupted edges to the similarity graph connecting different clusters. The results for the Banknote data are presented in Fig. 3.9. As shown, at the beginning all techniques

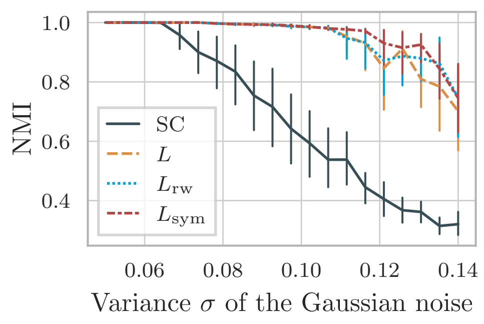


Figure 3.8: Robustness to noise. Our RSC clearly outperforms spectral clustering.

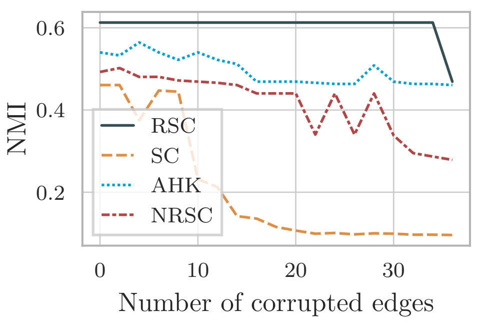


Figure 3.9: Robustness of all techniques on Banknote. RSC is most stable.

remain at their quality level obtained on the original data, with RSC obtaining the highest quality. Adding more corruptions, however, standard spectral clustering drops very quickly and sharply to low quality. In contrast, RSC stays at its highest level for the longest time. AHK is quite stable as well, while NRSC is much more sensitive.

3.7.4 Cluster Quality Comparison

Next, we provide an overview of the clustering quality. For all techniques we used the symmetric normalized Laplacian since it performed best. Even though our main aim is to improve spectral clustering approaches, we additionally report the results of two other popular clustering techniques: k-means and mean shift (density-based). For the later, we tuned the bandwidth parameter to obtain highest scores. As already mentioned in the setup, the competing techniques' parameters were tuned as well. For RSC, we simply used a very large θ and let the method automatically decide how many edges to mark as corrupted – one advantage of $\boldsymbol{L}_{\text{sym}}$. Table 3.1 summarizes the results for the different datasets. Besides using the full datasets, we use the principle of [57, 60] and additionally select subsets of the data. More precisely, from the Pendigits data we select specific digits indicated with Pendigits-xyz.

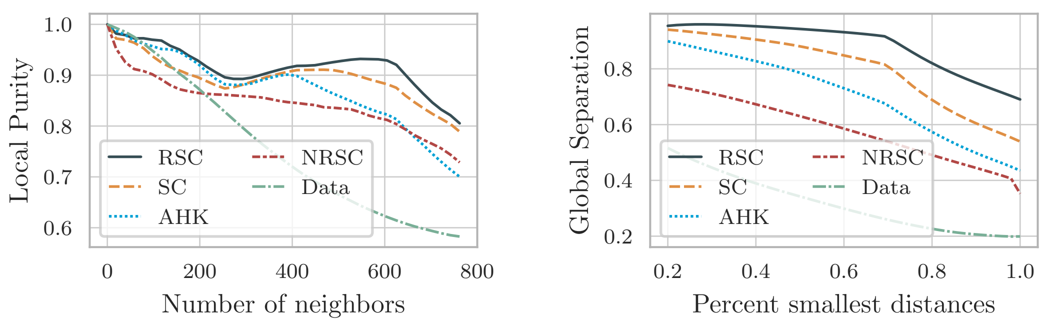
RSC significantly outperforms the baselines. On some datasets we even see a 15% improvement w.r.t. spectral clustering. Though, it is also fair to mention that not for all datasets an improvement can be achieved. Our method clearly outperforms k-means and density-based clustering and it finishes for all these datasets in a few seconds to minutes. In contrast, NRSC and AHK required already around one and three hours respectively on the larger MNIST data.

Table 3.1: Comparison of clustering performance (NMI) for different techniques and datasets.

Data	SC	NRSC	AHK	k-means	mean-shift	RSC
Moons	0.47	0.99	0.53	0.19	0.34	**1.00**
Banknote	0.46	0.47	0.52	0.03	0.03	**0.61**
USPS	0.78	0.83	0.77	0.61	0.15	**0.85**
MNIST-10 K	0.71	0.70	0.70	0.48	0.48	**0.73**
MNIST-20 K	0.70	0.76	0.71	0.48	d.n.f.	**0.78**
Iris	0.78	0.79	0.53	0.76	0.72	**0.80**
Pendigits	0.82	**0.83**	0.82	0.69	0.66	0.82
Pendigits-16	0.86	0.87	0.88	0.88	0.01	**0.91**
Pendigits-146	0.93	0.94	0.94	0.88	0.47	**0.96**

3.7.5 Local Purity and Global Separation

Local purity. Fig. 3.10a shows the results for the Banknote data for all competing techniques. The number of neighbors (i.e. x in PUR_x) is scaled from 1 to the maximal



(a) Local Purity   **(b)** Global Separation

Figure 3.10: Evaluation of local purity and global separation on the Banknote dataset.

cluster size to ensure that all scales are captured. As a baseline we also evaluate the purity of the original input data (i.e. the embedding space is the raw input data). The original data only has good purity for small x, but drops quickly. This indicates complex shapes like in Fig. 3.2. For banknote, RSC has consistently the highest local purity. The embedding well reflects the ground truth locally. SC and AHK perform slightly worse. NRSC, in contrast, drops quicker similar to the baseline. We see similar qualitative results for the other datasets as well.

Global separation. Fig. 3.10b shows the result for the Banknote data. We plot the average global separation among the two classes (i.e. average of $\mathrm{GS}_1(x)$ and $\mathrm{GS}_2(x)$). Again, the raw data shows the worst result, with scores consistently below 0.4 indicating no good separation/clusteredness of the class labels in the space. In contrast, RSC obtains high scores up to a large-scale x. That is, a very large fraction of the instances are well separated and clustered in the learned embedding. The competing approaches consistently perform worse, showing no good match between the ground truth and the clusteredness of the embedding. The results also indicate that *local purity* and *global separation* of an embedding are indeed two different properties. The learned embeddings of RSC capture well both properties confirming the benefit of our joint learning principle.

3.8 Conclusion

We proposed a robust spectral clustering technique for noisy data. Our core idea was to decompose the similarity graph into two latent factors: sparse corruptions and clean data. We jointly learned the spectral embedding as well as the corrupted data. We proposed three different algorithmic solutions using different Laplacians. Our experiments have shown that the learned embeddings clearly emphasize the clustering structure and that our method outperforms standard spectral clustering and other state-of-the-art competitors. In Chapter 4 we discuss a different notion of robustness for node embeddings. Namely, rather than explicitly removing noisy edges we embed nodes as Gaussian distributions which implicitly capture the uncertainty resulting from the corruptions.

3.9 Retrospective

In Gu et al. [77] the authors study the unsupervised link selection problem which they see as the network equivalent of the traditional feature selection problem. In short, their goal is to select a subset of informative edges to enhance the quality of the community structure. As it turns out, the resulting optimization problem is equivalent to our Problem 3.1 for the standard Laplacian (without local budget constraints). Unsurprisingly, given the widely diverging vocabulary (feature selection vs. node clustering) we did not identify this paper as part of the relevant work during the initial literature search. Nonetheless, we briefly discuss some relevant aspects for completeness.

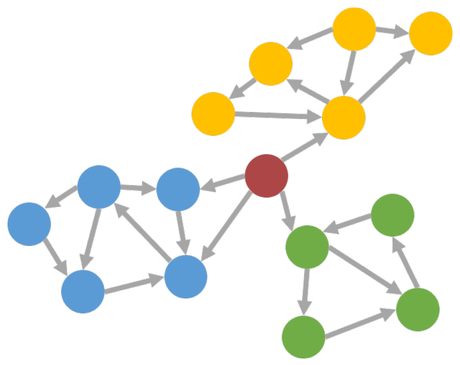
First, it is interesting to highlight that the approach they took to approximately solve the same problem differs from ours. Similar to our approach they propose an iterative algorithm: they compute the impact of removing each edge based on first-order partial derivatives and iteratively remove the edge with the highest score as in Eq. 3.9. Specifically, they update the graph with the single highest scoring edge and recompute the eigenvectors, repeating the procedure until the budget is exhausted. Compare this to our iterative Algorithm 3.1 where we select the top scoring edges at once based on the current estimate, and only then we recompute the embeddings (eigenvectors). Both algorithms perform similar in practice. It is also worth mentioning that our alternative derivation based on the connection to the multidimensional Knapsack problem gives us the best possible worst-case bound on the approximation error of our algorithm.

A noteworthy observation which we omitted in the original paper is that if we relax the binary constrains w.r.t. the selected edges, by replacing the $\|\cdot\|_0$ with the $\|\cdot\|_1$ in Eq. 3.6, the problem can be exactly solved as a semi-definite program (SDP) [77, 78]. This holds in general for optimizing any convex function of the graph Laplacian's eigenvalues [79].

Given that the eigenvalues for $\boldsymbol{L}_{\text{rw}}$ and $\boldsymbol{L}_{\text{sym}}$ are the same, i.e. λ is an eigenvalue of $\boldsymbol{L}_{\text{rw}}$ with eigenvector $\boldsymbol{u}$ if and only if λ is an eigenvalue of $\boldsymbol{L}_{\text{sym}}$ with eigenvector $\boldsymbol{w} = \boldsymbol{D}^{1/2}\boldsymbol{u}$, an alternative approach to solving Problem 3.3 is to first solve Problem 3.2 obtaining $\boldsymbol{H}^*$ and then simply return $\boldsymbol{D}^{1/2}\boldsymbol{H}^*$. Note that the solutions for the two alternatives would be the same if we can solve the problems exactly, however, since our approximations behave differently for the different problems the solutions might also be different. Analogously, we can first solve Problem 3.3 and then multiply by $\boldsymbol{D}^{-1/2}$ to obtain an alternative solution for Problem 3.2.

An interesting extension of RSC would be to select the edges based on the non-backtracking matrix (NB) of the graph rather then the Laplacian. In Krzakala et al. [80] the authors show that the spectrum of this matrix (related to the non-backtracking random walk) is better behaved and maintains a strong separation between the bulk eigenvalues and the eigenvalues relevant for community structure. Moreover, spectral clustering based on the NB matrix is optimal for graphs generated by the stochastic block model and detects clusters down to the theoretical limit. Our approach using eigenvalue perturbation theory can be adapted to approximate the eigenvalues of the NB matrix.

4 Gaussian Node Embeddings


(a) A simple graph with three main clusters marked by different colors.

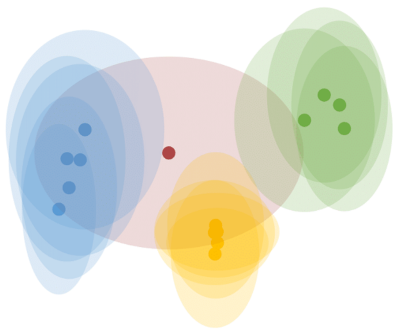

(b) Each node is represented as a two-dimensional Gaussian distribution.

Figure 4.1: Graph2Gauss embeds nodes as distributions thereby capturing uncertainty.

4.1 Introduction

Node embeddings are a powerful and increasingly popular approach for analyzing graph data [28]. By operating in the embedding space one can employ proved learning techniques and bypass the difficulty of incorporating complex node interactions. Beyond clustering (as in Chapter 3), other tasks such as link prediction, node classification, and graph visualization, all greatly benefit from learning node representations. For attributed graphs by leveraging both sources of information (network structure and attributes) we can learn more useful representations [81–83] compared to approaches that only consider the graph, such as the (robust) spectral embedding. Here we assume that the graph is explicitly given, although the approach can also be applied to similarity graphs as with RSC.

All existing (attributed) graph embedding approaches represent each node by a single point in a low-dimensional continuous vector space. Representing the nodes simply as points, however, has a crucial limitation: we do not have information about the uncertainty of that representation. Yet uncertainty is inherent when describing a node in a complex graph by a single point only. Imagine a node for which the different sources of information (attributes vs. graph structure) are conflicting with each other, e.g. pointing to different communities. Such discrepancy, and any other noise in the data, should be reflected in the uncertainty of its embedding. As a solution to this problem, we introduce an embedding approach that represents nodes as *Gaussian distributions*: each node is represented with an entire distribution rather than a single point, thereby capturing uncertainty.

To effectively deal with the non-i.i.d. nature of the data arising from the complex interactions between the nodes we propose an unsupervised *personalized ranking* formulation to learn the embeddings. Intuitively, from the point of view of a single node, we want nodes in its immediate neighborhood to be closest in the embedding space, while nodes multiple hops away should become increasingly more distant. Our ranking formulation arises by enforcing the distance between node embeddings to be proportional to the geodesic distance between the nodes. Taking into account this natural ranking from each node's point of view, we learn more powerful embeddings since we incorporate information about the network structure beyond first and second-order proximity.

Furthermore, when node attributes (e.g. text) are available our method is able to leverage them to easily generate embeddings for previously unseen nodes without additional training. In other words, our proposed approach Graph2Gauss is inductive, which is a significant benefit over existing methods that are inherently transductive and do not generalize to unseen nodes easily. This desirable inductive property is due to our learned encoder that maps node attributes to embeddings.

The main contributions of our approach are summarized as follows:

- We embed nodes as Gaussian distributions allowing us to capture **uncertainty**.
- Our unsupervised **personalized ranking** formulation exploits the natural ordering of the nodes capturing the network structure at multiple scales.
- We propose an **inductive** method that generalizes to unseen nodes and is applicable to different types of graphs: plain/attributed, directed/undirected.

4.2 Related Work

We focus on unsupervised learning of node embeddings for which many different approaches have been proposed. For a comprehensive recent survey see Cai et al. [28], Hamilton et al. [84], or Goyal and Ferrara [85]. Approaches such as DeepWalk and node2vec [86, 87] look at plain graphs and learn an embedding based on random walks by extending or adapting the Skip-Gram [88] architecture. LINE [89] uses first- and second-order proximity and trains the embedding via negative sampling. SDNE [90] similarly has a component that preserves second-order proximity and exploits first-order proximity to refine the representations. GraRep [91] is a factorization-based method that considers local and global structural information.

Tri-Party Deep Network Representation (TRIDNR) [82] considers node attributes, network structure and potentially node labels. CENE [92] similarly to Paper2Vec [83] treats the attributes as special kinds of nodes and learns embeddings on the augmented network. Text-Associated DeepWalk (TADW) [81] performs low-rank matrix factorization considering graph structure and text features. Heterogeneous networks are consider in [93, 94], while Huang et al. [95] similarly to Pan et al. [82] additionally consider labels. GraphSAGE [96] is a (partially) inductive method that generates embeddings by sampling and aggregating attributes from a node's local neighborhood. Unlike Graph2Gauss, to compute the embeddings GraphSAGE requires edge information at test/inference time.

Graph convolutional networks are another family of approaches that adapt conventional CNNs to graph data [19, 97–100]. They utilize the graph Laplacian and the spectral definition of a convolution and boil down to some form of aggregation over neighbors such as averaging. They can be thought of as implicitly learning an embedding, e.g. by taking the output of the last layer before the supervised component. See Monti et al. [98] for an overview. In contrast to Graph2Gauss, most of these methods are (semi-)supervised. The graph variational autoencoder (GAE) [101] is a notable exception that learns node embeddings in an unsupervised manner.

Few approaches consider the idea of learning an embedding that is a distribution. Vilnis and McCallum [102] are the first to learn Gaussian word embeddings to capture uncertainty. Closest to our work, He et al. [103] represent knowledge graphs and Dos Santos et al. [104] study heterogeneous graphs for node classification. Both approaches are not applicable for the context of unsupervised learning of (attributed) graphs that we are interested in. The method in He et al. [103] learns an embedding for each component of the triplets (head, tail, relation) in the knowledge graph. Note that we cannot naively employ this method by considering a single relation "has an edge" and a single entity "node". Since their approach considers similarity between entities and relations, all nodes would be trivially similar to the single relation. Considering the semi-supervised approach proposed in Dos Santos et al. [104] we cannot simply "turn off" the supervised component to adapt their method for unsupervised learning, since given the defined loss we would trivially map all nodes to the same Gaussian distribution. Additionally, both of these approaches do not consider node attributes.

4.3 Deep Gaussian Embedding

In this section we introduce our method Graph2Gauss (G2G) and detail how both the attributes and the network structure influence the learning of node representations. The embedding is carried out in two steps: (i) the node attributes are passed through a non-linear transformation via a deep neural network (encoder) and yield the parameters associated with the node's embedding distribution; (ii) we formulate an unsupervised loss function that incorporates the natural ranking of the nodes as given by the network structure w.r.t. a dissimilarity measure on the embedding distributions.

Problem definition. Let $G = (\boldsymbol{A}, \boldsymbol{X})$ be a directed attributed graph, where $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is an adjacency matrix representing the edges between N nodes and $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ collects the attribute information for each node where $\boldsymbol{X}_i$ is a D-dimensional attribute vector of the i^{th} node[1]. We aim to find a lower-dimensional Gaussian distribution embedding $\boldsymbol{h}_i = \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, $\boldsymbol{\mu}_i \in \mathbb{R}^L, \boldsymbol{\Sigma}_i \in \mathbb{R}^{L \times L}$ with $L \ll N, D$ such that nodes similar w.r.t. attributes and network structure are also similar in the embedding space given a dissimilarity measure $\Delta(\boldsymbol{h}_i, \boldsymbol{h}_j)$. In Fig. 4.1 for example we show nodes that are embedded as two-dimensional Gaussian distributions.

[1]Note, in the absence of node attributes we can simply use one-hot encoding for the nodes (i.e. $\boldsymbol{X} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix) and/or any other derived features such as node degrees.

4.3.1 Structure Representation via Personalized Ranking

To capture the structural information of the network in the embedding space, we propose a personalized ranking approach. That is, locally per node i we impose a ranking of all remaining nodes w.r.t. their distance to node i in the embedding space. More precisely, we exploit the k-hop neighborhoods of each node. Given some anchor node i, we define

$$\mathcal{V}_{ik} = \{j \in \mathcal{V} \mid i \neq j, \min(\mathrm{sp}(i,j), K) = k\} \tag{4.1}$$

to be the set of nodes who are exactly k hops away from node i (see Fig. 4.2). Here $\mathcal{V}$ is the set of all nodes, K is a hyperparameter denoting the maximum distance we are wiling to consider, and $\mathrm{sp}(i,j)$ returns either the length of the shortest path starting at node i and ending in node j or ∞ if node j is not reachable from node i.

Intuitively, we want all nodes belonging to the 1-hop neighborhood of i to be closer to i w.r.t. their embedding, compared to the all nodes in its 2-hop neighborhood, which in turn are closer than the nodes in its 3-hop neighborhood and so on up to K. Thus, the ranking that we want to ensure from the perspective of node i is

$$\Delta(\boldsymbol{h}_i, \boldsymbol{h}_{k_1}) < \Delta(\boldsymbol{h}_i, \boldsymbol{h}_{k_2}) < \cdots < \Delta(\boldsymbol{h}_i, \boldsymbol{h}_{k_K}) \quad k_1 \in \mathcal{V}_{i1}, k_2 \in \mathcal{V}_{i2}, \ldots, k_K \in \mathcal{V}_{iK} \tag{4.2}$$

or equivalently, we aim to satisfy the following pairwise constraints

$$\Delta(\boldsymbol{h}_i, \boldsymbol{h}_j) < \Delta(\boldsymbol{h}_i, \boldsymbol{h}_{j'}), \qquad i \in \mathcal{V}, j \in \mathcal{V}_{ik}, j' \in \mathcal{V}_{ik'}, k < k' \tag{4.3}$$

Going beyond mere first-order and second-order proximity this enables us to capture the network structure at multiple scales incorporating local and global structure.

Dissimilarity measure. To solve the above ranking task we have to define a suitable dissimilarity measure between the latent representation of two nodes. Since our latent representations are distributions, similarly to Dos Santos et al. [104] and He et al. [103] we employ the asymmetric KL divergence. This gives the additional benefit of handling directed graphs in a sound way. More specifically, given the latent Gaussian distribution representation of two nodes $\boldsymbol{h}_i, \boldsymbol{h}_j$ we define

$$\begin{aligned}\Delta(\boldsymbol{h}_i, \boldsymbol{h}_j) &= D_{\mathrm{KL}}\big(\mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \mid\mid \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big) \\ &= \frac{1}{2}\left[\mathrm{Tr}(\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\Sigma}_j) + (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) - L - \log\frac{\det(\boldsymbol{\Sigma}_j)}{\det(\boldsymbol{\Sigma}_i)}\right]\end{aligned} \tag{4.4}$$

Here we use the notation $\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$ to denote the outputs of some functions $\mu_\theta(\boldsymbol{X}_i)$ and $\Sigma_\theta(\boldsymbol{X}_i)$ applied to the attributes $\boldsymbol{X}_i$ of node i and $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix. The asymmetric KL divergence also applies to the case of an undirected graph by simply processing both directions of the edge. We could alternatively use a symmetric dissimilarity measure such as the Jensen-Shannon divergence, or the expected likelihood (probability product kernel). See Sec. 4.6 for additional insights on the KL divergence.

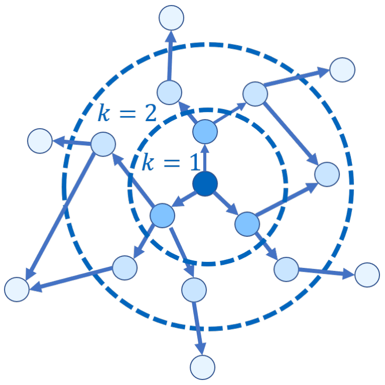
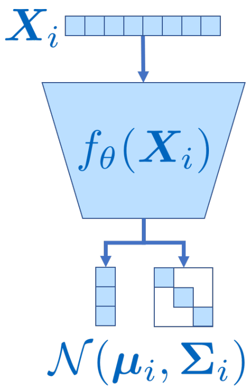


Figure 4.2: K-hop neighborhood.



$\boldsymbol{X}_i$
$f_\theta(\boldsymbol{X}_i)$
$\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$

Figure 4.3: Parametrized Encoder.

4.3.2 Deep Encoder

The functions $\mu_\theta(\boldsymbol{X}_i)$ and $\Sigma_\theta(\boldsymbol{X}_i)$ are deep feed-forward non-linear neural networks parametrized by θ. It is important to note that these parameters are shared across instances and thus enjoy statistical strength benefits. Additionally, we design $\mu_\theta(\boldsymbol{X}_i)$ and $\Sigma_\theta(\boldsymbol{X}_i)$ such that they share parameters as well. More specifically, a deep encoder $f_\theta(\boldsymbol{X}_i)$ processes the node's attributes and outputs an intermediate hidden representation, which is then in turn used to output $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ in the final layer of the architecture (see Fig. 4.3). We focus on diagonal covariance matrices.[2] The mapping from the nodes' attributes to their embedding via the deep encoder is precisely what makes Graph2Gauss inductive.

4.3.3 Energy-based Loss

Since it is intractable to find a solution that satisfies all of the pairwise constraints defined in Sec. 4.3.1 we turn to an energy-based learning approach. The idea is to define an objective function that penalizes ranking errors given the energy of the pairs. More specifically, denoting the KL divergence between two nodes as the respective energy, $E_{ij} = D_{\mathrm{KL}}\big(\mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \mid\mid \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big)$, we define the following loss to be optimized

$$\mathcal{L} = \sum_i \sum_{k<k'} \sum_{j \in \mathcal{V}_{ik}} \sum_{j' \in \mathcal{V}_{ik'}} \left({E_{ij}}^2 + \exp^{-E_{ij'}} \right) = \sum_{(i,j,j') \in \mathcal{D}_t} \left({E_{ij}}^2 + \exp^{-E_{ij'}} \right) \tag{4.5}$$

where $\mathcal{D}_t = \{(i, j, j') \mid \mathrm{sp}(i, j) < \mathrm{sp}(i, j')\}$ is the set of all valid triplets. The E_{ij} terms are positive examples whose energy should be lower compared to the energy of the negative examples $E_{ij'}$. Here, we employed the so called square-exponential loss [105] which unlike other typically used losses (e.g. hinge loss) does not have a fixed margin and pushes the energy of the negative terms to infinity with exponentially decreasing force. For a given anchor node i, the energy E_{ij} should be lowest for nodes j in its 1-hop neighborhood, followed by a higher energy for nodes in its 2-hop neighborhood, etc.

[2]To ensure that the diagonal covariance matrix is positive-definite in the final layer we output $\tilde{\sigma}_{id} \in \mathbb{R}$ and obtain $\sigma_{id} = \mathrm{elu}(\tilde{\sigma}_{id}) + 1$.

We can optimize the parameters θ of the deep encoder such that the loss $\mathcal{L}$ is minimized and the pairwise rankings are satisfied. Note again that the parameters are shared across all instances, meaning that we share statistical strength and can learn them more easily in comparison to treating the distribution parameters ($\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$) independently as free variables. We optimize the parameters using Adam [106] with a fixed learning rate.

Sampling strategy. For large graphs, the complete loss is intractable to compute imposing the need for a stochastic variant. The naive approach would be to sample triplets from $\mathcal{D}_t$ uniformly, i.e. replace $\sum_{(i,j,j')\in\mathcal{D}_t}$ with $\mathbb{E}_{(i,j,j')\sim\mathcal{D}_t}$ in Eq. 4.5. However, with the naive sampling we are less likely to sample triplets that involve low-degree nodes since high-degree nodes occur in many more pairwise constraints. This in turn means that we update the embedding of low-degree nodes less often which is not desirable.

Therefore, we propose an alternative node-anchored sampling strategy. Intuitively, for every node i, we randomly sample one other node from *each* of its neighborhoods (1-hop, 2-hop, etc.) and then optimize over all the corresponding pairwise constraints $(E_{i1} < E_{i2}, \ldots, E_{i1} < E_{iK}, E_{i2} < E_{i3}, \ldots E_{i2} < E_{iK}, \ldots, E_{iK-1} < E_{iK})$.

Naively applying the node-anchored sampling strategy and optimizing Eq. 4.5, however, would lead to biased estimates of the gradient. Proposition 4.1 shows how to adapt the loss such that it is equal in expectation to the original loss under our new sampling strategy. As a consequence, we have unbiased estimates of the gradient using stochastic optimization of the reformulated loss.

Proposition 4.1. *For all i, let $(j_1, \ldots, j_K)$ be independent uniform random samples from the sets $(\mathcal{V}_{i1}, \ldots, \mathcal{V}_{iK})$ and $|\mathcal{V}_{i*}|$ the cardinality of each set. Then $\mathcal{L}$ equals*

$$\mathcal{L}_s := \sum_i \mathbb{E}_{(j_1,\ldots,j_K)\sim(\mathcal{V}_{i1},\ldots,\mathcal{V}_{iK})}\left[\sum_{k<k'} |\mathcal{V}_{ik}| \cdot |\mathcal{V}_{ik'}| \cdot \left(E_{ij_k}^2 + \exp^{-E_{ij_{k'}}}\right)\right] = \mathcal{L} \tag{4.6}$$

See proof in Sec. C.1. For cases where the number of nodes N is particularly large we can further subsample mini-batches, by selecting anchor nodes i uniformly at random. Furthermore, in our experimental study, we analyze the effect of the sampling strategy on convergence, as well as the quality of the stochastic variant w.r.t. the obtained solution and the reached local optima.

4.3.4 Discussion

Inductive learning. While during learning we need both the network structure (to evaluate the ranking loss) and the attributes, once the learning concludes the embedding for a node can be obtained solely based on its attributes. This enables our method to easily handle the issue of obtaining representations for new nodes that were not part of the network during training. To do so we simply pass the attributes of the new node through our learned deep encoder. Most approaches cannot handle this issue at all, with a notable exception being SDNE and GraphSAGE [90, 96]. However, both approaches require the edges of the new node to obtain the node's representation, and cannot handle nodes that have no existing connections. In contrast, our method can handle even such nodes, since after the model is learned we rely only on the attribute information.

Plain graph embedding. Even though attributed graphs are often found in the real world, sometimes it is desirable to analyze plain graphs. As already discussed, our method easily handles plain graphs, when the attributes are not available, by using e.g. one-hot encoding of the nodes instead. As we later show in the experiments we are able to learn useful representations in this scenario, even outperforming some attributed approaches. Naturally, in this case we lose the inductive ability to handle unseen nodes. We compare the one-hot encoding version, termed G2G-OH, with our full method G2G that utilizes the attributes, as well as all remaining competitors.

Encoder architecture. Depending on the type of the node attributes (e.g. images, text) we could in principle use CNNs/RNNs to process them. We could also easily incorporate any of the proposed graph convolutional layers inheriting their benefits. However, we observe that in practice using simple feed-forward architecture with ReLU is sufficient, while being much faster and easier to train. Better yet, we observed that Graph2Gauss is not sensitive to the choice of hyperparameters such as number and size of hidden layers. We provide more details and sensible defaults in Sec. C.2.

Complexity. The time complexity for computing the original loss is $O(N^3)$ where N is the number of nodes. Using our node-anchored sampling strategy, the complexity of the stochastic version is $O(K^2N)$ where K is the maximum distance considered. Since a small value of $K \leq 2$ consistently showed good performance, K^2 becomes negligible and thus the complexity is $O(N)$, meaning linear in the number of nodes. This coupled with the small number of epochs T needed for convergence ($T \leq 2000$ for all shown experiments, see Fig. 4.6b) and an efficient GPU implementation also made our method faster than most competitors in terms of wall-clock time.

4.4 Embedding Evaluation

Setup. We compare Graph2Gauss with and without using attributes (G2G, G2G-OH) to several competitors namely: TRIDNR [82] and TADW [81] as representatives that consider attributed graphs, GAE [101] as the unsupervised graph convolutional representative, and node2vec [87] as a representative of the random-walk-based plain graph embeddings. Additionally, we include a strong Logistic Regression baseline that considers only the attributes. As with all other methods we train TRIDNR in a unsupervised manner, however, since it can only process raw text as attributes (rather than e.g. bag-of-words) it is not always applicable. Since TADW, and GAE only support undirected graphs we must symmetrize the graph before using them – giving them a substantial advantage, especially in the link prediction task.

In all experiments if the competing techniques use an L-dimensional embedding, G2G's embedding is actually only *half* of this dimensionality so that the overall number of 'parameters' per node (mean vector + variance terms of the diagonal $\mathbf{\Sigma}_i$) matches L.

Datasets. We use several attributed graph datasets. Cora [107] is a well-known citation network labeled based on the paper topic. While most approaches report on a small subset of this dataset we additionally extract from the original data the entire network and name these two datasets Cora ($N = 19793, E = 65311, D = 8710, K = 70$) and

Cora-ML ($N = 2995, E = 8416, D = 2879, K = 7$) respectively. Furthermore, we evaluate on three other commonly used benchmark citation datasets: Citeseer ($N = 4230, E = 5358, D = 2701, K = 6$) [108], DBLP [82] ($N = 17716, E = 105734, D = 1639, K = 4$) and PubMed ($N = 18230, E = 79612, D = 500, K = 3$) [29]. See Sec. A.1.1 for more details about the datasets and Sec. A.2 for information about the Graph2Gauss code.

4.4.1 Link Prediction

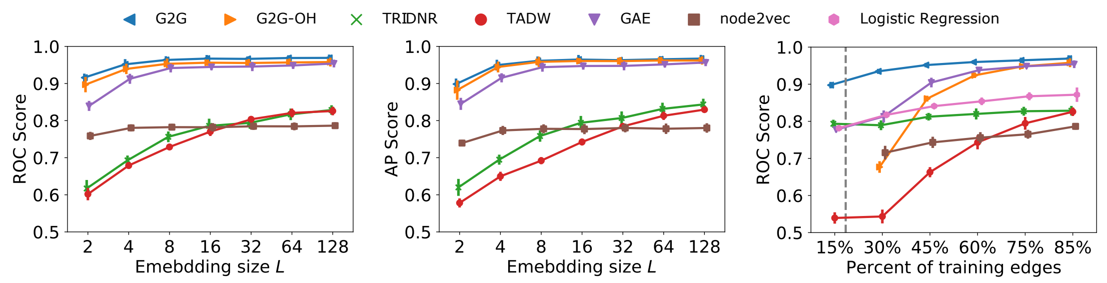
Setup. Link prediction is a commonly used task to evaluate embeddings. Given an observed graph we hide a random set of edges/non-edges and train on the resulting graph. As in previous work [90, 101], we create a validation/test set that contains 5%/10% randomly selected edges respectively and equal number of randomly selected non-edges. We used the validation set for hyperparameter tuning and early stopping and the test set only to report the performance. We report the area under the ROC curve (AUC) and the average precision (AP) scores for each method. To rank the candidate edges we use the negative energy $-E_{ij}$ for Graph2Gauss, and the exact approach reported in the respective papers for the baseline methods (e.g. dot product of the embeddings).

Performance on real-world datasets. Table 4.1 shows the performance on the link prediction task for different datasets and embedding size $L = 128$. As we can see our method significantly outperforms the competitors across all datasets which is a strong sign that the learned embeddings are useful. Furthermore, even the constrained version of our method G2G-OH that does not consider attributes at all outperforms the competitors on some datasets. While GAE achieves comparable performance on some of the datasets their approach does not scale to large graphs. In fact, for graphs beyond $15K$ nodes we had to revert to slow training on the CPU since the data did not fit on the GPU memory (12GB). The simple Logistic Regression baseline showed surprisingly strong performance, even outperforming some of the more complicated methods.

We also include the performance on the so called "Cora-ML Easy" dataset, obtained from the Cora-ML dataset by making it undirected and selecting only the nodes in the largest connected component. We see that while node2vec struggles on the original real-world data, it significantly improves in this "easy" setting. On the contrary, Graph2Gauss handles both settings effortlessly. This demonstrates that Graph2Gauss can be readily applied in realistic scenarios on potentially messy real-world data.

Table 4.1: Link prediction performance for real-world datasets with $L = 128$.

Method	**Cora-ML**		**Cora**		**Citeseer**		**DBLP**		**Pubmed**		**Cora-ML Easy**	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Logistic Regression	90.01	89.75	86.58	86.51	81.70	79.10	82.04	81.91	90.50	90.99	90.28	90.99
node2vec [87]	76.80	75.26	79.95	78.98	83.04	83.74	95.42	95.33	95.42	95.33	93.47	93.53
TADW [81]	81.26	81.34	76.56	78.06	70.14	72.93	65.67	59.85	62.72	68.02	83.53	82.47
TRIDNR [82]	84.51	85.69	81.61	81.08	87.23	88.87	92.01	91.62	NTA	NTA	85.59	86.16
GAE [101]	96.65	96.67	97.91	98.07	92.31	93.88	95.78	96.67	96.07	96.12	95.97	95.17
G2G-OH	96.95	97.54	98.41	98.63	95.89	95.78	98.29	98.46	96.75	96.47	96.98	96.42
G2G	**98.01**	**98.03**	**98.81**	**98.78**	**96.09**	**96.16**	**98.65**	**98.78**	**97.42**	**97.85**	**98.03**	**98.12**



(a) L vs. AUC **(b)** L vs. AP **(c)** $\%E$ vs. AUC

Figure 4.4: Link prediction performance for different embedding sizes (L) and of percentages of training edges ($\%E$) on Cora-ML. G2G consistently outperforms the baselines.

Sensitivity analysis. In Fig. 4.4a and Fig. 4.4b we show the performance w.r.t. the dimensionality of the embedding, averaged over 10 trials. G2G is able to learn useful embeddings with strong performance even for relatively small embedding sizes. Even for the case $L = 2$, where we embed the points as one-dimensional Gaussian distributions ($L = 1 + 1$ for the mean and the sigma of the Gaussian), G2G still outperforms all of the competitors irrespective of their much higher embedding sizes.

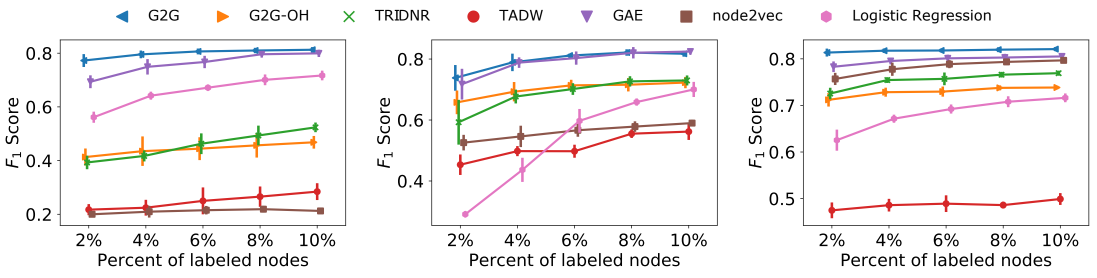
Finally, we evaluate the performance w.r.t. the percentage of training edges varying from 15% to 85%, averaged over 10 trials. We can see in Fig. 4.4c Graph2Gauss strongly outperforms the competitors, especially for small number of training edges. The dashed line indicates the percentage above which we can guarantee to have every node appear at least once in the training set.[3] The performance below that line is then indicative of the performance in the inductive setting. Since, the structure-only methods are unable to compute meaningful embeddings for unseen nodes we cannot report their performance below the dashed line.

4.4.2 Node Classification

Setup. Node classification is another task commonly used to evaluate the quality of the learned embeddings. We evaluate the node classification performance for three datasets (Cora-ML, Citeseer and DBLP) that have ground-truth classes. First, we train the embeddings on the entire training data in an *unsupervised* manner excluding the class labels. Then, following Perozzi et al. [86] we use varying percentage of randomly selected nodes and their learned embeddings along with their labels as training data for a (L_2-regularized) logistic regression. Finally, we evaluate the performance on the remaining nodes. We also optimize the regularization strength for each method/dataset via cross-validation. We show results averaged over 10 trials.

Performance on real-world datasets. Fig. 4.5 compares the methods w.r.t. the classification performance for different percentage of labeled nodes. We can see that our method clearly outperforms the baselines. Again, the constrained version of our method that does not consider attributes is able to outperform some of the competing approaches.

[3]This percentage is derived from the size of the minimum edge-cover set. For more details see Sec. C.2.



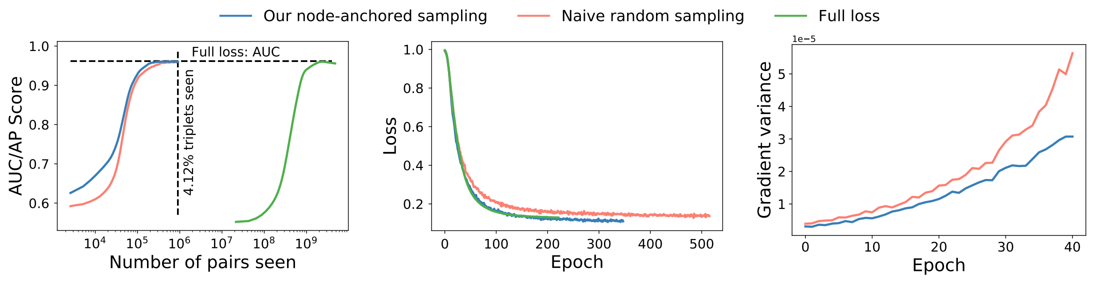
(a) Citeseer **(b)** Cora **(c)** DBLP

Figure 4.5: Classification performance comparison – both G2G and G2G-OH perform strongly.

We see that Graph2Gauss shows stable performance regardless of the percentage of labeled nodes. This means that it is sufficient to train using a small number of labeled nodes. It is highly desirable since labels are usually expensive to obtain.

4.4.3 Sampling Strategy

Fig. 4.6a shows the validation set ROC score for the link prediction task w.r.t. the number of triplets (i, j_k, j_l) seen. We can see that both sampling strategies are able to reach the same performance as the full loss in significantly fewer $(< 4.2\%)$ number of pairs seen (note the logarithmic scale). It also shows that the naive random sampling converges slower than the node-anchored sampling strategy. Fig. 4.6b gives us some insight as to why – our node-anchored sampling strategy achieves significantly lower loss. Finally, Fig. 4.6c shows that the gradients of our node-anchored sampling strategy have lower variance which is another reason for faster convergence.

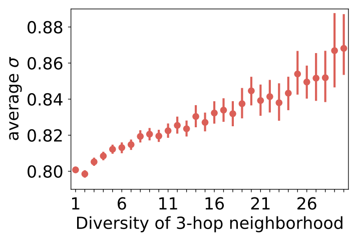


(a) Convergence **(b)** Loss comparison **(c)** Gradient variance

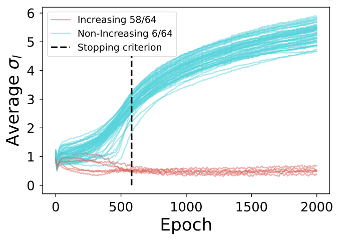
Figure 4.6: Our sampling strategy converges significantly faster than the full loss, while maintaining good performance. It also has lower loss and lower variance compared to naive sampling.

4.4.4 Embedding Uncertainty

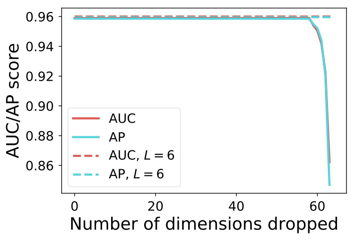
Learning an embedding that is a distribution rather than a point-vector allows us to capture uncertainty about the representation. We perform several experiments to



(a) Neighborhood diversity



(b) Latent dimensionality



(c) Dropping dimensions

Figure 4.7: The benefit of modeling the uncertainty of the nodes.

evaluate the benefit of modeling uncertainty. Fig. 4.7a shows that the learned uncertainty is correlated with neighborhood diversity, where for a node i we define diversity as the number of distinct classes among the nodes in its K-hop neighborhood ($\bigcup_{1 \le k \le K} \mathcal{V}_{ik}$). Since the uncertainty for a node i is an L-dimensional vector (diagonal covariance) we show the average across the dimensions. In line with our intuition, nodes with less diverse neighborhood have significantly lower variance compare to more diverse nodes whose immediate neighbors belong to many different classes, thus making their embedding more uncertain. Fig. 4.7 shows the result on the Cora dataset for $K = 3$-hop neighborhood. Similar results hold for the other datasets. This result is particularly impressive given the fact that we learn our embedding in a completely unsupervised manner, yet the uncertainty was able to capture the diversity w.r.t. the class labels of the neighbors of a node which were never seen during training.

Fig. 4.7b shows that using the learned uncertainty we are able to detect the intrinsic latent dimensionality of the graph. Each line represents the average variance over all nodes (y-axis) for a given dimension l for each epoch (x-axis). We can see that as the training progresses past the stopping criterion (link prediction performance on the validation set) and we start to overfit, some dimensions exhibit a relatively stable average variance, while for others the variance increases with each epoch. By creating a simply rule that monitors the average change of the variance over time we were able to automatically detect these relevant latent dimensions (colored in red). This result holds for multiple datasets and is shown here for Cora-ML. Interestingly, the number of detected latent dimensions (6) is close to the number of ground-truth communities (7).

The next obvious question is then how does the performance change if we remove these highly uncertain dimensions whose variance keeps increasing with training. Fig. 4.7c answers exactly that. By removing progressively more and more dimensions, starting with the most uncertain (on average) first we see imperceptibly small change in performance. Only once we start removing the true latent dimension we see a noticeable degradation in performance. The dashed lines show the performance if we re-train the model, setting $L = 6$, equal to the detected number of latent dimensions.

As a last study of uncertainty, in a use-case analysis, the nodes with high uncertainty reveal additional interesting patterns. For example in the Cora dataset, one of the highly uncertain nodes was the paper "The Use of Word Shape Information for Cursive Script

Table 4.2: Inductive link prediction performance for different datasets.

Method (% hidden)	**Cora-ML**		**Cora**		**Citeseer**		**DBLP**		**Pubmed**	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Logistic Regression (10%)	75.95	78.62	78.53	78.70	73.09	72.54	67.55	69.55	86.83	87.34
G2G (10%)	90.93	89.37	94.18	93.40	88.58	88.31	85.06	83.75	92.22	90.45
G2G (25%)	87.83	86.31	92.96	92.31	87.30	86.61	83.09	81.49	90.20	88.28

Recognition" – surprisingly, all citations (edges) of that paper (as extracted from the dataset) were towards other papers by one of the coauthors, R.J. Whitrow.

4.4.5 Generalization to Unseen Nodes

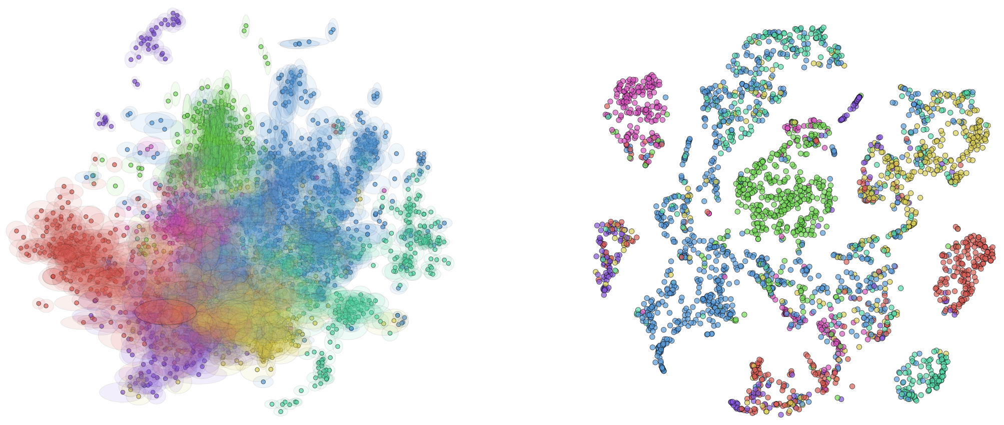
As discussed in Sec. 4.3.4 G2G is able to learn embeddings even for nodes that were not part of the networks structure during training time. Thus, it not only supports transductive but also inductive learning. To evaluate how our approach generalizes to unseen nodes we perform the following experiment: (i) first we completely hide 10%/25% of nodes from the network at random; (ii) we proceed to learn the node embeddings for the rest of the nodes; (iii) after learning is complete we pass the (new) unseen test nodes through our deep encoder to obtain their embedding; (iv) we evaluate by calculating the link prediction performance (AUC and AP scores) using all their edges and same number of randomly sampled non-edges.

As the results in Table 4.2 clearly show, since we are utilizing the rich attribute information, we are able to achieve strong performance for unseen nodes. This is true even when a quarter of the nodes are missing. This makes our method applicable in the context of large graphs where training on the entire network is not feasible. Note that SDNE [90] and GraphSAGE [96] cannot be applied in this scenario, since they also require the edges for the unseen nodes to produce an embedding. Graph2Gauss is the only inductive method that can obtain embeddings based *only* on the node attributes.

4.4.6 Network Visualization

One key application of node embedding approaches is creating meaningful visualizations of a network in 2D/3D that support tasks such as data exploration and understanding. Following Tang et al. [89] and Pan et al. [82] we first learn a lower-dimensional $L = 128$ embedding for each node and then map those representations in 2D with TSNE [109].

Additionally, since our method is able to learn useful representations even in low dimensions we embed the nodes as 2D Gaussian distributions and visualize the resulting embedding. This has the added benefit of visualizing the nodes' uncertainty as well. Fig. 4.8 shows the visualization for the Cora-ML dataset. We see that Graph2Gauss learns an embedding in which the different classes are clearly separated.



(a) G2G, $L = 2 + 2 = 4$ **(b)** G2G, $L = 128$, projected with TSNE

Figure 4.8: 2D visualization of the embeddings on the Cora-ML dataset. Color indicates the class label not used during training. Best viewed on screen.

4.5 Conclusion

We proposed Graph2Gauss – the first unsupervised approach that represents nodes in attributed graphs as Gaussian distributions and is therefore able to capture uncertainty. Analyzing the uncertainty reveals the latent dimensionality of a graph and gives insight into the neighborhood diversity of a node. Since we exploit the attribute information of the nodes we can effortlessly generalize to unseen nodes, enabling inductive reasoning. Graph2Gauss leverages the natural ordering of the nodes w.r.t. their neighborhoods via a personalized ranking formulation. The strength of the learned embeddings has been demonstrated on several tasks – specifically achieving high link prediction performance even in the case of low-dimensional embeddings.

4.6 Retrospective

Since we only learn a mapping from features to embeddings, one limitation of Graph2Gauss is that if two nodes have the same input features they will also have the same output embeddings regardless of their neighborhood structure and position in the graph. An easy way to tackle this limitation is to add additional features that distinguish between the nodes, e.g. one-hot encoding of the node IDs. However, doing so the model will no longer be inductive, which might be an acceptable trade-off depending on the application.

After publishing this work we made several observations. First, when learning node embeddings which are used in downstream node classification or link prediction tasks, the direction of the KL divergence did not matter in practice. That is training with either $\Delta(\boldsymbol{h}_i, \boldsymbol{h}_j)$ or $\Delta(\boldsymbol{h}_j, \boldsymbol{h}_i)$ for a directed edge $i \to j$ resulted in similar performance. The direction, however, still matters if we want to capture tree-like structures and entailment and whether nodes closer to the "root" should have lower or higher variance (see also [102]). Second, limiting the number of hops to two was often enough to achieve state-of-the-art results for most datasets, and using $k > 2$ hops yielded marginal (or no) improvements, despite higher-order proximity being one selling point of the method. Considering that the benchmark datasets exhibit strong homophily this is not surprising. Finally, for plain graphs, in the absence of node attributes we originally suggested to simply use one-hot encoding for the nodes (i.e. $\boldsymbol{X} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix). In subsequent experiments we observed that using the adjacency matrix $\boldsymbol{A}$ or the graph Laplacian plus the identity matrix ($\boldsymbol{X} = \boldsymbol{L} + \boldsymbol{I}$) works significantly better. Again, this is not surprising since the graph Laplacian encodes useful information about the graph.

In our other work [8, 10] we discovered that a predict then propagate scheme tends to outperform the message-passing schemes typically used in GNNs. Given an input graph, in the predict phase we generate predictions for each node individually using only that node's own features. In the propagate phase the individual logits are diffused with PageRank to incorporate the graph information in a single non-recursive step. Furthermore, in Klicpera et al. [10] we show that during inference we do not need to have access to the structure information. As long we use the PageRank vectors to diffuse information during training, simply predicting the individual logits during inference without propagation showed almost no loss in performance. Interestingly, this is analogous to Graph2Gauss where we similarly learn a mapping from features to embeddings and we utilize the graph structure only during training (except unsupervised). This points to a more general phenomenon. Another way to view this is that the graph serves as a regularizer during training.

Given that personalized PageRank vectors seem to capture the right information for many tasks (beyond the above example they have also been used to directly learn embeddings [110]) an interesting extension of Graph2Gauss would be to consider a ranking loss w.r.t. the PageRank scores rather than the node distances.

In this chapter we mostly discussed the benefits of Graph2Gauss for improving the clean performance. Zhu et al. [111] show evidence that Gaussian embeddings can also significantly improve the robustness of GNNs to adversarial attacks. They show that the effects of the adversarial perturbations can be absorbed in the variances of the Gaussian distribution. To remedy the effect of perturbation propagation which we discussed in the introduction, they further propose a variance-based attention mechanism.

We can embed data with hierarchical (tree-like) structure in low-dimensional hyperbolic spaces with low distortion [112]. Since real graphs are often hierarchical, and since Gaussian distributions are directly related to hyperbolic embeddings via the Fisher distance, which defines the same local geometry as the KL divergence [113], it is not surprising that G2G achieves excellent performance in low dimensions (Fig. 4.4a).

We also highlight an alternative sampling scheme for Eq. 4.5 proposed by Lienen [114]. They rewrite the loss such that we can obtain unbiased gradients by sampling edges from a complete k-partite graph which potentially allows for a more fine-grained control over the accuracy-performance trade-off.

5 Adversarial Attacks on Node Embeddings

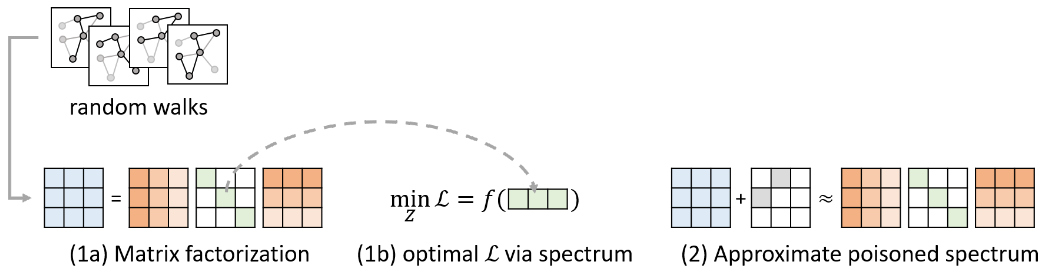

random walks
$\min_{Z} \mathcal{L} = f(\ )$
(1a) Matrix factorization
(1b) optimal $\mathcal{L}$ via spectrum
(2) Approximate poisoned spectrum

Figure 5.1: Overview of our adversarial attack. Given a set of random walks as input we express the optimal loss via the graph spectrum and then approximate the poisoned graph's spectrum.

5.1 Introduction

As we show in Chapter 3 and Chapter 4 graph representation learning is an effective approach for simultaneously tackling many downstream tasks such as link prediction, node classification, community detection, and visualization. To recap, the goal is to embed each node in a low-dimensional feature space such that the relevant information from the graph structure (and the node features if they are available) is captured. Among the variety of proposed approaches, techniques based on random walks (RWs) [86, 87] are often used in practice since they incorporate higher-order relational information.

Given the increasing popularity of these methods there is a strong need to analyse their robustness. In particular, we study the existence and effects of *adversarial* perturbations. This is critical, since especially in domains where graph embeddings are often used, such as the web, adversaries are common and false data is easy to inject. For example, spammers can easily create fake accounts on social networks. A large body of research [43, 115–120] shows that both traditional and deep learning methods can be attacked: even slight deliberate perturbations of the data can lead to wrong results. While adversarial attacks for graph models have been recently proposed [121–123], they are all limited to the semi-supervised learning setting.

Can we construct attacks that do no rely on a specific downstream task? Are node embedding methods just as easily fooled, since compared to semi-supervised models they do not incorporate a supervision signal that we can exploit? Answering these questions positively, this is the first work on adversarial perturbations for unsupervised embeddings.

In the previous chapters we took on the role of a defender. We assumed that the data is noisy or corrupted and we developed robust embeddings. Here we adopt the of role of an attacker – we poison the graph structure to asses the vulnerability of node embeddings to adversarial perturbations. Interestingly, the core mathematical tool which we rely on, eigenvalue perturbation theory (see Sec. 2.3), is the same in both cases.

Barring the few aforementioned attacks on graphs most existing adversarial attacks perturb the features of individual instances. In our case however, since we are operating on plain graph data (no features are available) we perturb the interactions (edges) between instances instead. Manipulating the network structure (the graph) is a common scenario in practice, with link spam farms [42] and Sybil attacks [124] as typical examples.

Since node embeddings are typically trained in an unsupervised and transductive fashion we cannot rely on a single end-task that our attack might exploit to find appropriate perturbations, and we have to handle a challenging poisoning attack where the model is learned *after* the attack. That is, the model cannot be assumed to be static as in most existing attacks. Lastly, since graphs are discrete, gradient-based approaches [115, 125] for finding adversarial perturbations that were designed for continuous data are not well suited. In particular, for methods based on random walks the gradient computation is not directly possible since sampling random walks is not a differentiable operation. The question is how to design efficient algorithms that are able to find adversarial perturbations in such a challenging – discrete and combinatorial – graph domain?

We propose a principled strategy to efficiently solve a challenging bi-level optimization problem associated with the poisoning attack by exploiting results from eigenvalue perturbation theory [39]. We assume an attacker with full knowledge about the data and the model, thus ensuring reliable vulnerability analysis in the worst case. Nonetheless, our experiments on transferability demonstrate that our strategy generalizes – attacks learned based on one model successfully fool other models as well. We study both general and targeted attacks, as well as the effect of restricting the attacker.

Overall, we shed light on an important problem that has not been studied so far. We show that node embeddings are sensitive to adversarial attacks. Relatively few changes are needed to significantly damage the quality of the embeddings even in the scenario where the attacker is restricted. Furthermore, our insights highlight that more work is needed to make node embeddings robust to adversarial perturbations and thus readily applicable in production systems.

5.2 Related Work

We focus on unsupervised node embedding approaches based on random walks (RWs) and further show how one can easily apply a similar analysis to attack spectral-based node embeddings. For a recent survey, also of other non-RW-based approaches, we refer to Cai et al. [28]. Moreover, while many semi-supervised learning methods [10, 19, 126] have been introduced, we focus on unsupervised methods since they are often used in practice due to their flexibility in simultaneously solving various downstream tasks.

Adversarial attacks. Attacking machine learning models has a long history, with seminal works on SVMs and logistic regression [15, 115]. Neural networks were also shown to be highly sensitive to small adversarial perturbations to the input [43, 127]. While most works focus on image classification, recent works also study adversarial examples in other domains [117, 128]. Different taxonomies exist characterizing the adversaries based on their goals, knowledge, and capabilities [129–131]. See Sec. 2.5 for more details.

The two dominant attacks types are poisoning attacks targeting the training data (the model is trained after the attack) and evasion attacks targeting the test data/application phase (the learned model is assumed fixed). Compared to evasion attacks, poisoning attacks are far less studied [115, 120, 125, 131, 132] since they usually require solving a challenging bi-level optimization problem.

Attacks on semi-supervised graph models. The robustness of semi-supervised graph classification methods to adversarial attacks has recently been analyzed [121–123]. The first work, introduced by Zügner et al. [121], linearizes a graph convolutional network (GCN) [19] to derive a closed-form expression for the change in class probabilities for a given edge/feature perturbation. They calculate a score for each possible edge flip based on the classification margin and greedily pick the top edge flips with highest scores. Later, Dai et al. [122] proposed a reinforcement (Q-)learning formulation where they decompose the selection of relevant edge flips into selecting the two end-points. Finally, Zügner and Günnemann [123] develop a general attack on the training procedure of a GCN model using meta-gradients. All three approaches focus on the semi-supervised graph classification task and take advantage of the supervision signal to construct the attacks. In contrast, our work focuses on general attacks on *unsupervised* node embeddings applicable to many downstream tasks.

Manipulating graphs. There is an extensive literature on optimizing the graph structure to manipulate: information spread in a network [133, 134], user opinions [135, 136], shortest paths [137, 138], page rank scores [139], and other metrics [140]. In the context of graph clustering, Chen et al. [141] measure the performance changes when injecting noise to a bipartite graph of DNS queries, but do not focus on automatically generating attacks. Zhao et al. [142] study poisoning attacks on multi-task relationship learning, although they exploit relations between tasks, they still deal with the classic scenario of i.i.d. instances within each task.

Robustness and adversarial training. Robustification of machine learning models, including graph-based models [1, 143], has been studied and is known as adversarial/robust machine learning. These approaches are out of the scope for this thesis. Adversarial training, e.g. via GANs [144], is similarly beyond our scope since the goal is to improve the embeddings, while our goal is to asses the vulnerability of existing embedding methods to adversarial perturbations.

5.3 Attacking Node Embeddings

We study poisoning attacks on the graph structure – the attacker is capable of adding or removing (flipping) edges in the original graph within a given budget. We focus mainly on approaches based on random walks and extend the analysis to spectral approaches (see Sec. D.2). All proofs are deferred to Sec. D.1.

5.3.1 Background and Preliminaries

Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected unweighted graph where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges, and $\boldsymbol{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the adjacency matrix. The goal of network representation learning is to find a low-dimensional embedding $\boldsymbol{z}_v \in \mathbb{R}^K$ for each node with $K \ll |\mathcal{V}|$. This dense low-dimensional representation should preserve information about the network structure – nodes similar in the original network should be close in the embedding space. DeepWalk [86] and node2vec [87] learn an embedding based on RWs by adapting the skip-gram architecture [88] for learning word embeddings. They sample finite (biased) RWs and use the co-occurrence of node-context pairs in a given window in each RW as a measure of similarity. To learn $\boldsymbol{z}_v$ they maximize the probability of observing v's neighborhood.

5.3.2 Attack Model

We denote with $\hat{\boldsymbol{A}}$ the adjacency matrix of the graph obtained after the attacker has modified certain entries in $\boldsymbol{A}$. We assume the attacker has a given, fixed budget and is only capable of modifying f entries, i.e. $||\hat{\boldsymbol{A}} - \boldsymbol{A}||_0 = 2f$ (times 2 since G is undirected). The goal of the attacker is to damage the quality of the learned embeddings, which in turn harms subsequent learning tasks that use the embeddings such as node classification or link prediction. We consider both a general attack that aims to degrade the embeddings of the network as a whole, and a targeted attack that aims to damage the embeddings regarding a specific target or specific task.

The quality of the embeddings is measured by the loss $\mathcal{L}(\boldsymbol{A}, \boldsymbol{Z})$ of the model under attack, with lower loss corresponding to higher quality, where $\boldsymbol{Z} \in \mathbb{R}^{N\times K}$ is the matrix containing the embeddings of all nodes. Thus, the goal of the attacker is to *maximize* the loss. We can formalize this as the following bi-level optimization problem:

$$\begin{aligned}
&\hat{\boldsymbol{A}}^* = \arg \max_{\hat{\boldsymbol{A}} \in \{0,1\}^{N\times N}} \mathcal{L}(\hat{\boldsymbol{A}}, \boldsymbol{Z}^*) \\
&\boldsymbol{Z}^* = \min_{\boldsymbol{Z}} \mathcal{L}(\hat{\boldsymbol{A}}, \boldsymbol{Z}) \\
&\text{subj. to } \|\hat{\boldsymbol{A}} - \boldsymbol{A}\|_0 = 2f, \quad \hat{\boldsymbol{A}} = \hat{\boldsymbol{A}}^T
\end{aligned} \tag{5.1}$$

Here, $\boldsymbol{Z}^*$ is always the *"optimal"* embedding resulting from the (to be optimized) graph $\hat{\boldsymbol{A}}$, i.e. it minimizes the loss, while the attacker tries to maximize the loss. Solving such a problem is challenging given its discrete and combinatorial nature, therefore we derive efficient approximations using eigenvalue perturbation theory.

5.3.3 General Attack

The first step in RW-based embedding approaches is to sample a set of random walks that serve as a training corpus further complicating the bi-level optimization problem. We have $\boldsymbol{Z}^* = \min_{\boldsymbol{Z}} \mathcal{L}(\{r_1, r_2, \dots\}, \boldsymbol{Z})$ with $r_i \sim \mathrm{RW}(\hat{\boldsymbol{A}})$, where $\mathrm{RW}(\cdot)$ is an intermediate stochastic procedure that generates RWs given the graph $\hat{\boldsymbol{A}}$ which we are optimizing. By flipping (even a few) edges in the graph, the attacker necessarily changes the set of possible

RWs, thus changing the training corpus. Therefore, this sampling procedure precludes any gradient-based methods. To tackle this challenge we leverage recent results that show that (given certain assumptions) RW-based embedding approaches are implicitly factorizing the Pointwise Mutual Information (PMI) matrix [145, 146]. We study DeepWalk as an RW-based representative approach since it's one of the most popular methods and has many extensions. Specifically, we use the results from Qiu et al. [146] to sidestep the stochasticity induced by sampling random walks.

Lemma 5.1 (Qiu et al. [146]). *DeepWalk is equivalent to factorizing $\hat{\boldsymbol{M}}$ with*

$$\hat{\boldsymbol{M}} = \log(\max(\boldsymbol{M}, 1)), \quad \boldsymbol{M} = \tfrac{vol(\boldsymbol{A})}{T \cdot b}\boldsymbol{S}, \quad \boldsymbol{S} = \big(\sum_{r=1}^{T} \boldsymbol{P}^r\big)\boldsymbol{D}^{-1}, \quad \boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A} \tag{5.2}$$

where the embedding $\boldsymbol{Z}^*$ is obtained by the Singular Value Decomposition (SVD) of $\hat{\boldsymbol{M}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ using the top-K largest singular values/vectors, i.e. $\boldsymbol{Z}^* = \boldsymbol{U}_K\boldsymbol{\Sigma}_K^{1/2}$.

Here, $\boldsymbol{D}$ is the diagonal degree matrix with $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$, T is the window size, b is the number of negative samples and $\mathrm{vol}(\boldsymbol{A}) = \sum_{ij} \boldsymbol{A}_{ij}$ is the volume. Since $\boldsymbol{M}$ is sparse and has many zero entries the matrix $\log(\boldsymbol{M})$ where the log is elementwise is ill-defined and dense. To cope with this, similar to the Shifted Positive PMI (PPMI), approach the elementwise maximum is introduced to form $\hat{\boldsymbol{M}}$. Using this insight we see that DeepWalk is equivalent to optimizing $\hat{\boldsymbol{M}}^*_K = \arg\min_{\hat{\boldsymbol{M}}_K} ||\hat{\boldsymbol{M}} - \hat{\boldsymbol{M}}_K||^2_F$, i.e $\hat{\boldsymbol{M}}^*_K$ is the best rank-K approximation to $\hat{\boldsymbol{M}}$. This in turn means that the loss for DeepWalk when using the *optimal* embedding $\boldsymbol{Z}^*$ for a given graph $\boldsymbol{A}$ is $\mathcal{L}_{\mathrm{DW}_1}(\boldsymbol{A}, \boldsymbol{Z}^*) = \big[\sum_{p=K+1}^{|\mathcal{V}|} \sigma_p^2\big]^{1/2}$ where σ_p are the singular values of $\hat{\boldsymbol{M}}(\boldsymbol{A})$ sorted decreasingly $\sigma_1 \geq \sigma_2 \cdots \geq \sigma_{|\mathcal{V}|}$. This result shows that we do not need to construct random walks, nor do we have to (explicitly) learn the embedding $\boldsymbol{Z}^*$ – it is implicitly considered via the singular values of $\hat{\boldsymbol{M}}(\boldsymbol{A})$. Accordingly, we have transformed the bi-level problem into a single-level optimization problem. However, maximizing $\mathcal{L}_{\mathrm{DW}_1}$ is still challenging due to the SVD and the discrete nature of the problem.

Gradient based approach. Maximizing $\mathcal{L}_{\mathrm{DW}_1}$ with a gradient-based approach is not straightforward since we cannot easily backpropagate through the SVD. To tackle this challenge we exploit ideas from eigenvalue perturbation theory [39] to efficiently approximate $\mathcal{L}_{\mathrm{DW}_1}(\boldsymbol{A})$ in closed form without needing to recompute the SVD.

Theorem 5.1. *Let $\boldsymbol{A}$ be the initial adjacency matrix and $\hat{\boldsymbol{M}}(\boldsymbol{A})$ be the respective co-occurrence matrix. Let $\boldsymbol{u}_p$ be the p-th eigenvector corresponding to the p-th largest eigenvalue of $\hat{\boldsymbol{M}}$. Given a perturbed matrix $\boldsymbol{A}'$, with $\boldsymbol{A}' = \boldsymbol{A} + \Delta\boldsymbol{A}$, and the respective change $\Delta\hat{\boldsymbol{M}}$, $\mathcal{L}_{DW_1}(\boldsymbol{A}') \approx \big[\sum_{p=K+1}^{N} \big(\boldsymbol{u}_p^T(\hat{\boldsymbol{M}} + \Delta\hat{\boldsymbol{M}})\boldsymbol{u}_p\big)^2\big]^{1/2} =: \mathcal{L}_{DW_2}(\boldsymbol{A}')$ is an approximation of the loss and the error is bounded by $|\mathcal{L}_{DW_1}(\boldsymbol{A}') - \mathcal{L}_{DW_2}(\boldsymbol{A}')| \leq ||\Delta\hat{\boldsymbol{M}}||_F$.*

For a small $\Delta\boldsymbol{A}$ and thus small $\Delta\hat{\boldsymbol{M}}$ we obtain a very good approximation, and if $\Delta\boldsymbol{A} = \Delta\hat{\boldsymbol{M}} = 0$ then the loss is exact. Intuitively, we can think of using eigenvalue perturbation as analogous to taking the gradient of the loss w.r.t. $\hat{\boldsymbol{M}}(\boldsymbol{A})$. Now, gradient-based optimization is efficient since $\nabla_{\boldsymbol{A}}\mathcal{L}_{\mathrm{DW}_2}(\boldsymbol{A})$ avoids recomputing the eigenvalue decomposition. The gradient provides useful information for a small ϵ change, however, here we are considering discrete flips, i.e. $\epsilon = \pm 1$ so its usefulness is limited. Moreover, using gradient-based optimization requires a dense instantiation of the adjacency matrix which has complexity $O(\mathcal{V}|^2)$ in both runtime and memory, infeasible for large graphs. This motivates the need for our more advanced approach.

Sparse closed-form approach. Our goal is to efficiently compute the change in the loss $\mathcal{L}_{\text{DW}_1}(\boldsymbol{A})$ given a set of flipped edges. To do so we will analyze the change in the spectrum of some of the intermediate matrices and then derive a bound on the change in the spectrum of the co-occurrence matrix, which in turn will give an estimate of the loss. First, we need some results.

Lemma 5.2. *$\boldsymbol{S} = \boldsymbol{U}(\sum_{r=1}^{T} \boldsymbol{\Lambda}^r)\boldsymbol{U}^T$ where the matrices $\boldsymbol{U}$ and $\boldsymbol{\Lambda}$ contain the eigenvectors and eigenvalues solving the generalized eigen-problem $\boldsymbol{A}\boldsymbol{u} = \lambda \boldsymbol{D}\boldsymbol{u}$.*

We see that the spectrum of $\boldsymbol{S}$ (and the spectrum of $\boldsymbol{M}$ by taking the scalars into account) is obtainable from the generalized spectrum of $\boldsymbol{A}$. In contrast to Lemma 5.2, Qiu et al. [146] factorize $\boldsymbol{S}$ using the (non-generalized) spectrum of $\boldsymbol{A}_{\text{norm}} := \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$. As we will show, our formulation using the generalized spectrum of $\boldsymbol{A}$ is key for an efficient approximation.

Let $\boldsymbol{A}' = \boldsymbol{A} + \Delta\boldsymbol{A}$ be the adjacency matrix after the attacker performed some edge flips. As above, by computing the generalized spectrum of $\boldsymbol{A}'$, we can estimate the spectrum of the resulting $\boldsymbol{S}'$ and $\boldsymbol{M}'$. However, recomputing the eigenvalues λ' of $\boldsymbol{A}'$ for every possible set of edge flips is still not efficient for large graphs, preventing an effective application of the method. Thus, we derive our first main result: an efficient approximation bounding the change in the singular values of $\boldsymbol{M}'$ for any edge flip.

Theorem 5.2. *Let $\Delta\boldsymbol{A}$ be a matrix with only 2 non-zero elements, namely $\Delta\boldsymbol{A}_{ij} = \Delta\boldsymbol{A}_{ji} = 1 - 2\boldsymbol{A}_{ij}$ corresponding to a single edge flip (i, j), and $\Delta\boldsymbol{D}$ the respective change in the degree matrix, i.e. $\boldsymbol{A}' = \boldsymbol{A} + \Delta\boldsymbol{A}$ and $\boldsymbol{D}' = \boldsymbol{D} + \Delta\boldsymbol{D}$. Let $\boldsymbol{u}_y$ be the y-th generalized eigenvector of $\boldsymbol{A}$ with generalized eigenvalue λ_y. Then the generalized eigenvalue λ'_y of $\boldsymbol{A}'$ solving $\boldsymbol{A}'\boldsymbol{u}'_y = \lambda'_y\boldsymbol{D}'\boldsymbol{u}'_y$ is approximately $\lambda'_y \approx \lambda_y + \Delta\lambda_y := \tilde{\lambda}'_y$ with:*

$$\Delta\lambda_y = \Delta w_{ij}(2\boldsymbol{u}_{yi} \cdot \boldsymbol{u}_{yj} - \lambda_y(\boldsymbol{u}_{yi}^2 + \boldsymbol{u}_{yj}^2)) \tag{5.3}$$

where $\boldsymbol{u}_{yi}$ is the i-th entry of the vector $\boldsymbol{u}_y$, and $\Delta w_{ij} = (1 - 2\boldsymbol{A}_{ij})$ indicates the direction of the edge flip, i.e ± 1.

By working with the generalized eigenvalue problem in Theorem 5.2 we were able to express $\boldsymbol{A}'$ and $\boldsymbol{D}'$ after flipping an edge as *additive* changes to $\boldsymbol{A}$ and $\boldsymbol{D}$, this in turn enabled us to leverage results from eigenvalue perturbation theory to efficiently approximate the change in the spectrum. If we used $\boldsymbol{A}_{\text{norm}}$ instead, the change to $\boldsymbol{A}'_{\text{norm}}$ would be multiplicative hindering efficient approximation. Using Eq. 5.3, instead of recomputing λ' we only need to compute $\Delta\lambda$ to obtain the approximation $\tilde{\lambda}'$ significantly reducing the complexity when evaluating different edge flips (i, j). Using this result, we can now efficiently bound the change in the singular values of $\boldsymbol{S}'$.

Lemma 5.3. *Let $\boldsymbol{A}'$ be defined as before and $\boldsymbol{S}'$ be the resulting matrix. The singular values of $\boldsymbol{S}'$ are bounded: $\sigma_p(\boldsymbol{S}') \leq \tilde{\sigma}_p := \frac{1}{d'_{min}} \cdot \left| \sum_{r=1}^{T} (\tilde{\lambda'}_{\pi(p)})^r \right|$ where π is a permutation ensuring that the final $\tilde{\sigma}_p$ are sorted decreasingly, and d'_{min} is the smallest degree in $\boldsymbol{A}'$.*

Now we can efficiently compute the loss for a rank-K factorization of $\boldsymbol{M}'$, which we would obtain when performing the edge flip (i, j), i.e.

$$\mathcal{L}_{\mathrm{DW}_3}(\boldsymbol{A}') = \frac{vol(\boldsymbol{A}) + 2\Delta w_{ij}}{T \cdot b} \left[\sum_{p=K+1}^{|V|} \tilde{\sigma}_p^2 \right]^{1/2} \tag{5.4}$$

where $\tilde{\sigma}_p$ is obtained by applying Lemma 5.3 and Theorem 5.2 and the leading constants follow from Lemma 5.1.

While the original loss $\mathcal{L}_{\mathrm{DW}_1}$ is based on the matrix $\hat{\boldsymbol{M}} = \log(\max(\boldsymbol{M}, 1))$, there are unfortunately currently no tools available to analyze the (change in the) spectrum of $\hat{\boldsymbol{M}}$ given the spectrum of $\boldsymbol{M}$. Therefore, we use $\mathcal{L}_{\mathrm{DW}_3}$ as a surrogate loss for $\mathcal{L}_{\mathrm{DW}_1}$ (Yang et al. [81] similarly exclude the element-wise logarithm). As our experiments show, the surrogate loss is effective and we can successfully attack the node embeddings that factorize the actual co-occurrence matrix $\hat{\boldsymbol{M}}$, as well as the original skip-gram model. Similarly, spectral embedding methods [147], factorize the graph Laplacian and have a strong connection to the RW-based approaches. We provide an analysis of their adversarial vulnerability in the appendix (Sec. D.2).

The overall algorithm. Our goal is to maximize $\mathcal{L}_{\mathrm{DW}_3}$ by performing f edge flips. While Eq. 5.3 enables us to efficiently compute the loss for a single edge, there are still $\mathcal{O}(|\mathcal{V}|^2)$ possible flips. To reduce the complexity when *adding* edges we instead form a candidate set by randomly sampling C candidate pairs (non-edges). This introduces a further approximation that nonetheless works well in practice. Since real graphs are usually sparse, for *removing*, all edges are viable candidates with one random edge set aside for each node to ensure we do not have singleton nodes. For every candidate we compute its impact on the loss via $\mathcal{L}_{\mathrm{DW}_3}$ and greedily choose the top f flips.[1]

The runtime complexity of our overall approach is then: $\mathcal{O}(|\mathcal{V}| \cdot |\mathcal{E}| + C \cdot |\mathcal{V}| \log |\mathcal{V}|)$. First, we can compute the generalized eigenvectors of $\boldsymbol{A}$ in a sparse fashion in $\mathcal{O}(|\mathcal{V}| \cdot |\mathcal{E}|)$. Then we sample C candidate edges, and for each we can compute the approximate eigenvalues in constant time (Theorem 5.2). To obtain the final loss, we sort the values leading to the overall complexity. For the examined datasets the wall-clock time for our approach is negligible: on the order of few seconds when calculating the change in eigenvalues. Furthermore, our approach is trivially parallelizable since every candidate edge flip can be evaluated in parallel.

5.3.4 Targeted Attack

If the goal of the attacker is to attack a specific target node $t \in \mathcal{V}$, or a specific downstream task, it is suboptimal to maximize the overall loss via $\mathcal{L}_{DW_*}$. Rather, we should define

[1] Periodically recomputing the exact eigenvalues and eigenvectors when using the greedy approach did not show any benefits.

some other *target specific* loss that depends on t's embedding – replacing the loss function of the *outer* optimization by another one operating on t's embedding. Thus, for any edge flip (i, j) we now need the change in t's embedding – meaning changes in the eigen*vectors* – which is inherently more difficult to compute compared to changes in eigen/singular-*values*. We study two cases: misclassifying a target node (node classification task) and manipulating the similarity of node pairs (link prediction task).

Surrogate embeddings. We define surrogate embeddings such that we can efficiently estimate the change for a given edge flip. Specifically, instead of performing an SVD of $\boldsymbol{M}$ (or equivalently $\boldsymbol{S}$ scaled) we define $\bar{\boldsymbol{Z}}^* = \boldsymbol{U}(\sum_{r=1}^{T} \boldsymbol{\Lambda}^r)$, as in Lemma 5.2. Experimentally, using $\bar{\boldsymbol{Z}}^*$ instead of $\boldsymbol{Z}^*$ as the embedding showed no significant change in the performance on downstream tasks. While we use the surrogate embeddings to select the adversarial edges, for evaluation we use the standard embeddings produced by DeepWalk. We can approximate $\bar{\boldsymbol{Z}}^*(\boldsymbol{A}')$ in closed form by approximating the generalized eigenvectors of $\boldsymbol{A}'$.

Theorem 5.3. *Let $\Delta\boldsymbol{A}, \Delta\boldsymbol{D}$ and Δw_{ij} be defined as before, and $\Delta\lambda_y$ be the change in the y-th generalized eigenvalue λ_y as derived in Theorem 5.2. Then, the y-th generalized eigenvector $\boldsymbol{u}'_y$ of $\boldsymbol{A}'$ after performing the edge flip (i, j) is approximately $\boldsymbol{u}'_y \approx \boldsymbol{u}_y + \Delta\boldsymbol{u}_y$*

$$\Delta\boldsymbol{u}_y = -\Delta w_{ij}(\boldsymbol{A} - \lambda_y\boldsymbol{D})^+\big(- \Delta\lambda_y\boldsymbol{u}_y \circ \boldsymbol{d} + E_i(\boldsymbol{u}_{yj} - \lambda_y\boldsymbol{u}_{yi}) + E_j(\boldsymbol{u}_{yi} - \lambda_y\boldsymbol{u}_{yj})\big) \tag{5.5}$$

where $E_i(x)$ returns a vector of zeros except at position i where the value is x, $\boldsymbol{d}$ is a vector of the node degrees, $\circ$ is the Hadamard product, and $(\cdot)^+$ is the pseudo-inverse.

Computing Eq. 5.5 seems expensive at first due to the pseudo-inverse term. However, note that this term does not depend on the particular edge flip we perform. Thus, we can pre-compute it once and furthermore, parallelize the computation for each y. The additional complexity of computing the pseudo-inverse for all y is $\mathcal{O}(K \cdot |\mathcal{V}|^{2.373})$. Similarly, we can pre-compute $\boldsymbol{u}_y \circ \boldsymbol{d}$, while the rest of the terms are all computable in $\mathcal{O}(1)$. Overall, the wall-clock time for computing the change in the eigenvectors is on the order of few minutes. For any edge flip we can now efficiently compute the optimal embedding $\bar{\boldsymbol{Z}}^*(\boldsymbol{A}')$ using Eq. 5.3 and Eq. 5.5. The t-th row of $\bar{\boldsymbol{Z}}^*(\boldsymbol{A}')$ is the desired embedding for a target node t after the attack.

Targeting node classification. Our goal is to misclassify a target node t given a downstream node classification task. To specify the targeted attack we need to define the candidate flips and the target-specific loss responsible for scoring the candidates. We let the candidate set contain all edges (and non-edges) directly incident to the target node, i.e. $\mathcal{C}_t = \{(v, t) \mid v \in \mathcal{V}, v \neq t\}$. We restricted our experiments to such candidate flips since initial experiments showed that they can do significantly more damage compared to candidate flips in other parts of the graph. This intuitively makes sense since the further away we are from node t we can exert less influence on it. Zügner et al. [121] show similar results (e.g. see their indirect attack). Note that for the general (non-targeted) attack all edges/non-edges are viable candidates.

To obtain the loss, we first pre-train a classifier on the clean embedding $\bar{\boldsymbol{Z}}^*$. Then we predict the class probabilities p_t of the target t using the compromised $\bar{\boldsymbol{Z}}^*_{t,\cdot}$ estimated for a given candidate flip and we calculate the classification margin $m(t) = p_{t,c(t)} - \max_{c \neq c(t)} p_{t,c}$, where $c(t)$ is the ground-truth class for t. That is, our loss is the difference between the probability of the ground-truth and the next most probable class after the attack. Finally, we select the top f flips with smallest margin m (note when $m(t) < 0$ node t is misclassified). In practice, we average over ten randomly trained classifiers. In future work we plan to treat this as a tri-level optimization problem.

Targeting link prediction. The goal of this targeted attack is given a set of target node pairs $\mathcal{T} \subset V \times V$ to decrease the similarity between the nodes that have an edge, and increase the similarity between nodes that do not have an edge by modifying *other* parts of the graph (i.e. we disallow to directly flip pairs in $\mathcal{T}$). For example, in an e-commerce graph representing users and items the goal might be to increase the similarity between a certain item and user by adding/removing connections between other users/items.

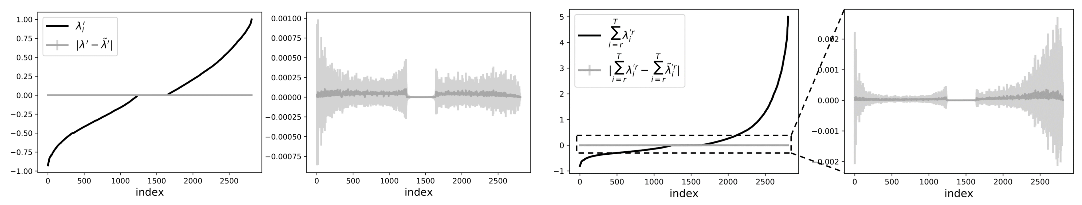
To achieve this goal, we first train an initial clean embedding on the graph excluding the edges in $\mathcal{T}$. Then, for a candidate flip we estimate the embedding $\bar{\boldsymbol{Z}}^*$ (Eq. 5.3 and Eq. 5.5) and use it to calculate the average precision score (AP score) on the target set $\mathcal{T}$, with $\bar{\boldsymbol{Z}}_i^*(\bar{\boldsymbol{Z}}_j^*)^T$ measuring the similarity of nodes i and j, i.e. the likelihood of the link (i, j). Low AP score then indicates that the edges in $\mathcal{T}$ are less likely (non-edges more likely respectively). Finally, we pick the top f flips with the lowest AP scores and use them to poison the network.

5.4 Experimental Evaluation

Setup. Since this is the first work considering adversarial attacks on node embeddings there are no known baselines. Similar to methods that optimize the graph structure [133, 134] we compare with several strong baselines. $\mathcal{B}_{\text{rnd}}$ randomly flips edges (we report averages over ten seeds), $\mathcal{B}_{\text{eig}}$ removes edges based on their eigencentrality in the line graph $L(\boldsymbol{A})$, and $\mathcal{B}_{\text{deg}}$ removes edges based on their degree centrality in $L(\boldsymbol{A})$ (equivalently sum of degrees in the graph). When adding edges we use the same baselines as above now calculated on the complement graph except for $\mathcal{B}_{\text{eig}}$ since it is infeasible to compute even for medium size graphs. $\mathcal{A}_{\text{DW}_2}$ denotes our gradient-based attack, $\mathcal{A}_{\text{DW}_3}$ our closed-form attack, $\mathcal{A}_{\text{link}}$ our targeted link prediction attack, and $\mathcal{A}_{\text{class}}$ is our targeted node classification attack.

The size of the sampled candidate set for adding edges under the general attack is 20K (we report averages over five trials). To evaluate the targeted link prediction attack we form the target pairs $\mathcal{T}$ by randomly sampling 10% of the edges from the clean graph and three times as many non-edges. This setup reflects the fact that in practice the attackers often care more about increasing the likelihood of a new edge, e.g. increasing the chance of recommending an item to a user.

We aim to answer the following questions: (Q1) how good are our approximations of the loss; (Q2) how much damage is caused to the overall embedding quality by our attacks compared to the baselines; (Q3) can we still perform a successful attack when the attacker is restricted; (Q4) what characterizes the selected (top) adversarial edges; (Q5) how do the targeted attacks affect downstream tasks; and (Q6) are the selected adversarial edges transferable to other models.



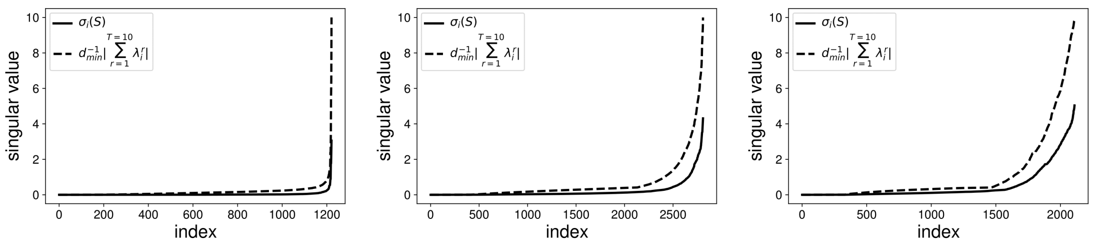
(a) Eigenvalues: $|\lambda' - \tilde{\lambda}'|$ **(b)** Sum of eigenvalues: $\left|\sum_{r=1}^{T} \lambda_i'^r - \sum_{r=1}^{T} \tilde{\lambda}_i'^r\right|$

Figure 5.2: Comparison between the true eigenvalues λ' after performing a flip (i.e. doing a full eigen-decomposition) and our approximation $\tilde{\lambda}'$. Since the difference is several orders of magnitude smaller than the eigenvalues (sums of powers of eigenvalues resp.) themselves, we show a “zoomed-in” view (note the difference in the scale on the y-axis) of the average absolute difference and the standard deviation across 5K randomly selected flips.

We set DeepWalk's hyperparameters to: $T = 5, b = 5, K = 64$ and use logistic regression for classification. We analyze three datasets: Cora-ML ($N = 2810, |E| = 15962$, McCallum et al. [107], Bojchevski and Günnemann [148]) and Citeseer ($N = 2110, |E| = 7336$, Giles et al. [108]) are citation networks commonly used to benchmark embedding approaches, and PolBlogs ($N = 1222, |E| = 33428$, Adamic and Glance [149]) is a graph of political blogs. Since we are in the poisoning setting, in all experiments after choosing the top f flips we re-train the standard embeddings produced by DeepWalk and report the final performance. Note, for the *general attack* the downstream node classification performance is *only a proxy* for estimating the embedding quality after the attack, it is not our goal to damage this task, but rather attack the unsupervised embeddings overall.

5.4.1 Approximation Quality

To estimate the approximation quality we randomly select 20K candidates from the Cora-ML graph and we compute Pearson's R score between the actual loss (including the elementwise logarithm) and our approximations. For example, for dimensionality $K = 32$ we have $R(\mathcal{L}_{\mathrm{DW}_2}, \mathcal{L}_{\mathrm{DW}_1}) = 0.11$ and $R(\mathcal{L}_{\mathrm{DW}_3}, \mathcal{L}_{\mathrm{DW}_1}) = 0.90$ showing that our



(a) PolBlogs **(b)** Cora-ML **(c)** Citeseer

Figure 5.3: Comparison between the singular values $\sigma_i(\boldsymbol{S})$ of $\boldsymbol{S}$ and our upper bound $d_{\min}^{-1}|\sum_{r=1}^{T} \lambda_i^r| \geq \sigma_i(\boldsymbol{S})$ for different graphs.

closed-form strategy approximates the loss significantly better than the gradient-based one. This also holds for $K = 64$ and $K = 128$.

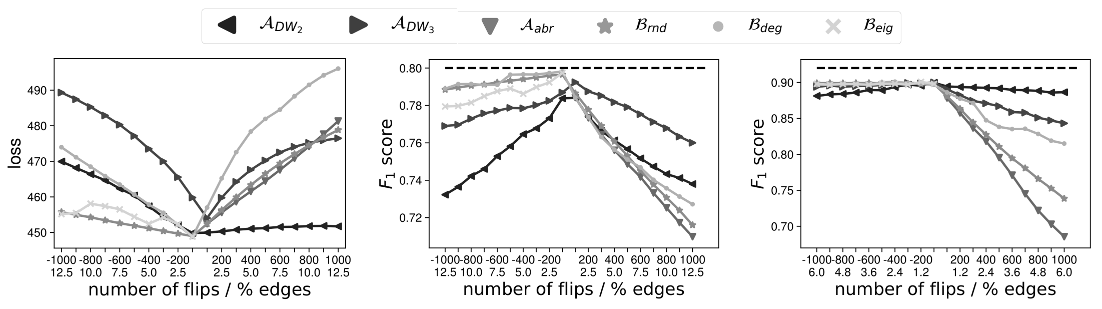
Additionally, we randomly select 5K candidates and we compare the true eigenvalues λ' after performing a flip (i.e. doing a full eigen-decomposition) and our approximation $\tilde{\lambda}'$ based on Theorem 5.2. We can see in Fig. 5.2a that the average absolute difference $|\lambda' - \tilde{\lambda}'|$ is negligible: several orders of magnitude smaller than the eigenvalues themselves. The difference between the terms $|\sum_{r=1}^{T} \lambda_i'^r - \sum_{r=1}^{T} \tilde{\lambda}_i'^r|$ used in Lemma 5.3 is similarly negligible as shown in Fig. 5.2b.

We also compare the true singular values $\sigma_i(\boldsymbol{S})$ of the matrix $\boldsymbol{S}$ and their respective upper bounds $d_{\min}^{-1}|\sum_{r=1}^{T} \lambda_i^r| \geq \sigma_i(\boldsymbol{S})$ obtained from Lemma 5.3. The gap is different across graphs and it is relatively small overall. We plot all these quantities for all graphs on Fig. 5.3. These results together show that we have a good approximation of both the eigenvalues and the singular values, leading to a good overall approximation of the loss.

5.4.2 General Attack

To better understand the attacks we investigate the effect of removing and adding edges separately. We select the top f edges from the respective candidate sets according to our approximation of the loss function. For adding edges, we also implemented an alternative add-by-remove strategy denoted as $\mathcal{A}_{\text{abr}}$. Here, we first add cf-many edges randomly sampled from the candidate set to the graph and subsequently remove $(c-1)f$-many of them (equals to only f changes in total). This strategy performed better empirically. Since the graph is undirected, for each (i, j) we also flip (j, i).

Fig. 5.4 answers question (Q2). Removed/added edges are denoted on the x-axis with negative/positive values respectively. On Fig. 5.4a we see that our strategies achieve a significantly higher loss compared to the baselines when removing edges. To analyze the change in the embedding quality we consider the node classification task (i.e. using it as a proxy to evaluate quality; this is *not* our targeted attack). Interestingly, $\mathcal{B}_{\text{deg}}$ is the



(a) Cora-ML: DeepWalk's loss **(b)** Cora-ML: Classification **(c)** PolBlogs: Classification

Figure 5.4: Vulnerability of the embeddings under the general attack for increasing number of flips. Positive (resp. negative) numbers on the x-axis indicate adding (resp. removing) edges. The percentage of flips is w.r.t. the total number of edges in the clean graph. The dotted line shows the performance on the clean graph before attacking.

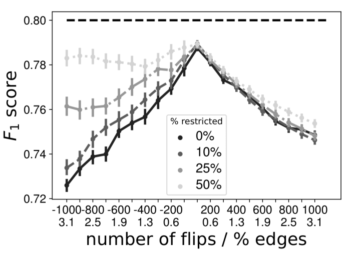
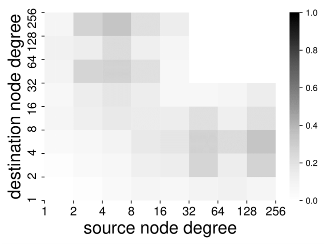
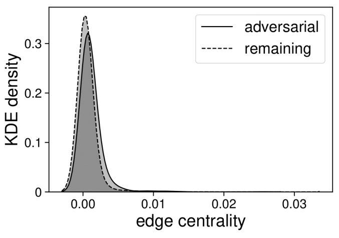


Figure 5.5: Classification performance on Cora-ML for increasingly restricted attacks.



(a) Degree centrality



(b) Edge centrality

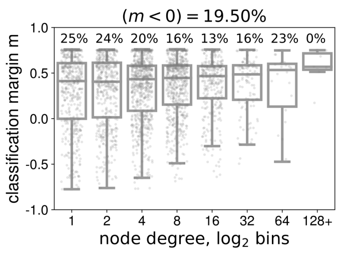
Figure 5.6: Analysis of the adversarial edges selected by $\mathcal{A}_{\mathrm{DW}_3}$ on the Cora-ML graph w.r.t. different centrality measures.

strongest baseline w.r.t. to the loss, but this is not true for the downstream task. As shown in Fig. 5.4b and Fig. 5.4c, our strategies significantly outperform the baselines. As expected, $\mathcal{A}_{\mathrm{DW}_3}$ and $\mathcal{A}_{\mathrm{abr}}$ perform better than $\mathcal{A}_{\mathrm{DW}_2}$. On Cora-ML our attack can cause up to around 5% *more* damage compared to the strongest baseline. On PolBlogs, by adding only 6% edges we can decrease the classification performance by more than 23%, while being more robust to removing edges.

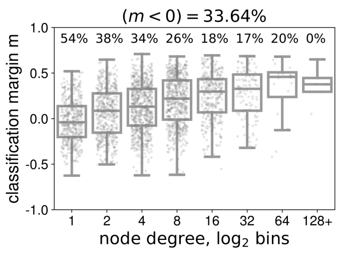
Restricted attacks. In the real world attackers cannot attack any node, but rather only specific nodes under their control, which translates to restricting the candidate set. To evaluate the restricted scenario, we first initialize the candidate sets as before, then we randomly denote a given percentage p_r of nodes as restricted and discard every candidate that includes them. As expected, the results in Fig. 5.5 show that for increasingly restrictive sets with $p_r = 10\%, 25\%, 50\%$, our attack is able to do less damage. However, we always outperform the baselines (not plotted), and even in the case when half of the nodes are restricted ($p_r = 50\%$) we are still able to damage the embeddings. With this we can answer question (Q3): attacks are successful even when restricted.

Analysis of the selected adversarial edges. A natural question to ask is what characterizes the adversarial edges that are selected by our attack, and whether its effectiveness can be explained by a simple heuristic such as attacking "important" edges (e.g. edges that have high centrality). To answer this question we analyze the top 1000 edges selected by $\mathcal{A}_{\mathrm{DW}_3}$ on the Cora-ML dataset. In Fig. 5.6a we analyze the adversarial edges in terms of node degrees. Specifically, for each edge we consider the degree of its source node and its destination node and plot it on the x-axis and y-axis respectively. The heatmap shows the number of adversarial edges divided by total number of edges for each degree (binned logarithmically). We see that low, medium and high-degree nodes are all represented and therefore we conclude that we cannot distinguish between adversarial and non-adversarial edges based solely on their degrees.

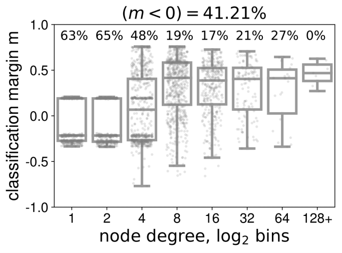
In Fig. 5.6b we plot the edge centrality distribution for the top 1000 adversarial edges and compare it with the edge centrality distribution of the remaining edges. We can see that there is no clear distinction. Both of these findings highlight the need for a principled method such as ours since using intuitive heuristics such as degree centrality or edge centrality cannot identify adversarial edges.



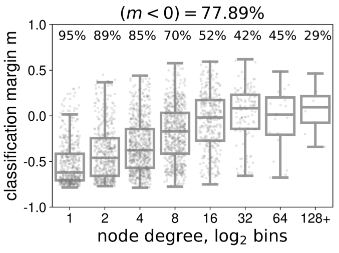
(a) Before attack



(b) Baseline $\mathcal{B}_{\text{rnd}}$

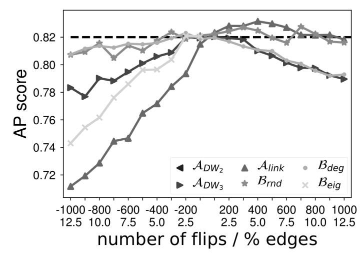


(c) Baseline $\mathcal{B}_{\text{deg}}$

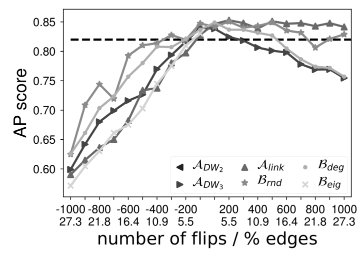


(d) Our $\mathcal{A}_{\text{class}}$ attack

Figure 5.7: Margins for clean and corrupted graphs for different attacks. Each dot represent one node binned logarithmically according to its degree. The number above each (box-) plot indicates the misclassification rate. Lower is better.



(a) Cora-ML



(b) Citeseer

Figure 5.8: Targeted attack on the link prediction task for different datasets.

5.4.3 Targeted Attack

To obtain a better understanding of the performance we study the margin $m(t)$ on Cora-ML before and after the attack considering every node t as a potential target. We allow a budget of only $(d_t + 3)$ flips per each node (where d_t is the degree of the target node t) ensuring that the degrees do not change noticeably after the attack. Each dot in Fig. 5.7 represents one node grouped by its degree in the clean graph (logarithmic bins). We see that low-degree nodes are easier to misclassify ($m(t) < 0$), and that high-degree nodes are more robust in general – the baselines have 0% success. Our method, however, can successfully attack even high-degree nodes. In general, our attack is significantly more effective across all bins – as shown by the numbers on top of each box – with 77.89% nodes successfully misclassified on average compared to e.g. only 33.64% for $\mathcal{B}_{\text{rnd}}$.

For the link prediction task (Fig. 5.8) we are similarly able to cause significant damage – e.g. $\mathcal{A}_{\text{link}}$ achieves almost 10% decrease in performance by flipping around 12.5% of edges on Cora-ML, significantly better than all other baselines. Here again, compared to adding edges, removing has a stronger effect. Overall, answering (Q5), both experiments confirm that our attacks hinder the various downstream tasks.

5.4.4 Transferability

The question of whether attacks learned for one model generalize to other models is important since in practice the attacker might not know the model used by the defender.

Table 5.1: Percent change in F_1 score (lower is better) compared to the clean graph. DW-SVD and DW-SGNS stand for DeepWalk, SE for Spectral Embedding, and LP for Label Propagation.

		DW-SVD	DW-SGNS	node2vec	SE	LP	GCN
Our approach	Cora-ML						
	$f = 250(03.1\%)$	-3.59	-3.97	-2.04	-2.11	-5.78	-3.34
	$f = 500(06.3\%)$	-5.22	-4.71	-3.48	-4.57	-8.95	-2.33
	Citeseer						
	$f = 250(06.8\%)$	-7.59	-5.73	-6.45	-3.58	-4.99	-2.21
	$f = 500(13.6\%)$	-9.68	-11.47	-10.24	-4.57	-6.27	-8.61
$\mathcal{B}_{\text{eig}}$ baseline	Cora-ML						
	$f = 250(03.1\%)$	-0.61	-0.65	-0.57	-0.86	-1.23	-6.33
	$f = 500(06.3\%)$	-0.71	-1.22	-0.64	-0.51	-2.69	-0.64
	Citeseer						
	$f = 250(06.8\%)$	-0.40	-1.16	-0.26	+0.11	-1.08	-0.70
	$f = 500(13.6\%)$	-2.15	-2.33	-1.01	+0.38	-3.15	-1.40

However, if transferability holds, such knowledge is not required. To obtain the perturbed graph, we remove the top f adversarial edges with $\mathcal{A}_{\text{DW}_3}$. The same perturbed graph is used to learn embeddings using several other state-of-the-art approaches: DeepWalk (DW) with both the SVD and the SGNS loss, node2vec [87], Spectral Embedding [147], Label Propagation [150], and GCN [19].

Table 5.1 shows the change in node classification performance compared to the embeddings learned on the clean graph for each method respectively. Answering (Q6), the results show that our attack generalizes: the adversarial edges have a noticeable impact on other models as well. We see that we can damage DeepWalk trained with the skip-gram objective with negative sampling (SGNS) showing that our factorization analysis via SVD is successful. We can even damage the performance of semi-supervised approaches such as Graph Convolutional Networks (GCN) and Label Propagation. Compared to the transferability of the strongest baseline $\mathcal{B}_{\text{eig}}$, shown in the lower section of Table 5.1, we can clearly see that our attack causes significantly more damage.

5.5 Conclusion

We demonstrated that node embeddings are vulnerable to adversarial attacks which can be efficiently computed and have a significant negative effect on downstream tasks such as node classification and link prediction. Furthermore, successfully poisoning the graph is possible with relatively small perturbations and under restriction. More importantly, our attacks generalize - the adversarial edges are transferable across different models. Developing effective defenses and more comprehensive modelling of the attacker's knowledge are important directions for improving network representation learning.

5.6 Retrospective

In this chapter we explored adversarial attacks for unsupervised node embeddings and we showed the we can identify adversarial edges based on the spectrum of the normalized adjacency matrix $\boldsymbol{A}_{\text{norm}}$, or equivalently the generalized spectrum of $(\boldsymbol{A}, \boldsymbol{D})$.

In Sec. D.2 we extend our analysis to spectral embeddings. In contrast to Chapter 3 where the goal was to identify corrupted edges already present in the graph, here the goal is to insert adversarial edges (poison the graph) to damage the embedding. This idea of manipulating the graph structure to induce a certain change in the spectrum is applicable more broadly and can be used to derive adversarial attacks for supervised Graph Neural Networks (GNNs). For example, the SGC model [151] can be written as

$$\text{softmax}(\boldsymbol{S}^l \boldsymbol{X} \boldsymbol{W}) \qquad \boldsymbol{S} = \tilde{\boldsymbol{D}}^{-1/2} \tilde{\boldsymbol{A}} \tilde{\boldsymbol{D}}^{-1/2} \qquad \tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I} \qquad \tilde{\boldsymbol{D}} = \boldsymbol{D} + \boldsymbol{I} \tag{5.6}$$

where $\boldsymbol{W}$ are the learned parameters and $\boldsymbol{X}$ are the node features. As before, we can easily characterize the change in the spectrum of $\boldsymbol{S}^l$ via the change in the generalized spectrum of $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{D}})$. It is also worth noting that for $l = 1$, SGC is equivalent to a one-layer GCN [19]. The question is how to choose the loss function in this case.

Similar to our approach, Chang et al. [152] suggest to maximize $\|(\boldsymbol{S}')^l \boldsymbol{X} - (\boldsymbol{S}')^l_K \boldsymbol{X}\|_F^2$ where $\boldsymbol{S}'$ is obtained from the perturbed adjacency matrix $\boldsymbol{A}' = \boldsymbol{A} + \Delta \boldsymbol{A}$ as in Eq. 5.6, and $(\boldsymbol{S}')^l_K$ is the best K-rank approximation of $(\boldsymbol{S}')^l$. The key difference is that here they additionally consider the node features. More generally, they discuss different models and view them as polynomial filters applied on the eigenvalues of the (shifted) graph Laplacian. Alternatively, as in Sec. 5.3.4, we can easily derive suitable loss functions for targeted attack on SGC and (by extension ignoring non-linear activations) for GCN.

We can draw further connections to our $\boldsymbol{\pi}$-PPNP model [10], which can be summarized as softmax$(\boldsymbol{\Pi} \boldsymbol{H})$ where $\boldsymbol{\Pi}$ is the personalized PageRank matrix, and $\boldsymbol{H} = f_\theta(\boldsymbol{X})$ are the logits (see Chapter 7 for more details). As we discussed in Sec. 2.2 we can express $\boldsymbol{\Pi}$ as

$$\boldsymbol{\Pi} = (1-\alpha)(\boldsymbol{I} - \alpha \boldsymbol{P})^{-1} = (1-\alpha)\sum_{k=1}^{\infty} \alpha^k \boldsymbol{P}^k = (1-\alpha)\boldsymbol{U}(\sum_{k=1}^{\infty} \alpha^k \boldsymbol{\Lambda}^k)\boldsymbol{U}^T \tag{5.7}$$

where $\boldsymbol{P} = \boldsymbol{A}\boldsymbol{D}^{-1} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$. For a given $\boldsymbol{A}'$ we can again estimate the spectrum $(\boldsymbol{\Lambda}', \boldsymbol{U}')$ and approximate $\boldsymbol{\Pi}'$ by truncating the infinite sum. Since the coefficients α^k quickly approach zero this will likely lead to a good approximation. In turn we can derive e.g. a global non-targeted attack on PPNP. For targeted attacks, as we show in Chapter 7, we can exactly compute the worst-case A' for certain threat models. This is guaranteed to either yield an adversarial example or a certificate that such an example does not exist.

In retrospective, it might be a good idea to renormalize the perturbed eigenvectors after adding the change computed using Theorem 5.3 to ensure that $\boldsymbol{U}'\boldsymbol{D}'\boldsymbol{U}'^T = \boldsymbol{I}$ holds, and that $\boldsymbol{U}'$ satisfies the eigenvector properties. Moreover, measuring the deviation from the identity matrix $\boldsymbol{I}$ can serve as another estimate of the approximation quality.

While our attacks were designed for the poisoning scenario, it is likely that the inferred adversarial edges can be successfully used for evasion attacks, since as we show in Sec. 5.4.4 they are transferable. Even though for DeepWalk evasion does not make sense (since it is not inductive), we can use the perturbed graph to attack other models.

Part III

Generative Models

6 Robust Attributed Graph Clustering

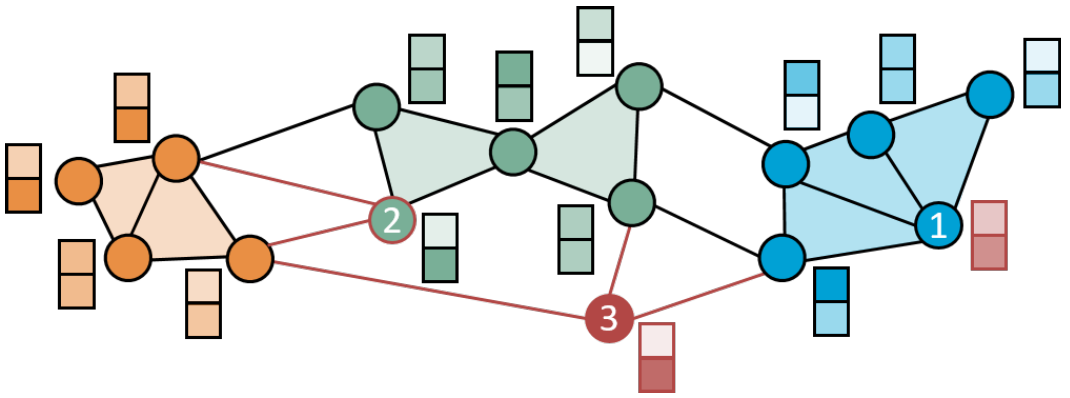


Figure 6.1: Three types of anomalies: node 1 is a partial anomaly w.r.t. the node attributes, node 2 is a partial anomaly w.r.t. the graph structure, and node 3 is a complete anomaly.

6.1 Introduction

Attributed graph clustering is an important research field [153] with application domains from social networks, over e-commerce, to gene analysis. By simultaneously utilizing both network structure and attribute information clustering results can be significantly improved. In real life scenarios datasets are often polluted by rare occurrences, anomalies, or corruptions. A spammer, for example, might be trying to connect to as many nodes as possible, inducing spurious edges and thus obscuring the real clusters in the data. Another source of anomalies are users on a social network obfuscating some of their attributes (age, political affiliation) on purpose due to privacy concerns. Since these anomalies hinder the cluster detection, robust attributed graph clustering methods have been proposed [154, 155]. Instead of first applying anomaly detection [156] then clustering the remaining (cleaned) data, we perform anomaly detection and clustering simultaneously. Such joint learning has shown excellent performance for many other tasks such as regression [53], PCA [54, 157], matrix factorization [158], and auto-regression [55].

The big challenge – not sufficiently addressed by the existing works – is that anomalies in attributed graphs materialize in different ways. Specifically one has to take into account challenging *camouflage* behavior. A user in a social network for example might show corrupted attributes (e.g. to hide their identity) but still their friendship relations are normal. That is, the user is corrupted in only one of the views. We call this a *partial* anomaly. Another example of a partial anomaly is a paper in a citation network, where the content of the paper fits well in some cluster, but the relevant citations are missing. This can happen in an emerging subfield if everyone is not aware of the latest literature.

Fig. 6.1 illustrates this principle in general. Node 1 fits nicely w.r.t. the graph structure but has different attributes compared to the other nodes in its cluster. On the other hand, node 2 perfectly fits the assigned cluster if we only consider its attributes, but it is obviously an anomaly with regards to the network structure. The crucial observation is that we can still identify the partial anomalies' latent cluster since two sources of information are given. In the social network example, despite the users' corrupted attributes we are still able to derive their cluster. We observed such partial anomalies in a variety of real-world datasets.

Existing approaches [154, 155] fail in handling partial anomalies. As soon as a node is corrupted in one of the views, it is marked as an anomaly and no longer belongs to any cluster, even though it might perfectly fit in the other view. Simply speaking, the benefit of having *both* network structure and associated attributes is not taken into account for anomaly detection in existing works. Solving this limitation, we propose a model for attributed graph clustering that accounts for partial anomalies: a node may be corrupted in one view but not in the other. As a strong benefit of this – and in contrast to all existing works – we are still able to infer a node's cluster even if it is (partially) corrupted. This also enables a comparison between the nodes' observed and expected information. For example, for node 1 we can infer different attributes from what we have observed based on its cluster membership.[1] Our model also handles complete anomalies such as node 3, which does not fit to any cluster w.r.t. neither the attributes nor the structure.

To realize these ideas, we propose a probabilistic generative model for attributed graphs, PAICAN (Partial Anomaly Identification and Clustering in Attributed Networks). We jointly model the attribute and network space, as well as the latent cluster assignments and the latent anomaly indicators. We do so by introducing a generalized, anomaly-aware Degree-Corrected Stochastic Block Models (DCSBM) combined with a Beta-Bernoulli mixture model. We can also reason about the uncertainty of the cluster and the anomaly assignments via their posterior distributions. The main contributions of this work are:

- **Generative model**: Our model jointly performs clustering and anomaly detection. It is the first work that realizes robust clustering for attributed graphs following a power-law degree distribution, thus capturing real-life properties.
- **Partial anomalies**: Our model accounts for nodes that might be only partially anomalous enabling us to assign them to meaningful clusters.
- **Scalable algorithm**: Using variational inference and exploiting special properties of our model we propose an algorithm with runtime linear in the number of edges.

6.2 Related Work

For a general overview of attributed graph clustering we refer to [153]. In line with the focus of this chapter we describe here primarily works with the following aspects:

[1]Gao et al. [155] illustrate an example similar to node 1, though still fail to derive a cluster assignment.

robustness to anomalies and principled probabilistic generative models. So far, only two approaches jointly perform clustering and anomaly detection in attributed graphs: CODA [155] and FocusCO [154]. Both detect complete anomalies only. They do not exploit the fact that instances can be partially corrupted. CODA has the additional disadvantages of poor scalability and high sensitivity to hyperparameter choice and initialization, thus, requiring multiple restarts. FocusCO being a semi-supervised method needs labeled data in the form of examples of similar nodes. In contrast, our technique does not require supervision. We compare against both techniques in our experimental study.

Other approaches have been introduced that follow the spirit of generative models, but are *not* robust to anomalies. Note that it is not sufficient to simply treat anomalies as an additional cluster since anomalies might not show any specific clustering behavior. Therefore, we cannot simply use these non-robust techniques for anomaly detection. Among the non-robust methods, PICS [159], CESNA [160] and SIAN [161] only derive point estimates of the learned parameters. BAGC [162] and GBAGC [163] learn a posterior distribution over the model parameters but do not account for the power-law distribution of node degrees. LSBM [164] uses agglomerative multilevel MCMC for inference and hence also learns a posterior distribution. So far, only SIAN, LSBM, and our PAICAN handle realistic network structure, all by relying on variants of DCSBMs. Even though none of the above approaches is able to handle scenarios of corrupted data, we compare against PICS, BAGC, SIAN, and LSBM in our experimental evaluation.

In the related task of multi-view anomaly detection for vector data [165] instances which behave differently across views are detected. Our model of partial anomalies can capture such behavior with the additional benefit of performing clustering. Likewise, our approach handles classical anomaly detection where instances show an overall unusual behavior. Focusing on a different notion of robustness, various methods for subspace clustering on attributed graphs have been introduced [63, 166]. Their goal is to derive robust clusters even if subsets of the attributes are noisy, and do not consider anomalies.

6.3 The PAICAN Model

Let G be an undirected attributed graph with N nodes, and let $\boldsymbol{A}_{ij}$ be an element of the adjacency matrix $\boldsymbol{A} \in \{0,1\}^{N \times N}$ of the graph. We denote with $\boldsymbol{X} \in \{0,1\}^{N \times D}$ the attribute matrix where for each node i, $\boldsymbol{X}_i$ is a D-dimensional vector of binary attributes. We denote with K the number of clusters. An overview of our probabilistic generative model is given in Fig. 6.2. Note that the latent variables $\boldsymbol{c}$ and $\boldsymbol{z}$ are *shared* between the

Table 6.1: The latent variable $\boldsymbol{c}$ indicates different types of (partial) anomalies.

	graph	
attributes	good	anomalous
good	$c_i = 0$	$c_i = 1$
anomalous	$c_i = 2$	$c_i = 3$

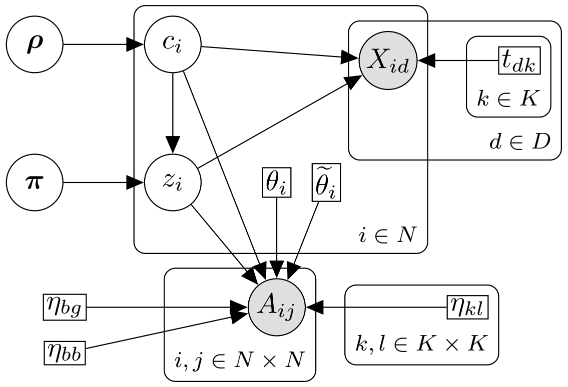


Figure 6.2: Probabilistic Graphical Model of PAICAN.

graph and the attribute space. The variables $\boldsymbol{c} = \{c_i\}_{i=1}^N$ indicate if a node is (partially) anomalous. Given the two views in an attributed graph, anomalies might materialize in different ways, as indicated in Table 6.1. Accordingly, we define $c_i \sim \text{Categorical}(\boldsymbol{\rho})$, $\boldsymbol{\rho} \sim \text{Dirichlet}_4(\boldsymbol{\beta})$ where $\boldsymbol{\rho}$ is the usual Dirichlet prior.

To simplify the notation we introduce the following two shortcuts: $c_i^A = 0$ if $c_i \in \{0, 2\}$, else $c_i^A = 1$, indicating whether the node is good or anomalous w.r.t. the graph, and similarly regarding the attributes $c_i^X = 0$ if $c_i \in \{0, 1\}$, else $c_i^X = 1$.

The latent variables $\boldsymbol{z} = \{z_i\}_{i=1}^N$ encode the group/cluster assignment for each node: $z_i \mid c_i \sim \text{Categorical}(\boldsymbol{\pi}), \boldsymbol{\pi} \sim \text{Dirichlet}_K(\boldsymbol{\alpha})$, and are defined if and only if $c_i \neq 3$. That is, we can only reason about the cluster assignment of node i if it's not a complete anomaly.

6.3.1 Graph Model

To incorporate anomalies into the graph structure we propose an anomaly-aware Degree-Corrected Stochastic Block Model (DCSBM), as a generalization of the well-established DCSBM [167, 168]. The probability of an edge between two nodes i and j is defined[2] as:

$$\boldsymbol{A}_{ij} \sim \begin{cases} \text{Poisson}(\frac{1}{2}^{\mathbb{I}[i=j]}\theta_i\theta_j\eta_{z_i z_j}) & \left. i \le j, c_i^A = 0, c_j^A = 0 \right\} \text{ Case 1} \\ \text{Poisson}(\widetilde{\theta}_i \eta_{\text{bg}}) & \left. i < j, c_i^A = 1, c_j^A = 0 \right\} \text{ Case 2} \\ \text{Poisson}(\widetilde{\theta}_j \eta_{\text{bg}}) & \left. i < j, c_i^A = 0, c_j^A = 1 \right\} \text{ Case 2} \\ \text{Poisson}(\frac{1}{2}^{\mathbb{I}[i=j]}\widetilde{\theta}_i\widetilde{\theta}_j\eta_{\text{bb}}) & \left. i \le j, c_i^A = 1, c_j^A = 1 \right\} \text{ Case 3} \end{cases} \tag{6.1}$$

Case 1: Both nodes are good. If both considered nodes are good we have a classic DCSBM. Using DCSBM as our base model we can capture diverse connection patterns and network topologies such as assortativity, homophily/heterophily, bipartite graphs, etc. The block matrix of group edge probabilities is denoted with $\boldsymbol{\eta} \in [0, 1]^{K \times K}$, and

[2]For undirected graphs we only need to consider $i \le j$. As discussed in [168], in the sparse regime the Poisson distribution represents the Bernoulli distribution well and simplifies the derivations. The established DCSBM and follow up works are also based on the (non-truncated) Poisson distribution.

$\boldsymbol{\theta} = \{\theta_i\}_{i=1}^N$ is the vector representing the *latent degrees* of the nodes. Nodes with higher (latent) degree are more likely to form an edge, thus, enabling us to represent networks with a power-law degree distribution.

Case 2: Only one node is anomalous. If exactly one of the nodes is anomalous, e.g. a spammer i tries to establish a connection with a regular user j, we argue as follows: As in a DCSBM, there might be anomalous nodes which try to establish more connections than other anomalies. Thus, it is reasonable to account for different (latent) degrees of anomalies, indicated by $\widetilde{\boldsymbol{\theta}} = \{\widetilde{\theta}_i\}_{i=1}^N$. However, since the anomalous connection itself is often originated by the anomaly – i.e. a normal user in a social network is not really interested in establishing a connection to a spammer – a high latent degree θ_j of the good node should not be taken into account. Meaning, an anomaly does not specifically prefer nodes with a high degree but uniformly establishes connections to other nodes. Accordingly, only the latent degree of the anomalous node $\widetilde{\theta}$ is considered. Similarly, η_{bg} denotes the base probability of an edge between any anomalous and any good node.

Case 3: Both nodes are anomalous. If both nodes are anomalous, we do not assume any specific clustering behavior. Instead we assume a basic connectivity model which takes into account the nodes' latent degrees $\widetilde{\boldsymbol{\theta}}$ as well as some base probability η_{bb} that denotes the probability of any two anomalous nodes forming an edge.

Discussion of $\boldsymbol{\theta}$ and $\widetilde{\boldsymbol{\theta}}$. The graph model defined above is not only intuitive but also fulfills two interesting properties. First, the maximum likelihood estimate of $\widetilde{\theta}_i$ corresponds to the observed degree. Similarly, the MLE for θ_i is the number of "good" neighbors of node i, i.e. i's degree w.r.t. the good nodes (anomalies are excluded). From a generative perspective, a node is either good or anomalous regarding the graph structure. Thus, for each instance either only $\widetilde{\theta}_i$ or θ_i is used. Hence, in principle, we can combine both vectors $\boldsymbol{\theta}$ and $\widetilde{\boldsymbol{\theta}}$ to a single one. However, since later in our learning procedure we compute each node's posterior distribution, i.e. each node is good/anomalous with a some probability, it is beneficial to model both variables separately.

6.3.2 Attribute Model

We use an anomaly-aware Bernoulli mixture model (BMM). Let $\boldsymbol{T} \in [0,1]^{K\times D}$ be the matrix of mixture probabilities, where $\boldsymbol{T}_{dk}$ represent the probability of attribute d having a value of 1 for the nodes in group k, we have

$$\boldsymbol{X}_{id} \sim \begin{cases} \text{Bernoulli}(\boldsymbol{T}_{dz_i}) & c_i^X = 0 \\ \text{Bernoulli}(b) & c_i^X = 1 \end{cases} \tag{6.2}$$

If the node is good ($c_i^X = 0$) this is a standard BMM. Otherwise we can draw no conclusions about the distribution and we pick a non-informative base probability $b = 0.5$ for the Bernoulli distribution.

The inferred probabilities $\boldsymbol{T}_{dk}$ yield insight into the importance of different attributes for different clusters, and in the context of text data, $\boldsymbol{T}$ takes the role of a topic distribution. It is trivial to extend this model to numerical attributes e.g. via Gaussian Mixture Models.

6.4 Posterior Inference

We are interested in the posterior distribution of the latent variables $\boldsymbol{z}$ and $\boldsymbol{c}$ as well as point estimates for the remaining parameters – MAP estimates for the latent variables $(\boldsymbol{\pi}, \boldsymbol{\rho})$ and MLE for $(\boldsymbol{\eta}, \eta_{\text{bg}}, \eta_{\text{bb}}, \boldsymbol{\theta}, \widetilde{\boldsymbol{\theta}}, \boldsymbol{T})$. For inference, we employ a mean-field variational inference (MFVI) approximation, i.e. we learn a variational distribution q aiming to maximize the evidence lower bound (ELBO) [169]:

$$\mathcal{L} = \mathbb{E}_q[\log p(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{z}, \boldsymbol{c} \mid \boldsymbol{\pi}, \boldsymbol{\rho}, \boldsymbol{\eta}, \eta_{\text{bg}}, \eta_{\text{bb}}, \boldsymbol{\theta}, \widetilde{\boldsymbol{\theta}}, \boldsymbol{T})] - \mathbb{E}_q[\log q(\boldsymbol{z}, \boldsymbol{c})] \tag{6.3}$$

Our coordinate ascent MFVI algorithm has closed-form locally optimal updates and is guaranteed to converge to a local optimum. We use the following mean-field family:

$$\begin{aligned} & q(\boldsymbol{z}, \boldsymbol{c} \mid \boldsymbol{\psi}, \boldsymbol{\phi}) = \prod_i q(z_i \mid \boldsymbol{\psi}_i) \prod_i q(c_i \mid \boldsymbol{\phi}_i) \\ \text{s.t} \quad & q(z_i \mid \boldsymbol{\psi}_i) \sim \mathit{Categorical}(\boldsymbol{\psi}_i), \quad q(c_i \mid \boldsymbol{\phi}_i) \sim \mathit{Categorical}(\boldsymbol{\phi}_i) \end{aligned} \tag{6.4}$$

where the free variational parameters $\boldsymbol{\psi}_i \in [0,1]^K, \boldsymbol{\phi}_i \in [0,1]^4$ satisfy $\sum_{k=1}^K \boldsymbol{\psi}_{ik} = 1$, $\sum_{m=0}^3 \boldsymbol{\phi}_{im} = 1$. As shortcuts for later use, we define $\boldsymbol{\phi}_{i0}^A = \boldsymbol{\phi}_{i0} + \boldsymbol{\phi}_{i2}$, $\boldsymbol{\phi}_{i1}^A = \boldsymbol{\phi}_{i1} + \boldsymbol{\phi}_{i3}$, $\boldsymbol{\phi}_{i0}^X = \boldsymbol{\phi}_{i0} + \boldsymbol{\phi}_{i1}$, $\boldsymbol{\phi}_{i1}^X = \boldsymbol{\phi}_{i2} + \boldsymbol{\phi}_{i3}$, denoting whether node i is corrupted or not in the graph and the attribute space respectively. Given our model, the ELBO decomposes as follows:

$$\begin{aligned} \mathcal{L} = & \underbrace{\mathbb{E}_q[\log p(\boldsymbol{A} \mid \boldsymbol{z}, \boldsymbol{c}, \boldsymbol{\eta}, \eta_{\text{bg}}, \eta_{\text{bb}}, \boldsymbol{\theta}, \widetilde{\boldsymbol{\theta}})]}_{:=\mathcal{L}_A} + \underbrace{\mathbb{E}_q[\log p(\boldsymbol{X} \mid \boldsymbol{z}, \boldsymbol{c}, \boldsymbol{T})]}_{:=\mathcal{L}_X} \\ & + \mathbb{E}_q[\log p(\boldsymbol{z} \mid \boldsymbol{c}, \boldsymbol{\pi})] + \mathbb{E}_q[\log p(\boldsymbol{c} \mid \boldsymbol{\rho})] - \mathbb{E}_q[\log q(\boldsymbol{z}, \boldsymbol{c})] \end{aligned} \tag{6.5}$$

The last four terms are straightforward (see Sec. E.2) and can all be evaluated in linear time w.r.t. the number of nodes and dimensions. For $\mathcal{L}_A$ we obtain:

$$\begin{aligned} \mathcal{L}_A = \sum_{i<j} \Bigg[& \underbrace{\sum_{kl} \boldsymbol{\psi}_{ik} \boldsymbol{\psi}_{jl} \boldsymbol{\phi}_{i0}^A \boldsymbol{\phi}_{j0}^A \big(\boldsymbol{A}_{ij} \log(\theta_i \theta_j \eta_{kl}) - \theta_i \theta_j \eta_{kl}\big)}_{\text{case 1}} \\ & + \underbrace{\boldsymbol{\phi}_{i1}^A \boldsymbol{\phi}_{j0}^A \big(\boldsymbol{A}_{ij} \log(\widetilde{\theta}_i \eta_{\text{bg}}) - \widetilde{\theta}_i \eta_{\text{bg}}\big) + \boldsymbol{\phi}_{i0}^A \boldsymbol{\phi}_{j1}^A \big(\boldsymbol{A}_{ij} \log(\widetilde{\theta}_j \eta_{\text{bg}}) - \widetilde{\theta}_j \eta_{\text{bg}}\big)}_{\text{case 2}} \\ & + \underbrace{\boldsymbol{\phi}_{i1}^A \boldsymbol{\phi}_{j1}^A \big(\boldsymbol{A}_{ij} \log(\widetilde{\theta}_i \widetilde{\theta}_j \eta_{\text{bb}}) - \widetilde{\theta}_i \widetilde{\theta}_j \eta_{\text{bb}}\big)}_{\text{case 3}} \Bigg] \\ + \sum_i \Bigg[& \underbrace{\sum_k \boldsymbol{\psi}_{ik} \boldsymbol{\phi}_{i0}^A \big(\boldsymbol{A}_{ii} \log(\tfrac{1}{2} \theta_i^2 \eta_{kk}) - \tfrac{1}{2} \theta_i^2 \eta_{kk}\big) + \boldsymbol{\phi}_{i1}^A \big(\boldsymbol{A}_{ii} \log(\tfrac{1}{2} \widetilde{\theta}_i^2 \eta_{\text{bb}}) - \tfrac{1}{2} \widetilde{\theta}_i^2 \eta_{\text{bb}}\big)}_{\text{self-loops}} \Bigg] \end{aligned} \tag{6.6}$$

While this term seems to be quadratic in the number of nodes, which is impractical for large networks, we will derive a method that is *linear* in the number of edges.

6.4.1 Variational Expectation-Maximization

We use variational expectation-maximization (EM) [169] which is an iterative update scheme. In the variational E-step we find the optimal variational parameters of q (Eq. 6.8 - Eq. 6.9), and in the variational M-step we compute MAP/ML estimates for the remaining parameters regarding the ELBO (Eq. 6.16 - Eq. 6.18), repeated until convergence. As we will show in Sec. 6.5, in practice the ELBO converges after a small number of iterations (≤ 30). For clarity, we present here the equations for graphs without self-loops ($\boldsymbol{A}_{ii} = 0$). Full derivations and proofs are available in Sec. E.2.

We first note one result which is crucial to obtain linear complexity in the number of edges. This result helps to obtain an efficient computation of the first term in Eq. 6.6. Given the MLE/MAP estimates as derived in the M-Step, it holds for all k:

$$\sum_j \sum_l \boldsymbol{\psi}_{jl} \boldsymbol{\phi}^A_{j0} \theta_j \eta_{kl} = 1 \tag{6.7}$$

E-Step: Update of $\boldsymbol{\psi}$ (i.e. $\boldsymbol{z}$) and $\boldsymbol{\phi}$ (i.e. $\boldsymbol{c}$). We employ coordinate ascent, i.e. we optimize each variational parameter while holding the others fixed. In this case, we can derive closed-form updates for the optimal parameters (see [169] (Ch. 10)).

The optimal variational parameters for the cluster assignments $\boldsymbol{\psi}_{ik}$ are:

$$\begin{aligned} \boldsymbol{\psi}^{\text{new}}_{ik} \propto \exp\Bigg(& \boldsymbol{\phi}^A_{i0} \Bigg[\sum_{j \in \mathcal{N}_i} \boldsymbol{\phi}^A_{j0} \sum_l \boldsymbol{\psi}_{jl} \log(\theta_i \theta_j \eta_{kl}) - \theta_i - \frac{1}{2}\theta_i^2 \eta_{kk} + \theta_i^2 \boldsymbol{\phi}^A_{i0} \sum_l \boldsymbol{\psi}_{il} \eta_{kl} \Bigg] \\ & + \boldsymbol{\phi}^X_{i0} \sum_d \log \operatorname{Ber}(\boldsymbol{X}_{id} \mid \boldsymbol{T}_{dk}) + (1 - \boldsymbol{\phi}_{i3}) \log \pi_k \Bigg) \end{aligned} \tag{6.8}$$

Here, we defined $\mathcal{N}_i$ as the set of neighbors of i and used the result of Eq. 6.7. Normalizing them to 1, i.e. $\sum_k \boldsymbol{\psi}^{\text{new}}_{ik} = 1$, gives the final update.

Similarly, the optimal variational parameters for the anomaly assignments $\boldsymbol{\phi}_{im}$ are:

$$\begin{aligned} \boldsymbol{\phi}^{\text{new}}_{i0} &\propto \exp(\hat{\boldsymbol{\phi}}^A_{i0} + \hat{\boldsymbol{\phi}}^X_{i0} + \log \rho_0) \\ \boldsymbol{\phi}^{\text{new}}_{i1} &\propto \exp(\hat{\boldsymbol{\phi}}^A_{i1} + \hat{\boldsymbol{\phi}}^X_{i0} + \log \rho_1) \\ \boldsymbol{\phi}^{\text{new}}_{i2} &\propto \exp(\hat{\boldsymbol{\phi}}^A_{i0} + \hat{\boldsymbol{\phi}}^X_{i1} + \log \rho_2) \\ \boldsymbol{\phi}^{\text{new}}_{i3} &\propto \exp(\hat{\boldsymbol{\phi}}^A_{i1} + \hat{\boldsymbol{\phi}}^X_{i1} + \log \rho_3 - \sum_k \boldsymbol{\psi}_{ik} \log \pi_k) \end{aligned} \tag{6.9}$$

Here, the updates are based on the following terms regarding the attribute space:

$$\hat{\boldsymbol{\phi}}^X_{i0} = \sum_k \boldsymbol{\psi}_{ik} \Bigg(\sum_d \log \operatorname{Ber}(\boldsymbol{X}_{id} \mid \boldsymbol{T}_{dk}) \Bigg), \qquad \hat{\boldsymbol{\phi}}^X_{i1} = D \log(0.5) \tag{6.10}$$

and regarding the graph space:

$$\begin{aligned}\hat{\boldsymbol{\phi}}^A_{i0} &= \sum_{j\in\mathcal{N}_i} \boldsymbol{\phi}^A_{j0} \sum_{kl} \boldsymbol{\psi}_{ik}\boldsymbol{\psi}_{jl} \log(\theta_i\theta_j\eta_{kl}) - \theta_i(1-\theta_i\boldsymbol{\phi}^A_{i0}\sum_{kl}\boldsymbol{\psi}_{ik}\boldsymbol{\psi}_{jl}\eta_{kl}) \\ &\quad + \sum_{j\in\mathcal{N}_i} \boldsymbol{\phi}^A_{j1}\log(\widetilde{\theta}_j\eta_{\text{bg}}) - \eta_{\text{bg}}(\widetilde{\theta}^B - \boldsymbol{\phi}^A_{i1}\widetilde{\theta}_i) - \frac{1}{2}\theta_i^2\sum_k \boldsymbol{\psi}_{ik}\eta_{kk} \qquad (6.11)\\ \hat{\boldsymbol{\phi}}^A_{i1} &= \log(\widetilde{\theta}_i\eta_{\text{bg}})\sum_{j\in\mathcal{N}_i}\boldsymbol{\phi}^A_{j0} - \eta_{\text{bg}}\widetilde{\theta}_i(g-\boldsymbol{\phi}^A_{i0}) \\ &\quad + \sum_{j\in\mathcal{N}_i}\boldsymbol{\phi}^A_{j1}\log(\widetilde{\theta}_i\widetilde{\theta}_j\eta_{\text{bb}}) - \widetilde{\theta}_i\eta_{\text{bb}}(\widetilde{\theta}^B - \boldsymbol{\phi}^A_{i1}\widetilde{\theta}_i) - \frac{1}{2}\widetilde{\theta}_i^2\eta_{\text{bb}} \qquad (6.12)\end{aligned}$$

where we defined $g = \sum_i \boldsymbol{\phi}^A_{i0}$ and $\widetilde{\theta}^B = \sum_i \boldsymbol{\phi}^A_{i1}\widetilde{\theta}_i$. The crucial observation is that the terms g and $\widetilde{\theta}^B$ can be maintained incrementally, i.e. after updating the parameters of node i both terms can be recomputed in constant time.

Overall, for each node i the updates of $\boldsymbol{\psi}_i$ and $\boldsymbol{\phi}_i$ can be computed in linear time w.r.t. the number of its neighbors $\mathcal{N}_i$. Thus, updating *all* variables (the full E-step) can be done in linear time w.r.t. the number edges, and also linear time in the number of dimensions.

M-Step: Update of remaining parameters. We first simplify $\mathcal{L}_A$ (Eq. 6.6) by introducing some abbreviations:

$$\begin{aligned} d_i^G &= \sum_{j\in\mathcal{N}_i}\boldsymbol{\phi}^A_{j0}, \qquad d_i = |\mathcal{N}_i|, \qquad D_k^G = \sum_i \theta_i\boldsymbol{\psi}_{ik}\boldsymbol{\phi}^A_{i0}, \qquad D^B = \sum_i \widetilde{\theta}_i\boldsymbol{\phi}^A_{i1} \\ m_{\text{bg}} &= \sum_{ij}\boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j0}\boldsymbol{A}_{ij}, \qquad m_{\text{bb}} = \sum_{ij}\boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j1}\boldsymbol{A}_{ij}, \qquad m_{kl} = \sum_{i\neq j}\boldsymbol{A}_{ij}\boldsymbol{\psi}_{ik}\boldsymbol{\psi}_{il}\boldsymbol{\phi}^A_{i0}\boldsymbol{\phi}^A_{j0}\end{aligned}$$

Here D_k^G is the total degree of good nodes in cluster k, and D^B is the total degree of bad nodes. Observe that all these terms can be computed in linear time w.r.t. the number of edges or nodes. Furthermore, as also noted in [167, 168], since the likelihood stays the same if we increase all $\{\theta_i \mid z_i = k\}$ by some factor, given that we also decrease $\eta_{kl}, \forall l$ by the same factor we need constraints to ensure identifiability. Conveniently we pick as constraints (w.r.t. θ and $\widetilde{\theta}$ respectively)

$$D_k^G \overset{!}{=} \sum_i d_i^G\boldsymbol{\psi}_{ik}\boldsymbol{\phi}^A_{i0} \quad \text{and} \quad D^B \overset{!}{=} \sum_i d_i\boldsymbol{\phi}^A_{i1} \qquad (6.13)$$

Combining all aspects and after simplification we obtain:

$$\begin{aligned}\mathcal{L}_A &= \frac{1}{2}\big(\sum_{kl} m_{kl}\log\eta_{kl} - D_k^G D_l^G\eta_{kl} + m_{\text{bb}}\log\eta_{\text{bb}} + D^B D^B\eta_{\text{bb}}\big) \\ &\quad + \frac{1}{2}\sum_i\sum_{kl}\boldsymbol{\psi}_{ik}\boldsymbol{\psi}_{il}\theta_i^2\boldsymbol{\phi}^A_{i0}(\boldsymbol{\phi}^A_{i0}\eta_{kl} - \eta_{kk}) + m_{\text{bg}}\log\eta_{\text{bg}} - gD^B\eta_{\text{bg}} \qquad (6.14)\\ &\quad + \sum_i \boldsymbol{\phi}^A_{i0}\log\theta_i d_i^G + \boldsymbol{\phi}^A_{i1}\log\widetilde{\theta}_i d_i + \sum_i \widetilde{\theta}_i\boldsymbol{\phi}^A_{i1}(1-\boldsymbol{\phi}^A_{i1})(\eta_{\text{bg}} - \frac{1}{2}\eta_{\text{bb}})\end{aligned}$$

We can further simplify this equation based on the following observations. If we have a rather clear decision whether a node is a graph corruption or not, i.e. $\boldsymbol{\phi}_{i1}^A \to 0$ or $\boldsymbol{\phi}_{i1}^A \to 1$, the term $\sum_i \widetilde{\theta}_i \boldsymbol{\phi}_{i1}^A (1 - \boldsymbol{\phi}_{i1}^A)(\eta_{\text{bg}} - \frac{1}{2}\eta_{\text{bb}})$ evaluates to zero. Similarly, for clear clustering assignment, when $\boldsymbol{\psi}_{ik} \to 1$ for a single k, the term $\frac{1}{2}\sum_i \sum_{kl} \boldsymbol{\psi}_{ik}\boldsymbol{\psi}_{il}\theta_i^2 \boldsymbol{\phi}_{i0}^A(\boldsymbol{\phi}_{i0}^A \eta_{kl} - \eta_{kk})$ becomes zero. This is indeed what we observed for real data.

Besides, while most terms in $\mathcal{L}_A$ grow quadratically with N (e.g. $D_k^G D_l^G$), these terms grow only linearly. Thus, removing them only introduces an error of at most $\frac{1}{N}$. Therefore, for large graphs we can safely drop both terms, since the error they introduce approaches zero in the limit case. We provide further analysis in Sec. E.3. Overall, we get:

$$\mathcal{L}_A = \Bigg[\frac{1}{2}\sum_{kl} m_{kl}\log\eta_{kl} - \frac{1}{2}D_k^G D_l^G \eta_{kl} + \sum_i \boldsymbol{\phi}_{i0}^A d_i^G \log\theta_i + \boldsymbol{\phi}_{i1}^A d_i \log\widetilde{\theta}_i + m_{\text{bg}}\log\eta_{\text{bg}} \\ + \frac{1}{2}m_{\text{bb}}\log\eta_{\text{bb}} - \frac{1}{2}D^B D^B \eta_{\text{bb}} - gD^B\eta_{\text{bg}}\Bigg] \cdot \Big(1 + \mathcal{O}\big(\frac{1}{N}\big)\Big) \tag{6.15}$$

Using this in the ELBO – and taking the identifiability constraints via Lagrange multipliers into account – we can directly compute the MAP/ML estimates:

$$\theta_i = d_i^G, \qquad \widetilde{\theta}_i = d_i, \qquad \boldsymbol{T}_{dk} = \frac{\sum_i r_{ik}\boldsymbol{X}_{id}}{R_k} \tag{6.16}$$

$$\eta_{kl} = \frac{m_{kl}}{D_k^G D_l^G}, \qquad \eta_{\text{bg}} = \frac{m_{\text{bg}}}{D^B g}, \qquad \eta_{\text{bb}} = \frac{m_{\text{bb}}}{D^B D^B} \tag{6.17}$$

where we have defined $r_{ik} = \boldsymbol{\phi}_{i0}^X \boldsymbol{\psi}_{ik}$ as expected responsibilities and $R_k = \sum_i r_{ik}$ as expected fraction of ones in the cluster k. The MAP estimates are

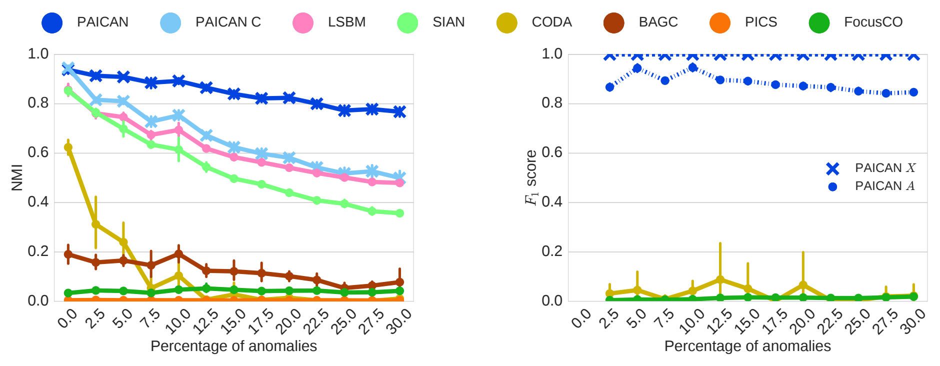
$$\pi_k = \frac{\sum_i (1 - \boldsymbol{\phi}_{i3})\boldsymbol{\psi}_{ik} + \alpha_k}{\sum_i (1 - \boldsymbol{\phi}_{i3}) + \sum_k \alpha_k}, \qquad \rho_m = \frac{\sum_i \boldsymbol{\phi}_{im} + \beta_m}{N + \sum_m \beta_m} \tag{6.18}$$

Using these closed-form estimates the full M-step is also linear in the number of edges.

6.5 Experiments

We compare with CODA [155], FocusCO [154], PICS [159], BAGC [163], LSBM [164], and SIAN [161]. There are no competing methods that handle partial anomalies. To evaluate clustering quality we use normalized mutual information (NMI). To ensure a fair evaluation of the non-robust techniques, we exclude the generated and detected complete corruptions from the NMI calculation. That is, they are not penalized when they add the corruptions to specific clusters – being a big advantage. To make sure that robust techniques do not simply mark all instances as corruptions, we evaluate the anomaly detection using the F_1 score. Both metrics need to be high simultaneously.

For all methods we provide the true number of clusters K. We restart the nondeterministic methods multiple times, and we tune the hyperparameters. For the baselines we pick the solution achieving highest NMI, while for our approach, we simply perform



(a) NMI **(b)** F_1 score

Figure 6.3: Clustering and anomaly detection performance on synthetic data.

several restarts with different initializations and pick the one that gives us the highest likelihood. The different initializations include multiple random cluster assignments, as well as cluster assignments obtained from a baseline DCSBM. Due to this set-up CODA, FocusCO, LSBM, and SIAN get a strong benefit. PICS and BAGC are deterministic.

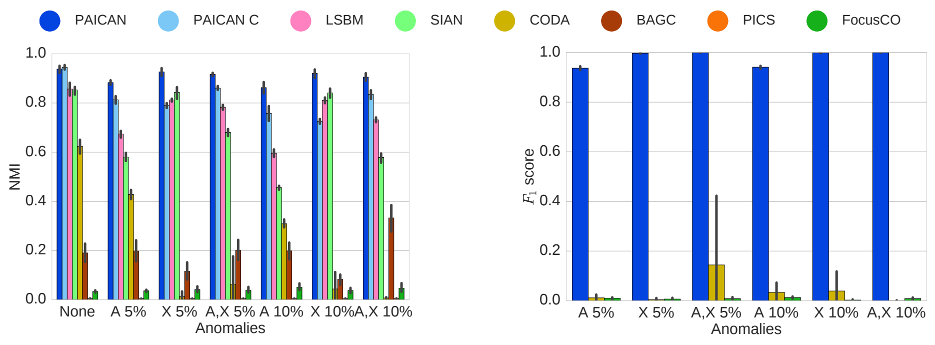
We additionally include a *constrained* version of PAICAN where we disable the detection of anomalies, called PAICAN-C. Note that we do not compare with classic DCSBM since PAICAN-C is essentially a pure attributed DCSBM (without any anomalies), which is a strictly stronger baseline. For more details on the experimental setup including all used datasets and the PAICAN source code see Sec. E.1.

6.5.1 Synthetic Data

To ensure a fair evaluation on synthetic data, we do not simply generate data according to our generative model, we use the configuration model [170] instead. Given a degree sequence $\boldsymbol{\theta}$ that follows a power-law distribution $p(x) \propto x^{-\alpha}$ and density ratio $\frac{E_{\text{in}}}{E}$, where E_{in} is the number of edges within the clusters, the adjacency matrix $\boldsymbol{A}$ for the good nodes is generated according to the configuration model conditioned on randomly generated cluster assignments $\boldsymbol{z}$. We also generate anomalous nodes forming random edges. We generate he attribute matrix $\boldsymbol{X}$ for the good nodes given topic probabilities drawn from $\text{Beta}(0.1, 5)$. We generate the attributes for the anomalous nodes using an uninformative prior. Unless otherwise noted we generate 5000 nodes, 100 attributes and 5 clusters. For each setting of the parameters we generate 10 different random synthetic datasets and report the mean and standard deviation of the relevant metric (NMI or F_1 score).

Robustness and anomaly detection. We are interested in answering the following three questions: (i) How is the clustering quality affected as we increase the percentage of anomalous nodes in the data; (ii) How many anomalous nodes can we actually detect; (iii) What is the effect of partial vs. complete anomalies.

To answer the first two questions, we vary the percentage of anomalies p_a from 0% to 30%, where we distributed the anomalies randomly such that $0.45 \cdot p_a$ are partial anomalies w.r.t. graph space, $0.45 \cdot p_a$ w.r.t. to the attribute space, and the remaining



(a) NMI

(b) F_1 score

Figure 6.4: Performance for different ratios of partially anomalous nodes.

$0.10 \cdot p_a$ are generated as complete anomalies. The results are shown in Fig. 6.3. As we can see our method is robust and is able to maintain a high clustering quality despite the presence of anomalies. If we disable anomaly detection (PAICAN-C), the quality drop is more evident. Similarly, for LSBM and SIAN we can see a clear decrease of the performance as the percentage of anomalies increases.

Considering Fig. 6.3b we can answer the second question. Here, we plot the F_1 score w.r.t. the ground-truth anomalies. Since PAICAN is able to distinguish between graph and attribute corruptions, we can analyze its performance in greater detail. PAICAN A and PAICAN X indicate the F_1 score regarding the graph and attribute corruptions respectively. We observe that PAICAN is slightly better at detecting attribute corruptions, though, in any case it clearly outperforms the competitors[3]. Finally, to answer the third question, we analyze in Fig. 6.4a and Fig. 6.4b how the methods behave when nodes are partially anomalous. As before, we examine the NMI and F_1 score for 0%, 5% and 10% anomalies – here generating either only graph anomalies (A), attribute anomalies (X), or complete anomalies (A, X). Again, PAICAN consistently performs best.

Degree distribution and density ratio. Despite the fact that most real-world networks have power-law like degree distributions, many graph clustering methods are not equipped to properly handle such scenarios. To illustrate this effect we generate data where we vary the power-law exponent to values often encountered in real-world networks ($2.0 \leq \alpha \leq 3.0$) [171]. We also include the simple case of uniform 'blocky' clusters (marked as 'None'), i.e. all degrees are the same. Fig. 6.5a shows the results. Our method clearly outperforms all competitors and is not sensitive to the degree distribution. Furthermore, PAICAN has high stability as shown by the low standard deviation across different runs.

We also explore how the methods behave w.r.t the density ratio $\frac{E_{in}}{E}$. We see in Fig. 6.5b that most methods start failing as soon as the ratio of intra-cluster edges becomes too small, with PAICAN being able to handle the disassortative case the best.

[3]PAICAN is the only method able to handle data with both power-law distributed degrees and anomalies (see also Fig. 6.5). Therefore, although FocusCO and CODA can detect anomalies in principle, they struggle in the power-law distributed case. They are relatively better for the less common case of "blocky" clusters, however, PAICAN still outperforms them (see Fig. E.1).

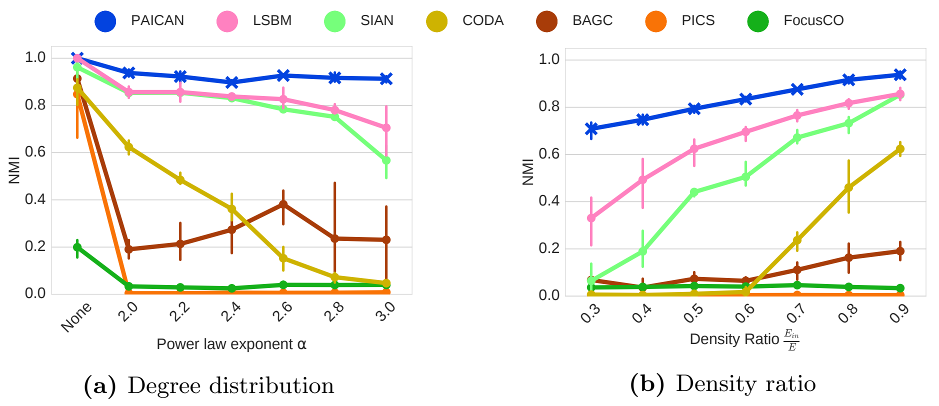


Figure 6.5: Effect of different degree distributions and density ratios on the clustering quality.

Runtime complexity. The complexity of our method is linear in the number of edges and dimensions. This is confirmed in Fig. 6.6. BAGC and LSBM do not scale linear w.r.t. the number of edges, while CODA does not perform well when increasing the number of attributes (note the log scale in Fig. 6.6b). SIAN has the worst scaling out of all the methods even though performs relatively well w.r.t. NMI. All of the methods except CODA are not affected by the number of attributes.

6.5.2 Real-world Data

Dataset description. We use six attributed graph datasets. Cora-ML [107] is a well-known citation network. The Lazega Lawyers [172] is a friendship network among attorneys. HVR [161] is a dataset of highly recombinant malaria parasite genes. DBLP [82] is a co-authorship network of computer science researchers. In the Parliament dataset nodes are parliament members having an edge if they cosigned a bill together. We extract an Amazon co-purchase attributed graph from review data [173] using binary product category indicators as attributes. We also create a new SocialPapers dataset where nodes

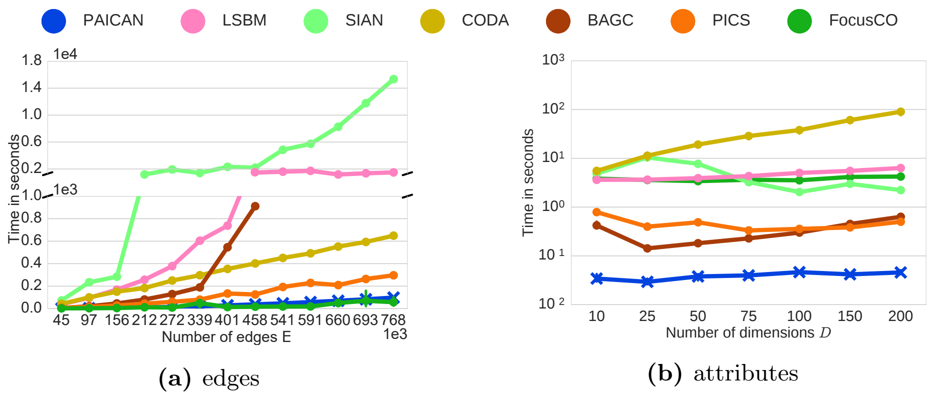


Figure 6.6: Runtime vs. number of edges and attributes. PAICAN scales linearly.

Table 6.2: Comparison of NMI for real-world datasets.

	CODA	FocusCO	BAGC	PICS	LSBM	SIAN	PAICAN
Lawyers	0.50	0.28	0.14	0.27	0.50	0.58	**0.66**
Parliament	0.06	0.00	0.53	0.47	0.77	0.73	**0.78**
Cora-ML	d.n.f.	0.13	0.15	0.04	0.52	0.39	**0.53**
SocialPapers	d.n.f.	0.25	0.17	0.10	0.50	d.n.f.	**0.52**
HVR	0.71	0.50	0.18	0.44	0.83	0.77	**0.89**

represent biomedical papers forming edges if they are frequently mentioned by the same users on social media. See Sec. A.1 for detailed description of all datasets.

Ground-truth evaluation. Table 6.2 shows the NMI achieved by PAICAN and the baselines on datasets with ground-truth labels. We see that PAICAN consistently outperforms the competitors. The non-robust LSBM performs relatively well for most but not all graphs. CODA shows promising results for some graphs, but has scaling issues.

Convergence and runtime. Fig. 6.7 shows the evolution of the ELBO per iteration for the Cora-ML graph. PAICAN quickly converges after a few iterations showing the effectiveness of our variational method. Overall, PAICAN easily handles large graphs as the runtime statistics (seconds per iteration) on the real-world data in Table 6.3 show.

6.5.3 Case Studies

Anomaly detection. We analyzed the DBLP dataset. Overall, PAICAN found 37 partial attribute corruptions, 12 partial graph corruptions, and 71 complete corruptions. Since we have no anomaly ground truth we manually analyze the detected anomalous nodes. As an example, the author Srinivasan Parthasarathy was marked as anomalous in attribute space. Inspecting his ego-network, we see that he fits nicely in graph space since most of his neighbors belong to the same cluster (see Fig. E.3 for a plot of the ego-network). Inspecting his attributes however, we observe that most of his co-authors published in just a few conferences (mainly KDD, ICDM, SDM) while he published in

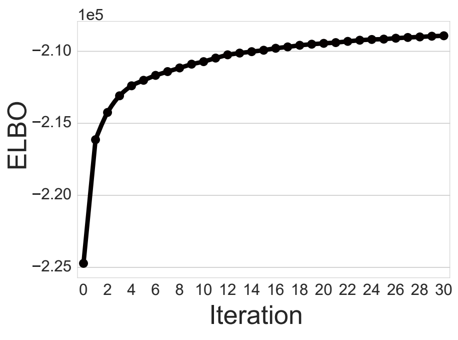


Figure 6.7: ELBO convergence

Table 6.3: Runtime for different datasets

Dataset	Nodes	Edges	Runtime
Lawyers	71	575	0.01 s
HVR	307	3263	0.01 s
Parliament	451	11 646	0.01 s
Cora-ML	2995	8416	0.15 s
DBLP	17 716	105 734	4.46 s
Soc. Papers	20 007	2 088 048	25.56 s

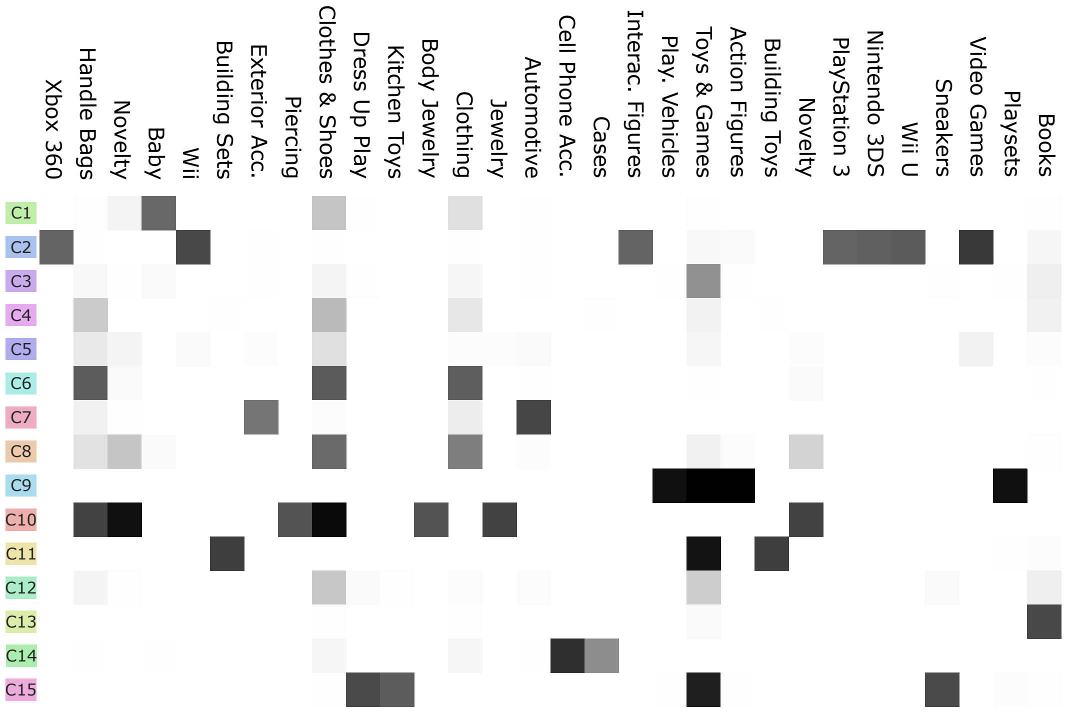


Figure 6.8: Learned topics and detected clusters in the Amazon co-purchase graph.

18 different ones (including e.g. EDBT, IJCAI). We can conclude that he is justifiably marked as a partial attribute anomaly.

Clustering. To enable visual inspection of the clustering we select a small subset ($N = 1549, E = 36934, D = 661$) of the Amazon dataset. The results for $K = 15$ are visualized in Fig. 6.8. The learned topic distribution $\boldsymbol{T}$ is shown, where for easier visualization we only plot dimensions where $\boldsymbol{T}_{dk} > 0.5$ for at least one cluster. Intuitively, this plot shows the "active" categories for each cluster. For example, the products in cluster $C2$ have [Wii U, Nintendo 3DS, PlayStation 3, Xbox 360] as categories, clearly showing a coherent cluster of products related to gaming consoles. Similarly, inspecting the topics of $C10$ shows products about jewelry, and $C14$ cell phone cases related products. This case study demonstrates that PAICAN learns meaningful clusters.

6.6 Conclusion

We proposed PAICAN, a probabilistic model for attributed graph clustering. PAICAN jointly learns the clustering structure as well as potential anomalies. In particular, exploiting the two views of information in attributed graphs, PAICAN introduces the notion of partial anomalies. For learning, we proposed a scalable variational EM algorithm, whose runtime complexity is linear in the number of edges and attributes. Our experimental study confirmed the robustness of PAICAN regarding partial and complete corruptions.

6.7 Retrospective

An interesting aspect which we did not highlight in this chapter (and the original publication) is that the node order (i.e. fixed or random) in which we update $\boldsymbol{\psi}_i$ and $\boldsymbol{\phi}_i$ in the E-step of our coordinate ascent algorithm did not noticeably impact the performance. In practice we sidestep this question and we update $\boldsymbol{\psi}_i$ and $\boldsymbol{\phi}_i$ for all nodes i at once. By taking advantage of vectorized computations on the GPU (via TensorFlow) this significantly reduced the algorithm runtime without affecting the performance.

While it is generally known that the performance of variational inference methods tends to be highly dependent on the initialization, in retrospective, it would have been more transparent to highlight that this also holds for our model. As we discuss in Sec. 6.5 for evaluation we perform several restarts with different initializations and pick the one that gives us the highest likelihood. In practice, the best performance was usually obtained by initializing with clusters inferred with a standard DCSBM (again using VI and updating all parameters at once) whose initialization in turn is based on spectral clustering.

A considerable portion of the effort when developing probabilistic generative models such as PAICAN comes down to the manual derivation of the variational updates which is time-consuming and error-prone. This is especially true for graphs where we have to additionally deal with the non-i.i.d. aspect of the data. Consequently, it is difficult to quickly iterate over different models since the updates have to be re-derived every time. When we originally worked on this model, black-box variational inference (BBVI) methods [174] where starting to gain popularity, although the state of the art was far from today's powerful probabilistic programming languages. If we were to design a similar model today, a more practical approach would be to focus on exploring different model variants by implementing the model in generic probabilistic programming languages such as Edward [175] and taking advantage of BBVI.[4] Then, once we converge on a reasonable model we can manually derive the exact updates if necessary. Relatedly, even though having "free" variational parameters (one per node, unconstrained) is technically more expressive, using an inference model where the variational parameters are the output of a learnable parametric function (e.g. a GNN) tends to perform better in practice [101, 176]. The general principle behind this success is statistical strength sharing, which also prevents overfitting, and was similarly beneficial for our Graph2Gauss model (Chapter 4).

Given the model is generative it would have been beneficial to highlight some of its other inherent advantages. For example, probabilistic generative models are generally useful for inferring missing data which in the context of graphs can be seen as link prediction. In fact, in Bojchevski et al. [7] where we derive NetGAN – a deep generative model for graphs – we showed that a standard DCSBM is a strong baseline. Therefore, it might be interesting to use PAICAN not only for clustering but also for robust link prediction. Since it generalizes DCSBM and additionally considers node features and (partial) anomalies, it is reasonable to conclude that it will also show strong performance.

[4]However, support for non-i.i.d. data is still limited in all probabilistic programming languages.

Part IV

Semi-Supervised Learning

7 Certifiable Robustness to Graph Perturbations

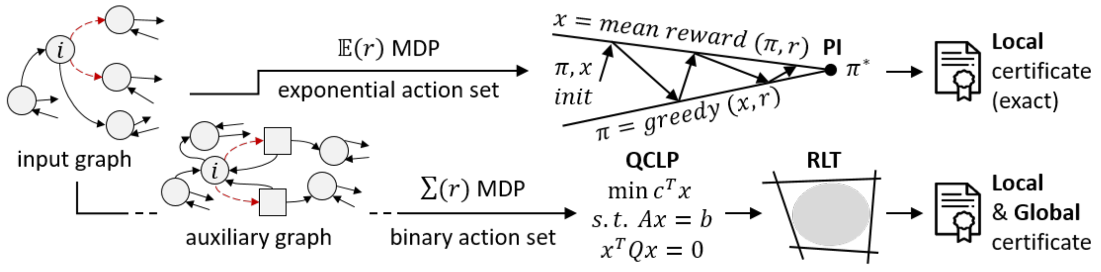


Figure 7.1: Overview of the proposed certificate. The upper part outlines our approach for local budget only: the exact certificate is efficiently computed with policy iteration. The lower part outlines our 3 step approach for local and global budget: (a) formulate an MDP on an auxiliary graph, (b) augment the corresponding LP with quadratic constraints to enforce the global budget, and (c) apply the RLT relaxation to the resulting QCLP (see Sec. 7.4.3 for details).

7.1 Introduction

In Chapter 5 we show that an adversary can easily craft perturbations that can have a significant negative effect on node embeddings. Importantly, even though we derive the attacks based on an unsupervised (surrogate) model, they generalize to other models, and can even damage the performance of semi-supervised approaches such as graph convolutional networks and label propagation. Given the mounting evidence that graph-based models suffer from poor adversarial robustness [121, 122], the question is how to develop effective defenses. This is critical since in the domains where they are often deployed (e.g. the Web), adversaries are pervasive and attacks have a low cost [177–179].

In this chapter we focus on the certifiable robustness of semi-supervised node classification models. Given a single large graph and the class labels of a few nodes the model predicts the labels of the remaining unlabeled nodes. Our goal is to provide provable guarantees that there exists no adversarial perturbation that can change the predictions.

Graph Neural Networks (GNNs) have emerged as the de-facto way to tackle this task, significantly improving performance over the previous state of the art. They are used for various high impact applications across many domains from protein interface prediction [17], to fraud detection [18], and breast cancer classification [21]. Therefore, it is crucial to asses their sensitivity to adversaries and ensure (certify) that they behave as expected.

However, despite their popularity there is scarcely any work on certifying or improving the robustness of GNNs. As shown in Zügner et al. [121] node classification with GNNs is not robust and can even be attacked on multiple fronts – slight perturbations of either the node features or the graph structure can lead to wrong predictions. Moreover, since we are dealing with non i.i.d. data by taking the graph structure into account, robustifying GNNs is more difficult compared to traditional models – perturbing only a few edges affects the predictions for all nodes. What can we do to fortify GNNs and make sure they produce reliable predictions in the presence of adversarial perturbations?

We propose the first method for provable robustness regarding perturbations of the graph structure. Our approach is applicable to a general family of models where the predictions are a linear function of (personalized) PageRank. This includes GNNs [10] and other graph-based models such as label/feature propagation [38, 180]. In Chapter 8 we derive a more general (model-agnostic) certificate. In this chapter we contribute:

- **Certificates**: Given a trained (PageRank-based) model and a general set of admissible graph perturbations we can efficiently verify whether a node is certifiably robust – there exists no perturbation that can change its prediction. We also provide non-robustness certificates via adversarial examples.

- **Robust training**: We investigate robust training schemes based on our certificates and show that they improve both robustness and clean accuracy.

Interestingly, in contrast to existing works on provable robustness [143, 181, 182] that derive bounds (by relaxing the problem), by taking advantage of the structure of the problem we can efficiently compute exact certificates for specific threat models.

7.2 Related Work

Neural networks [43, 183], and recently graph neural networks [121–123] and node embeddings [3] were shown to be highly sensitive to small adversarial perturbations. There exist many (heuristic) approaches aimed at robustifying these models, however, they have only limited usefulness since there is always a new attack able to break them, leading to a cat-and-mouse game between attackers and defenders. A more promising line of research studies certifiable robustness [181, 182, 184]. Certificates provide guarantees that no perturbation regarding a specific threat model will change the prediction of an instance. So far there has been almost no work on certifying graph-based models.

Different heuristics have been explored in the literature to improve robustness of graph-based models: (virtual) adversarial training [185–188], trainable edge weights [189], graph encoder refining and adversarial contrastive learning [190], transfer learning [191], smoothing distillation [185], decoupling structure from attributes [192], measuring logit discrepancy [193], allocating reliable queries [194], representing nodes as Gaussian distributions [111], and Bayesian graph neural networks [195]. Other robustness aspects of graph-based models (e.g. noise or anomalies) have also been investigated [1, 4, 196]. However, none of these works provide provable guarantees or certificates.

Zügner and Günnemann [143] is the only work that proposes robustness certificates for graph neural networks (GNNs). However, their approach can handle perturbations only to the *node attributes*. Our approach is completely orthogonal to theirs since we consider adversarial perturbations to the *graph structure* instead. Furthermore, our certificates are also valid for other semi-supervised learning approaches such as label/feature propagation. Nonetheless, there is a critical need for both types of certificates given that GNNs are shown to be vulnerable to attacks on both the attributes and the structure. In Chapter 8 we consider perturbations of the node features and the graph jointly.

7.3 Background and Preliminaries

Let $G = (\mathcal{V}, \mathcal{E})$ be an attributed graph with $N = |\mathcal{V}|$ nodes and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. We denote with $\boldsymbol{A} \in \{0,1\}^{N \times N}$ the adjacency matrix and $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ the matrix of D-dimensional node features for each node. Given a subset $\mathcal{V}_L \subseteq \mathcal{V} = \{1, \dots, N\}$ of labelled nodes the goal of semi-supervised node classification is to predict for each node $v \in \mathcal{V}$ one class in $\mathcal{C} = \{1, \dots, K\}$. We focus on deriving (exact) robustness certificates for graph neural networks via optimizing personalized PageRank. We also show (Sec. F.1) how to apply our approach for label/feature propagation [180].

Topic-sensitive PageRank. The topic-sensitive (personalized) PageRank [34, 35] vector $\boldsymbol{\pi}_G(\boldsymbol{z})$ for a graph G and a probability distribution over nodes $\boldsymbol{z}$ is defined[1] as $\boldsymbol{\pi}_{G,\alpha}(\boldsymbol{z}) = (1-\alpha)(\boldsymbol{I}_N - \alpha \boldsymbol{A}\boldsymbol{D}^{-1})^{-1}\boldsymbol{z}$. Here $\boldsymbol{D}$ is a diagonal matrix of node out-degrees with $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$. Intuitively, $\boldsymbol{\pi}(\boldsymbol{z})_u$ represent the probability of random walker on the graph to land at node u when it follows edges at random with probability α and teleports back to the node v with probability $(1-\alpha)\boldsymbol{z}_v$. For $\boldsymbol{z} = \boldsymbol{e}_v$, the v-th canonical basis vector, we get the personalized PageRank vector for node v. We drop the index on G, α and $\boldsymbol{z}$ in $\boldsymbol{\pi}_{G,\alpha}(\boldsymbol{z})$ when they are clear from the context. See Sec. 2.2 for more details.

Graph neural networks. As an instance of graph neural network (GNN) methods we consider an adaptation of the recently proposed PPNP approach [10] since it shows superior performance on the semi-supervised node classification task [197]. PPNP unlike message-passing GNNs decouples the feature transformation from the propagation.

We have $\boldsymbol{Y} = \operatorname{softmax}\big(\boldsymbol{\Pi}^{\text{sym}}\boldsymbol{H}\big)$ with

$$\boldsymbol{H}_{v,:} = f_\theta(\boldsymbol{X}_{v,:}), \quad \boldsymbol{\Pi}^{\text{sym}} = (1-\alpha)(\boldsymbol{I}_N - \alpha \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2})^{-1} \tag{7.1}$$

where $\boldsymbol{I}_N$ is the identity, $\boldsymbol{\Pi}^{\text{sym}} \in \mathbb{R}^{N \times N}$ is a symmetric propagation matrix, $\boldsymbol{H} \in \mathbb{R}^{N \times C}$ collects the individual per-node logits, and $\boldsymbol{Y} \in \mathbb{R}^{N \times C}$ collects the final predictions after propagation. A neural network f_θ outputs the logits $\boldsymbol{H}_{v,:}$ by processing the features $\boldsymbol{X}_{v,:}$ of every node v independently. Multiplying them with $\boldsymbol{\Pi}^{\text{sym}}$ we obtain the diffused logits $\boldsymbol{H}^{\text{diff}} := \boldsymbol{\Pi}^{\text{sym}}\boldsymbol{H}$ which implicitly incorporate the graph structure and avoid the expensive multi-hop message-passing procedure.

To make PPNP more amenable to theoretical analysis we replace $\boldsymbol{\Pi}^{\text{sym}}$ with the personalized PageRank matrix $\boldsymbol{\Pi} = (1-\alpha)(\boldsymbol{I}_N - \alpha \boldsymbol{D}^{-1}\boldsymbol{A})^{-1}$ which has a similar

[1]In practice, instead of inverting the matrix we solve the associated sparse linear system of equations.

spectrum. Here each row $\mathbf{\Pi}_{v,:} = \boldsymbol{\pi}(\boldsymbol{e}_v)$ equals to the personalized PageRank vector of node v. This model which we denote as $\boldsymbol{\pi}$-PPNP has similar prediction performance to PPNP. We can see that the diffused logit after propagation for class c of node v is a linear function of its personalized PageRank score: $\boldsymbol{H}^{\text{diff}}_{v,c} = \boldsymbol{\pi}(\boldsymbol{e}_v)^T \boldsymbol{H}_{:,c}$, i.e. a weighted combination of the logits of all nodes for class c. Similarly, the margin

$$m_{c_1,c_2}(v) = \boldsymbol{H}^{\text{diff}}_{v,c_1} - \boldsymbol{H}^{\text{diff}}_{v,c_2} = \boldsymbol{\pi}(\boldsymbol{e}_v)^T (\boldsymbol{H}_{:,c_1} - \boldsymbol{H}_{:,c_2}) \tag{7.2}$$

defined as the difference in logits for node v for two given classes c_1 and c_2 is also linear in $\boldsymbol{\pi}(\boldsymbol{e}_v)$. If $\min_c m_{y_v,c}(v) < 0$, where y_v is the ground-truth label for v, the node is *misclassified* since the prediction equals $\arg\max_c \boldsymbol{H}^{\text{diff}}_{v,c}$.

7.4 Robustness Certificates

7.4.1 Threat Model, Fragile Edges, Global and Local Budget

We investigate the scenario in which a subset of edges in a directed graph are “fragile”, i.e. an attacker has control over them, or in general we are not certain whether these edges are present in the graph. Formally, we are given a set of fixed edges $\mathcal{E}_f \subseteq \mathcal{E}$ that cannot be modified (assumed to be reliable), and set of fragile edges $\mathcal{F} \subseteq (\mathcal{V} \times \mathcal{V}) \setminus \mathcal{E}_f$. For each fragile edge $(i, j) \in \mathcal{F}$ the attacker can decide whether to include it in the graph or exclude it from the graph, i.e. set $\boldsymbol{A}_{ij}$ to 1 or 0 respectively. For any subset of included $\mathcal{F}_+ \subseteq \mathcal{F}$ edges we can form the perturbed graph $\tilde{G} = (\mathcal{V}, \tilde{\mathcal{E}} := \mathcal{E}_f \cup \mathcal{F}_+)$. An *excluded* fragile edge $(i, j) \in \mathcal{F} \setminus \mathcal{F}_+$ is a non-edge in $\tilde{G}$. This formulation is general, since we can set $\mathcal{E}_f$ and $\mathcal{F}$ arbitrarily. For example, for our certificate scenario given an existing clean graph $G = (\mathcal{V}, \mathcal{E})$ we can set $\mathcal{E}_f = \mathcal{E}$ and $\mathcal{F} \subseteq (\mathcal{V} \times \mathcal{V}) \setminus \mathcal{E}$ which implies the attacker can only add new edges to obtain perturbed graphs $\tilde{G}$. Or we can set $\mathcal{E}_f = \emptyset$ and $\mathcal{F} = \mathcal{E}$ so that the attacker can only remove edges, and so on. There are $2^{|\mathcal{F}|}$ (exponential) number of valid configurations leading to different perturbed graphs which highlights that certificates are challenging for graph perturbations.

In reality, perturbing an edge is likely to incur some cost for the attacker. To capture this we introduce a *global budget*. The constraint $|\tilde{\mathcal{E}} \setminus \mathcal{E}| + |\mathcal{E} \setminus \tilde{\mathcal{E}}| \leq B$ implies that the attacker can make at most B perturbations. The first term equals to the number of newly added edges, and the second to the number of removed existing edges. Here, including an edge that already exists does not count towards the budget. This is only a design choice that depends on the application, and our method works in general. Furthermore, perturbing many edges for a single node might not be desirable, thus we also allow to limit the number of perturbations locally. Let $\mathcal{E}^v = \{(v, j) \in \mathcal{E}\}$ be the set of edges that share the same source node v. Then, the constraint $|\tilde{\mathcal{E}}^v \setminus \mathcal{E}^v| + |\mathcal{E}^v \setminus \tilde{\mathcal{E}}^v| \leq b_v$ enforces a local budget b_v for the node v. By setting $b_v = |\mathcal{F}^v|$ and $B = |\mathcal{F}|$ we can model an unconstrained attacker. Letting $\mathcal{P}(\mathcal{F})$ be the power set of $\mathcal{F}$, we define the set of admissible perturbed graphs

$$\begin{aligned}\mathcal{Q}_{\mathcal{F}} = \{(\mathcal{V}, \tilde{\mathcal{E}} := \mathcal{E}_f \cup \mathcal{F}_+) \mid \mathcal{F}_+ \in \mathcal{P}(\mathcal{F}),\ &|\tilde{\mathcal{E}} \setminus \mathcal{E}| + |\mathcal{E} \setminus \tilde{\mathcal{E}}| \leq B, \\ &|\tilde{\mathcal{E}}^v \setminus \mathcal{E}^v| + |\mathcal{E}^v \setminus \tilde{\mathcal{E}}^v| \leq b_v, \forall v\}\end{aligned} \tag{7.3}$$

7.4.2 Robustness Certificates

Problem 7.1. *Given a graph G, a set of fixed $\mathcal{E}_f$ and fragile $\mathcal{F}$ edges, global budget B, local budgets b_v, target node t, and a model with logits $\boldsymbol{H}$. Let y_t denote the class of node t (predicted or ground-truth). The worst-case margin between class y_t and class c under any admissible perturbation $\tilde{G} \in \mathcal{Q}_{\mathcal{F}}$ is:*

$$m^*_{y_t,c}(t) = \min_{\tilde{G}\in\mathcal{Q}_{\mathcal{F}}} m_{y_t,c}(t) = \min_{\tilde{G}\in\mathcal{Q}_{\mathcal{F}}} \boldsymbol{\pi}_{\tilde{G}}(\boldsymbol{e}_t)^T(\boldsymbol{H}_{:,y_t} - \boldsymbol{H}_{:,c}) \tag{7.4}$$

If $m^*_{y_t,*}(t) = \min_{c\neq y_t} m^*_{y_t,c}(t) > 0$, node t is certifiably robust w.r.t. the logits $\boldsymbol{H}$, and the admissible set $\mathcal{Q}_{\mathcal{F}}$.

Our goal is to verify whether no admissible $\tilde{G} \in \mathcal{Q}_{\mathcal{F}}$ can change the prediction for a target node t. From Problem 7.1 we see that if the worst margin over all classes $m^*_{y_t,*}(t)$ is positive, then $m^*_{y_t,c}(t) > 0$, for all $y_t \neq c$, which implies that there exists *no* adversarial example within $\mathcal{Q}_{\mathcal{F}}$ that leads to a change in the prediction to some other class c, that is, the logit for the given class y_t is always largest.

Challenges and core idea. From a cursory look at Eq. 7.4 it appears that finding the minimum is intractable. After all, our domain is discrete and we are optimizing over exponentially many configurations. Moreover, the margin is a function of the personalized PageRank which has a non-trivial dependency on the perturbed graph. But there is hope: For a fixed $\boldsymbol{H}$, the margin $m_{y_t,c}(t)$ is a linear function of $\boldsymbol{\pi}(\boldsymbol{e}_t)$. Thus, Problem 7.1 reduces to optimizing a linear function of personalized PageRank over a specific constraint set. This is the core idea of our approach. As we will show, if we consider only local budget constraints the exact certificate can be efficiently computed. This is in contrast to most certificates for neural networks that rely on different relaxations to make the problem tractable. Including the global budget constraint, however, makes the problem hard (see Sec. F.5). For this case we derive an efficient to compute lower bound on the worst-case margin. Thus, if the lower bound is positive we can still guarantee that our classifier is robust w.r.t. the set of admissible perturbations.

7.4.3 Optimizing Topic-sensitive PageRank

We are interested in optimizing a linear function of the topic-sensitive PageRank vector of a graph by modifying its structure. That is, we want to configure a set of fragile edges into included/excluded to obtain a perturbed graph $\tilde{G}$ maximizing the objective. Formally, we study the general problem:

Problem 7.2. *Given a graph G, a set of admissible perturbations $\mathcal{Q}_{\mathcal{F}}$ as in Problem 7.1, and any fixed $\boldsymbol{z}, \boldsymbol{r} \in \mathbb{R}^N, \alpha \in (0,1)$ solve the following optimization problem:*

$$\max_{\tilde{G}\in\mathcal{Q}_{\mathcal{F}}} \boldsymbol{r}^T \boldsymbol{\pi}_{\tilde{G},\alpha}(\boldsymbol{z}) \tag{7.5}$$

Setting $\boldsymbol{r} = -(\boldsymbol{H}_{:,y_t} - \boldsymbol{H}_{:,c})$ and $\boldsymbol{z} = \boldsymbol{e}_t$, we see that Problem 7.1 is a special case of Problem 7.2. We can think of $\boldsymbol{r}$ as a reward/cost vector, i.e. $\boldsymbol{r}_v$ is the reward that a random walker obtains when visiting node v. The objective value $\boldsymbol{r}^T\boldsymbol{\pi}(\boldsymbol{z})$ is proportional to the overall reward obtained during an infinite random walk with teleportation since $\boldsymbol{\pi}(\boldsymbol{z})_v$ exactly equals to the frequency of visits to v.

Variations and special cases of this problem have been previously studied [139, 198–203]. Notably, Fercoq et al. [201] cast the problem as an average cost infinite horizon Markov decision process (MDP), also called ergodic control problem, where each node corresponds to a state and the actions correspond to choosing a subset of included fragile edges, i.e. we have $2^{|\mathcal{F}^v|}$ actions at each state v (see also Fig. 7.2a). They show that despite the exponential number of actions, the problem can be efficiently solved in polynomial time, and they derive a value iteration algorithm with different local constraints. They enforce that the final perturbed graph has at most b_v total number of edges per node, while we enforce that at most b_v edges per node are perturbed (Sec. 7.4.1).

7.4.4 Optimizing PageRank with Local Constraints Only

Inspired by the MDP idea we derive a policy iteration (PI) algorithm which also runs in polynomial time [202]. Intuitively, every policy corresponds to a perturbed graph in $\mathcal{Q}_{\mathcal{F}}$, and each iteration improves the policy. The PI algorithm allows us to: incorporate our local constraints easily, take advantage of efficient solvers for sparse systems of linear equations (line 3 in Algorithm 7.1), and implement the policy improvement step in parallel (lines 4-6 in Algorithm 7.1). It can easily handle very large sets of fragile edges and it scales to large graphs.

Proposition 7.1. *Algorithm 7.1 finds an optimal solution for Problem 7.2 with only local constraints in a number of steps independent of the graph size.*

We provide the proof in Sec. F.3 in the appendix. The main idea for Algorithm 7.1 is starting from a random policy, in each iteration we first compute the mean reward before teleportation $\boldsymbol{x}$ for the current policy by solving a sparse linear system of equations (line 3), and then greedily select the top b_v edges that improve the policy (lines 4-6). This algorithm is guaranteed to converge to the optimal policy, and thus to the optimal configuration of fragile edges.

Certificate for local budget only. Proposition 7.1 implies that for local constraints only, the optimal solution does not depend on the teleport vector $\boldsymbol{z}$. Regardless of the

Algorithm 7.1 Policy Iteration with Local Budget

```
Require: Graph $G = (\mathcal{V}, \mathcal{E})$, reward $\boldsymbol{r}$, fixed $\mathcal{E}_f$ and fragile $\mathcal{F}$ edges, local budgets $b_v$
1: Initialization: $\mathcal{W}_0 \subseteq \mathcal{F}$ as any arbitrary subset, $\boldsymbol{A}^G$ corresponding to $G$
2: while $\mathcal{W}_k \neq \mathcal{W}_{k-1}$ do
3:     Solve $(\boldsymbol{I}_N - \alpha \boldsymbol{D}^{-1}\boldsymbol{A})\boldsymbol{x} = \boldsymbol{r}$ for $\boldsymbol{x}$, where $\boldsymbol{A}_{ij} = 1 - \boldsymbol{A}^G_{ij}$ if $(i,j) \in \mathcal{W}_k$ ▷ flip edges
4:     Let $l_{ij} \leftarrow (1 - 2\boldsymbol{A}^G_{ij})(\boldsymbol{x}_j - \frac{\boldsymbol{x}_i - \boldsymbol{r}_i}{\alpha})$ for all $(i,j) \in \mathcal{F}$   ▷ calculate the improvement
5:     Let $\mathcal{L}_v \leftarrow \{(v,j) \in \mathcal{F} \mid l_{vj} > 0 \ \wedge \ l_{vj} \geq \text{top } b_v \text{ largest } l_{vj}\}, \ \forall v \in \mathcal{V}$
6:     $\mathcal{W}_k \leftarrow \bigcup_v \mathcal{L}_v, \quad k \leftarrow k + 1$
7: end while
8: return $\mathcal{W}_k$        ▷ optimal graph $\tilde{G} \in \mathcal{Q}_{\mathcal{F}}$ obtained by flipping all $(i,j) \in \mathcal{W}_k$ of $G$
```

node t (i.e. which $\boldsymbol{z} = \boldsymbol{e}_t$ in Eq. 7.4), the optimal edges to perturb are the same if the admissible set $\mathcal{Q}_\mathcal{F}$ and the reward $\boldsymbol{r}$ are the same. This means that for a fixed $\mathcal{Q}_\mathcal{F}$ we only need to run the algorithm $K \times K$ times to obtain the certificates *for all* N nodes: For each pair of classes c_1, c_2 we have a different reward vector $\boldsymbol{r} = -(\boldsymbol{H}_{:,c_1} - \boldsymbol{H}_{:,c_2})$, and we can recover the *exact* worst-case margins $m^*_{y_t,*}(t)$ for all N nodes by just computing $\boldsymbol{\Pi}$ on the resulting $K \times K$ many perturbed graphs $\tilde{G}$. Now, $m^*_{y_t,*}(t) > 0$ implies certifiable robustness, while $m^*_{y_t,*}(t) < 0$ implies certifiable *non-robustness* due to the exactness of our certificate, i.e. we have found an adversarial example for node t.

7.4.5 Optimizing PageRank with Global and Local Constraints

Algorithm 7.1 cannot handle a global budget constraint, and in general solving Problem 7.2 with global budget is NP-hard. More specifically, it generalizes the Link Building problem [203] – find the set of k optimal edges that point to a given node such that its PageRank score is maximized – which is W[1]-hard and for which there exists no fully-polynomial time approximation scheme (FPTAS). It follows that Problem 7.2 is also W[1]-hard and allows no FPTAS. We provide the proof and more details in Sec. F.5 in the appendix.

Therefore, we develop an alternative approach that consists of three steps and is outlined in the lower part of Fig. 7.1:

(a) We propose an alternative un constrained MDP based on an auxiliary graph which reduces the action set from exponential to binary, adding only $|\mathcal{F}|$ auxiliary nodes

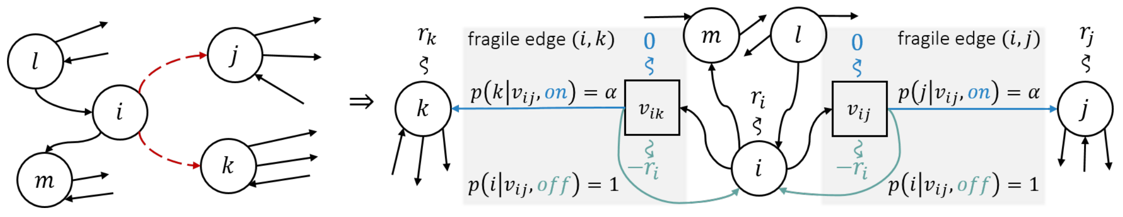
(b) We reformulate the problem as a non-convex Quadratically Constrained Linear Program (QCLP) to be able to handle the global budget

(c) We utilize the Reformulation Linearization Technique (RLT) to construct a convex relaxation of the QCLP enabling us to efficiently compute a lower bound on the worst-case margin

(a) Auxiliary graph. Given an input graph we add one auxiliary node v_{ij} for each fragile edge $(i, j) \in \mathcal{F}$. We define a total cost infinite horizon MDP on this auxiliary graph (Fig. 7.2b) that solves Problem 7.2 *without* constraints. The MDP is defined by the 4-tuple $(\mathcal{S}, (\mathcal{A}_i)_{i\in\mathcal{S}}, p, r)$, where $\mathcal{S}$ is the state space (preexisting and auxiliary nodes), and $\mathcal{A}_i$ is the set of admissible actions in state i.

Given action $a \in \mathcal{A}_i$, $p(j|i, a)$ is the probability to go to state j from state i and $r(i, a)$ the instantaneous reward. Each *preexisting* node i has a single action $\mathcal{A}_i = \{a\}$, reward $r(i, a) = \boldsymbol{r}_i$, and uniform transitions $p(v_{ij}|i, a) = d_i^{-1}, \forall v_{ij} \in \mathcal{F}^i$, discounted by α for the fixed edges $p(j|i, a) = \alpha \cdot d_i^{-1}, \forall (i, j) \in \mathcal{E}_f$, where $d_i = |\mathcal{E}_f^i \cup \mathcal{F}^i|$ is the degree. For each *auxiliary* node we allow two actions $\mathcal{A}_{v_{ij}} = \{\text{on}, \text{off}\}$. For action "off" node v_{ij} goes back to node i with probability 1 and obtains reward $-\boldsymbol{r}_i$: $p(i|v_{ij}, \text{off}) = 1, r(v_{ij}, \text{off}) = -\boldsymbol{r}_i$. For action "on" node v_{ij} goes only to node j with probability α (the model is substochastic) and obtains 0 reward: $p(j|v_{ij}, \text{on}) = \alpha, r(v_{ij}, \text{on}) = 0$. We introduce fewer auxiliary nodes compared to previous work [199, 204].

(b) Global and local budgets QCLP. Based on this unconstrained MDP, we can derive a corresponding linear program (LP) solving the same problem [205]. Since the



(a) $\mathcal{A}_i$=$\{\emptyset, \{j\}, \{k\}, \{j,k\}\}$ **(b)** $\mathcal{A}_i = \{a\}$ and $\mathcal{A}_{v_{ij}} = \mathcal{A}_{v_{ik}} = \{\text{off}, \text{on}\}$

Figure 7.2: Construction of the auxiliary graph. For each fragile edge (i,j) marked with a red dashed line, we add one node v_{ij} and two actions: {on, off} to the auxiliary graph. If the edge is configured as "on" v_{ij} goes back to node i with probability 1. If configured as "off" it goes only to node j with probability α.

MDP on the auxiliary graph has (at most) binary action sets, the LP has only $2|\mathcal{V}|+3|\mathcal{F}|$ constraints and variables. This is in strong contrast to the LP corresponding to the previous average cost MPD [201] operating directly on the original graph that has an exponential number of constraints and variables. Lastly, we enrich the LP for the auxiliary graph MDP with additional constraints enforcing the local and global budgets. The local budget constraints are linear, however, the global budget requires quadratic constraints resulting in a quadratically constrained linear program (QCLP) equivalent to Problem 7.2.

Proposition 7.2. *Solving the following QCLP with decision variables $x_v, x^0_{ij}, x^1_{ij}, \beta^0_{ij}, \beta^1_{ij}$ is equivalent to solving Problem 7.2 with local and global constraints, i.e. the value of the objective function is the same in the optimal solution*

$$\max \sum\nolimits_{v\in\mathcal{V}} x_v \boldsymbol{r}_v - \sum\nolimits_{(i,j)\in\mathcal{F}} x^0_{ij}\boldsymbol{r}_i \tag{7.6a}$$

$$x_v - \underbrace{\alpha\sum\nolimits_{(i,v)\in\mathcal{E}_f}\frac{x_i}{d_i}}_{\textit{incoming fixed edges}} - \underbrace{\alpha\sum\nolimits_{(j,v)\in\mathcal{F}} x^1_{jv}}_{\textit{incoming "on" edges}} - \underbrace{\sum\nolimits_{(v,k)\in\mathcal{F}} x^0_{vk}}_{\textit{returning "off" edges}} = (1-\alpha)\boldsymbol{z}_v \tag{7.6b}$$

$$x^0_{ij} + x^1_{ij} = \frac{x_i}{d_i}, \qquad x^0_{ij} \geq 0, \qquad x^1_{ij} \geq 0 \tag{7.6c}$$

$$\sum\nolimits_{(v,i)\in\mathcal{F}} \underbrace{[(v,i)\in\mathcal{E}]x^0_{ij}}_{\textit{removed existing edges}} + \underbrace{[(v,i)\notin\mathcal{E}]x^1_{ij}}_{\textit{newly added edges}} \leq \frac{x_v}{d_v}b_v, \qquad x_v \geq 0 \tag{7.6d}$$

$$x^0_{ij}\beta^1_{ij} = 0, \quad x^1_{ij}\beta^0_{ij} = 0, \quad \beta^1_{ij} = 1-\beta^0_{ij}, \quad 0 \leq \beta^0_{ij} \leq 1 \tag{7.6e}$$

$$\sum\nolimits_{(i,j)\in\mathcal{F}}[(i,j)\in\mathcal{E}]\beta^0_{ij} + [(i,j)\notin\mathcal{E}]\beta^1_{ij} \leq B \tag{7.6f}$$

We can recover $\boldsymbol{\pi}^*(\boldsymbol{z})_v$ from x^*_v via $\boldsymbol{\pi}^*(\boldsymbol{z})_v = (1-k_v d_v^{-1})x^*_v$, with $k_v = |\{x^{*0}_{vj} \mid x^{*0}_{vj} > 0\}|$.

Key idea and insights. Eq. 7.6b and Eq. 7.6c correspond to the LP of the unconstrained MDP. Intuitively, the variable x_v maps to the PageRank score of node v, and from the variables x^0_{ij}/x^1_{ij} we can recover the optimal policy: if the variable x^0_{ij} (respectively x^1_{ij}) is non-zero then in the optimal policy the fragile edge (i,j) is turned off (respectively on). Since there exists a deterministic optimal policy, only one of them is non-zero but never both. Eq. 7.6d corresponds to the local budget. Remarkably, despite the variables x^0_{ij}/x^1_{ij} not being integral, since they share the factor $x_i d_i^{-1}$ from Eq. 7.6c we can exactly count the number of edges that are turned off or on using only linear constraints. Eq. 7.6e and Eq. 7.6f enforce the global budget. From Eq. 7.6e we have that whenever x^0_{ij} is non-zero it follows that $\beta^1_{ij} = 0$ and $\beta^0_{ij} = 1$ since that is the only configuration that satisfies the constraints (similarly for x^1_{ij}). Intuitively, this effectively makes the $\beta^0_{ij}/\beta^1_{ij}$ variables "counters" and we can utilize them in Eq. 7.6f to enforce the total number of perturbed edges to not exceed B. See detailed proof in Sec. F.3.

(c) Efficient Reformulation Linearization Technique (RLT). The quadratic constraints in our QCLP make the problem non-convex and difficult to solve. We relax the problem using the Reformulation Linearization Technique (RLT) [206] which gives us an upper bound on the objective. The alternative SDP-relaxation [78] based on semidefinite programming is not suitable for our problem since the constraints are trivially satisfied (see Sec. F.4 for details). While in general, the RLT introduces many new variables (replacing each product term $m_i m_j$ with a variable M_{ij}) along with multiple new linear inequality constraints, it turns out that in our case the solution is compact:

Proposition 7.3. *Given fixed upper bounds $\overline{x_v}$ for x_v and using the RLT relaxation, the quadratic constraints in Eq. 7.6e and Eq. 7.6f transform into the single* linear *constraint*

$$\sum\nolimits_{(i,j)\in\mathcal{F}} [(i,j) \in \mathcal{E}] x^0_{ij} d_i (\overline{x_i})^{-1} + [(i,j) \notin \mathcal{E}] x^1_{ij} d_i (\overline{x_i})^{-1} \leq B \tag{7.7}$$

Proof provided in Sec. F.3. By replacing Eq. 7.6e and Eq. 7.6f with Eq. 7.7 in Proposition 7.2, we obtain a linear program which can be efficiently solved. Remarkably, we only have x_v, x^0_{ij}, x^1_{ij} as decision variables since we were able to eliminate all other variables. The solution is an upper bound on the solution for Problem 7.2 and a lower bound on the solution for Problem 7.1. The final relaxed QCLP can also be interpreted as a constrained MPD with a single additional constraint (Eq. 7.7) which admits a possibly randomized optimal policy with at most one randomized state [207].

Certificate for local and global budget. To solve the relaxed QCLP and compute the final certificate we need to provide the upper bounds $\overline{x_v}$ for the constraint in Eq. 7.7. Since the quality of the RLT relaxation depends on the tightness of these upper bounds, we have to carefully select them. We provide here one solution (see Sec. F.6 in the appendix for a faster to compute, but less tight, alternative): Given an instance of Problem 7.2, we can set the reward to $\boldsymbol{r} = \boldsymbol{e}_v$ and invoke Algorithm 7.1, which is highly efficient, using the same fragile set and the same local budget. Since this explicitly maximizes x_v, the objective value of the problem is guaranteed to give a valid upper bound $\overline{x}_v$. Invoking this procedure for every node, leads to the required upper bounds.

Now, to compute the certificate with local and global budget for a target node t, we solve the relaxed problem for all $c \neq y_t$, leading to objective function values $L_{ct} \geq -m^*_{y_t,c}(t)$ (minus due to the change from min to max). Thus, $L_{*,t} = \min_{c\neq y_t} -L_{ct}$ is a lower bound on the worst-case margin $m^*_{y_t,*}(t)$. If the lower bound is positive then node t is guaranteed to be certifiably robust – there exists no adversarial attack (among all graphs in $\mathcal{Q}_\mathcal{F}$) that can change the prediction for node t.

For our policy iteration approach if $m^*_{y_t,*}(t) < 0$ we are guaranteed to have found an adversarial example since the certificate is exact, i.e. we also have a non-robustness certificate. However, in this case, if the lower bound $L_{*,t}$ is negative we do not necessarily have an adversarial example. Instead, we can perturb the graph with the optimal configuration of fragile edges for the relaxed problem, and inspect whether the predictions change. See Fig. 7.1 for an overview of both approaches.

7.5 Robust Training for Graph Neural Networks

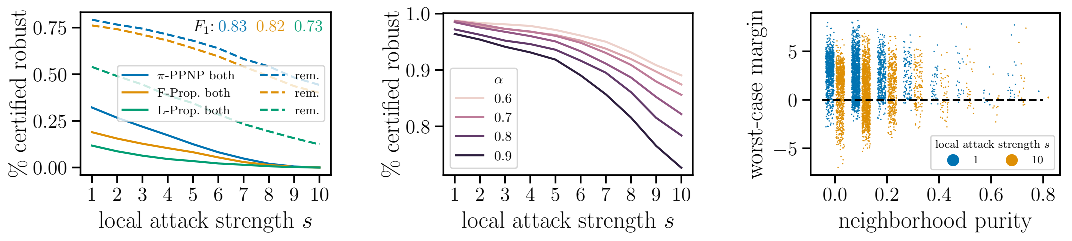
In Sec. 7.4 we introduced two methods to efficiently compute certificates given a trained $\boldsymbol{\pi}$-PPNP model. We now show that these can naturally be used to go one step further – to *improve* the robustness of the model. The main idea is to utilize the worst-case margin during training to encourage the model to learn more robust weights. Optimizing some robust loss $\mathcal{L}_\theta$ with respect to the model parameters θ (e.g. for $\boldsymbol{\pi}$-PNPP θ are the neural network parameters) that depends on the worst-case margin $m^*_{y_v,*}(v)$ is generally hard since it involves an inner optimization problem, namely finding the worst-case margin. This prevents us to easily take the gradient of $m^*_{y_v,c}(v)$ (and, thus, $\mathcal{L}_\theta$) w.r.t. the parameters θ. Previous approaches tackle this challenge by using the dual [182].

Inspecting our problem, however, we see that we can directly compute the gradient. Since $m^*_{y_v,c}(v)$ (respectively the corresponding lower bound) is a linear function of $\boldsymbol{H} = f_\theta(\boldsymbol{X})$ and $\boldsymbol{\pi}_G$, and furthermore the admissible set $\mathcal{Q}_\mathcal{F}$ over which we are optimizing is compact, it follows from Danskin's theorem [208] that we can simply compute the gradient of the loss at the optimal point. We have $\frac{\partial m^*_{y_v,c}(v)}{\partial \boldsymbol{H}_{i,y_v}} = \boldsymbol{\pi}^*(\boldsymbol{e}_v)_i$ and $\frac{\partial m^*_{y_v,c}(v)}{\partial \boldsymbol{H}_{i,c}} = -\boldsymbol{\pi}^*(\boldsymbol{e}_v)_i$, i.e. the gradient equals to the optimal ($\pm$) PageRank scores computed by our certificate.

To improve robustness Wong and Kolter [182] proposed to optimize the *robust cross-entropy loss*: $\mathcal{L}_{\text{RCE}} = \mathcal{L}_{\text{CE}}(y^*_v, -\boldsymbol{m}^*_{y_v}(v))$, where $\mathcal{L}_{\text{CE}}$ is the standard cross-entropy loss operating on the logits, and $\boldsymbol{m}^*_{y_v}(v)$ is a vector such that at index c we have $m^*_{y_v,c}(v)$. Previous work has shown that if the model is overconfident there is a potential issue when using $\mathcal{L}_{\text{RCE}}$ since it encourages high certainty under the worst-case perturbations [123]. Therefore, we also study the alternative *robust hinge loss*. Since the attacker wants to minimize the worst-case margin $m^*_{y_t,*}(t)$ (or its lower bound), a straightforward idea is to try to maximize it during training. To achieve this we add a hinge loss penalty term to the standard cross-entropy loss. Specifically:

$$\mathcal{L}_{\text{CEM}} = \sum\nolimits_{v \in \mathcal{V}_L} \left[\mathcal{L}_{\text{CE}}(y^*_v, \boldsymbol{H}^{\text{diff}}_{v,:}) + \sum\nolimits_{c \in \mathcal{C}, c \neq y^*_v} \max(0, M - m^*_{y^*_v,c}(v)) \right] \tag{7.8}$$

The second term for a single node v is positive if $m^*_{y_v,c}(v) < M$ and zero otherwise – the node v is certifiably robust with a margin of at least M. Effectively, if all training nodes are robust, the second term becomes zero, thus, reducing $\mathcal{L}_{\text{CEM}}$ to the standard cross-entropy loss with robustness guarantees. Note again that we can easily compute the gradient of these losses w.r.t. the (neural network) parameters θ.



(a) Cora-ML, local budget **(b)** Citeseer, impact of α **(c)** Cora-ML, purity

Figure 7.3: Increasing local attack strength s (local budget $b_v = \max(d_v - 11 + s, 0)$) decreases ratio of certified nodes. (a) The graph is more robust to removing edges, $\boldsymbol{\pi}$-PPNP is most robust overall. (b) Lowering the damping factor α improves the robustness. (c) Nodes with higher neighborhood purity are more robust.

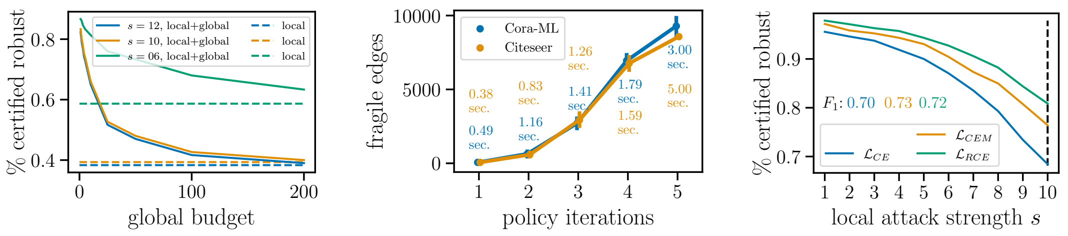
7.6 Experimental Results

Setup. We focus on evaluating the robustness of $\boldsymbol{\pi}$-PPNP and label/feature propagation using our two certification methods. We demonstrate our claims on two publicly available datasets: Cora-ML ($N = 2,995, |\mathcal{E}| = 8,416, D = 2,879, K = 7$) [2, 107] and Citeseer ($N = 3,312, |\mathcal{E}| = 4,715, D = 3,703, K = 6$) [29] with further experiments on Pubmed ($N = 19,717, |\mathcal{E}| = 44,324, D = 500, K = 3$) [29] in the appendix. We configure $\boldsymbol{\pi}$-PPNP with one hidden layer of size 64 and set $\alpha = 0.85$. We select 20 nodes per class for the train/validation set and use the rest for the test set. We compute the certificates w.r.t. the predictions, i.e. we set y_t in $m^*_{y_t,*}(t)$ to the predicted class for node t on the clean graph. See Sec. F.2 for further experiments and Sec. F.7 for more details. Note, we do not need to compare to any previously introduced adversarial attacks on graphs [121–123], since by the definition of a certificate for a certifiably robust node w.r.t. a given admissible set $\mathcal{Q}_\mathcal{F}$ there exist no successful attack within that set.

We construct several different configurations of fixed and fragile edges to gain a better understanding of the robustness of the methods to different kind of adversarial perturbations. Namely, "both" refers to the scenario where $\mathcal{F} = \mathcal{V} \times \mathcal{V}$, i.e. the attacker is allowed to add or remove *any* edge in the graph, while "rem." refers to the scenario where $\mathcal{F} = \mathcal{E}$ for a given graph $G = (\mathcal{V}, \mathcal{E})$, i.e. the attacker can only remove existing edges. In addition, for all scenarios we specify the fixed set as $\mathcal{E}_f = \mathcal{E}_{\text{mst}}$, where $(i, j) \in \mathcal{E}_{\text{mst}}$ if (i, j) belongs to the minimum spanning tree (MST) on the graph G[2].

Robustness certificates: Local budget only. We investigate the robustness of different graphs and semi-supervised node classification methods when the attacker has only local budget constraints. We set the local budget $b_v = \max(d_v - 11 + s, 0)$ relative to the degree d_v of node v in the clean graph, and we vary the *local attack strength* s with lower s leading to a more restrictive budget. Such relative budget is justified since higher degree nodes tend to be more robust in general [121, 143]. We then apply our policy iteration algorithm to compute the (exact) worst-case margin for each node.

[2] Fixing the MST ensures that every node is reachable by every other node for any policy. This simplifies our earlier exposition regarding the MDPs and can be relaxed to e.g. reachable at the optimal policy.



(a) Cora-ML, global budget **(b)** Certificate efficiency **(c)** Citeseer, robust training

Figure 7.4: (a) The global budget can significantly restrict the attacker compared to having only local constraints. (b) Even for large fragile sets Algorithm 7.1 only needs few iterations to find the optimal PageRank. (c) Our robust training successfully increases the percentage of certifiably robust nodes. The increase is largest for the local attack strength that we used during training ($s = 10$, dashed line).

In Fig. 7.3a we see that the number of certifiably robust nodes when the attacker can only remove edges is significantly higher compared to when they can also add edges which is consistent with previous work on adversarial attacks [121]. As expected, the share of robust nodes decreases with higher budget, and $\boldsymbol{\pi}$-PPNP is significantly more robust than label propagation since besides the graph it also takes advantage of the node attributes. Feature propagation has similar performance (F_1 score) but it is less robust. Note that since our certificate is exact, the remaining nodes are certifiably non-robust! In Sec. F.2 in the appendix we also investigate certifiable accuracy – the ratio of nodes that are both certifiably robust and at the same time have a correct prediction. We find that the certifiable accuracy is relatively close to the clean accuracy, and it decreases gracefully as we in increase the budget.

Analyzing influence on robustness. In Fig. 7.3b we see that decreasing the damping factor α is an effective strategy to significantly increase the robustness with no noticeable loss in accuracy (at most 0.5% for any α, not shown). Thus, α provides a useful trade-off between robustness and the size of the effective neighborhood: higher α implies higher PageRank scores (i.e. higher influence) for the neighbors. In general we recommend to set the value as low as the accuracy allows. In Fig. 7.3c we investigate what contributes to certain nodes being more robust than others. We see that neighborhood purity – the share of nodes with the same class in a respective node's two-hop neighborhood – plays an important role. High purity leads to high worst-case margin, which translates to improved certifiable robustness.

Robustness certificates: Local and global budget. We demonstrate our second approach based on the relaxed QCLP problem by analyzing the robustness as we increase the global budget. We set $\mathcal{F} = \mathcal{E}$, i.e. the attacker can only remove edges, and vary the local attack strength s corresponding to local budget $b_v = \max(d_v - 11 + s, 0)$. We see in Fig. 7.4a that by additionally enforcing a global budget we can significantly restrict the success of the attacker compared to having only a local budget (dashed lines). The global constraint increases the number of robust nodes, validating our approach.

Efficiency. Fig. 7.4b demonstrates the efficiency of our approach: even for fragile sets as large as 10^4, Algorithm 7.1 finds the optimal solution in just a few iterations. Since each iteration is itself efficient by utilizing sparse matrix operations, the overall wall-clock runtime (shown as text annotation) is on the order of few seconds. In Sec. F.2 in the appendix, we further investigate the runtime as we increase the number of nodes in the graph, as well as the runtime of our relaxed QCLP.

Robust training. While not being our core focus, we investigate whether robust training improves the certifiable robustness of GNNs. We set the fragile set $\mathcal{F} = \mathcal{E}$ and vary the local budget. The vertical line on Fig. 7.4c indicates the local budget used to train the robust models with losses $\mathcal{L}_{\text{RCE}}$ and $\mathcal{L}_{\text{CEM}}$. We see that both of our approaches are able to improve the percent of certifiably robust nodes, with the largest improvement (around 13% increase) for the local attack strength we trained on ($s = 10$). Furthermore, the F_1 scores on the test split for Citeseer are as follows: 0.70 for $\mathcal{L}_{\text{CE}}$, 0.72 for $\mathcal{L}_{\text{RCE}}$, and 0.73 for $\mathcal{L}_{\text{CEM}}$, i.e. the robust training besides improving the ratio of certified nodes, it also improves the clean predictive accuracy of the model. $\mathcal{L}_{\text{RCE}}$ has a higher certifiable robustness, but $\mathcal{L}_{\text{CEM}}$ has a higher F_1 score. There is room for improvement in how we approach the robust training: e.g. similar to Zügner and Günnemann [143] we can optimize over the worst-case margin for the unlabeled in addition to the labeled nodes. We leave this as a future research direction.

7.7 Conclusion

We derive the first (non-)robustness certificate for graph neural networks regarding perturbations of the graph structure, and the first certificate overall for label/feature propagation. Our certificates are flexible w.r.t. the threat model, can handle both local (per node) and global budgets, and can be efficiently computed. We also propose a robust training procedure that increases the number of certifiably robust nodes while improving the predictive accuracy. One limitation of this certificate is that we do not consider perturbations of the node features and the graph structure jointly. We tackle this limitation (among other things) in Chapter 8.

7.8 Retrospective

Our certificate is "constructive" in the sense that as the solution of Problem 7.1 we obtain not only the worst-case margin but also the set of adversarial edges that achieve it. In retrospective, we can utilize these edges for an adversarial attack. In particular, we could use them to estimate the gap between the certifiably robust and the certifiably non-robust nodes under the global budget constraint by simply testing whether the corresponding perturbed graphs lead to a misclassification when $m^*_{y_t,*}(t) < 0$.[3] This gap is also indicative of the tightness of our lower bound. Moreover, it is likely that the adversarial edges are transferable and can be used to attack other models.

[3]The gap for local budget only is always zero since the certificate is exact.

For problems with binary classes we can extend our approach to derive a non-targeted global attack for label propagation. That is, the goal is to find a single perturbed graph (single set of adversarial edges) that leads to as many misclassified nodes as possible. This is in contrast to the current certificate where the adversarial edges can differ for different target nodes. For a global attack on the GCN model see Zügner and Günnemann [123].

Let $\boldsymbol{y} = \{0,1\}^N$ be the vector of ground-truth binary labels. We configure the unnormalized teleport vector $\tilde{\boldsymbol{z}}$ such that $\tilde{\boldsymbol{z}}_i = \boldsymbol{y}_i$ if node i is in the training set and $\tilde{\boldsymbol{z}}_i = 0.5$ otherwise. The normalized $\boldsymbol{z}$ is obtained by setting $\boldsymbol{z}_i = \tilde{\boldsymbol{z}}_i / c$, where $c = \sum_i \tilde{\boldsymbol{z}}_i$ is the normalization constant. Let $\boldsymbol{\pi}$ be the corresponding PageRank vector, i.e. the solution to $(\boldsymbol{I} - \alpha \boldsymbol{P}) \cdot \boldsymbol{\pi} = (1-\alpha)\boldsymbol{z}$. Then, we predict class 1 for node i if $\tilde{\boldsymbol{\pi}}_i > 0$ and class 0 otherwise, where $\tilde{\boldsymbol{\pi}}_i = \boldsymbol{\pi}_i \cdot c - 0.5$. This prediction is equivalent to Label Propagation [38], and $\sum_i \mathbb{I}[\tilde{\boldsymbol{y}}_i \tilde{\boldsymbol{\pi}}_i > 0]$ equals the number of correctly classified nodes where $\tilde{\boldsymbol{y}}_i = \boldsymbol{y}_i - 0.5$. Therefore, $\min_{\tilde{G} \in \mathcal{Q}_{\mathcal{F}}} \tilde{\boldsymbol{y}}^T \tilde{\boldsymbol{\pi}}_{\tilde{G}}$ is a good proxy for minimizing the accuracy. We can directly apply our proposed algorithms to solve this problem since it boils down to minimizing a linear function of PageRank.

A recent preprint [209] studies how different adversarial immunization strategies affect the certifiable robustness of $\boldsymbol{\pi}$-PPNP. Their preliminary results show that immunizing edges based on betweenness centrality is the strongest baseline compared to cosine similarity between attributes, random choice, and bridgeness.

8 Efficient Robustness Certificates for Discrete Data

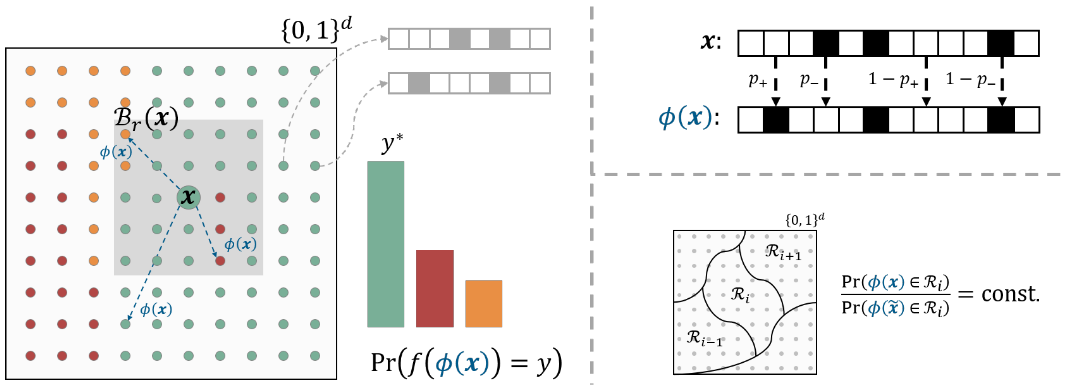


Figure 8.1: The smoothed classifier predicts the majority under random perturbations (left, colors indicate labels). The majority vote changes slowly in a ball around $\boldsymbol{x}$. For binary data the randomization scheme ϕ flips zeros with probability p_+ and ones with p_- (top right). To derive the certificate we partition the input space into regions of constant likelihood ratio (bottom right).

8.1 Introduction

Even a seemingly accurate classifier is of limited use if slight perturbations of the input can lead to misclassification. As we show in Chapter 7 certificates can provide provable guarantees that no perturbation regarding a specific threat model will change the prediction of an instance. However, obtaining meaningful robustness guarantees is challenging since it often involves solving a difficult optimization problem.

An overwhelming majority of certificates in the literature can handle only continuous data. The few approaches that tackle discrete data either work for a small class of models, or stay general while sacrificing efficiency or tightness. In this chapter we propose a general certificate that can be used for any discrete data including sequences (text, audio), discretized images, and molecules. In line with the rest of the thesis we highlight its use for graphs – a particularly important instance of discrete data.

Specifically, we focus on Graph Neural Networks since they are a fundamental building block for many machine learning models today, even though they suffer from poor adversarial robustness [30, 210, 211]. Even in scenarios where adversaries are unlikely, understanding and improving their robustness to worst-case noise is important, especially in safety-critical applications. While some (heuristic) defenses exist [212, 213], we should never assume that the attackers will not be able to break them in the future [214]. Robustness certificates, on the other hand, are by definition unbreakable. Given a clean input $\boldsymbol{x}$ and a perturbation set $\mathcal{B}_r(\boldsymbol{x})$ encoding the threat model (e.g. all inputs within an l_p-ball of radius r centered at $\boldsymbol{x}$) the goal is to verify that the prediction for $\boldsymbol{x}$ and $\forall \tilde{\boldsymbol{x}} \in \mathcal{B}_r(\boldsymbol{x})$ is the same. If this holds, we say that $\boldsymbol{x}$ is certifiably robust w.r.t. $\mathcal{B}_r(\boldsymbol{x})$.

Existing certificates for graphs, including our certificate from Chapter 7, handle either attribute perturbations [215] or structure perturbations [216], but not both, and only work for a small class of models. Furthermore, they are valid only for node-level classification, and extending these techniques to new models and threat scenarios is not straightforward. The approach that we develop in this chapter handles both types of perturbations and applies to any GNN. This includes, for the first time, graph-level classification models for which there are no existing certificates.

We utilize randomized smoothing [217] – a powerful general technique for building certifiably robust models. Inspired by connections to differential privacy [218], this method boils down to randomly perturbing the input and reporting the output/class corresponding to the "majority vote" on the randomized samples. Given any function $f(\cdot)$, e.g. any GNN, we can build a "smoothed" function $g(\cdot)$ that produces a similar output to f (e.g. comparable accuracy if f is a classifier) and for which we can easily provide (probabilistic) robustness guarantees. Importantly, to compute the certificate we need to consider *only* the output of f for each sample. This is precisely what makes it particularly appealing for certifying GNNs since it allows us to sidestep a complex analysis of the message-passing dynamics and the non-linear interactions between the nodes. However, randomized smoothing has some limitations which we discuss in Sec. G.10.

The bulk of the work on randomized smoothing [217–219] focuses on continuous data and guarantees in terms of l_1, l_2 or l_∞ balls, which is not suited for the discrete data domain. Only a few approaches can tackle discrete data with l_0-ball guarantees [220–222]. None of these approaches attempt to certify discrete *graph* data, and there are several major challenges we need to overcome to successfully do so. Jia et al. [223] apply randomized smoothing to only certify the robustness of community detection against structural perturbations. Their certificate also suffers from the same limitations.

The biggest limitation of *all* previous certificates for discrete data is that they rely on randomization schemes that do not take sparsity into account. A common scheme is to randomly flip bits in the input with a given probability p. This is clearly not feasible for graph data due to the sparsity of real-world graphs. Even for a small flip probability (e.g. $p = 0.01$) applying this scheme would introduce too many random edges in the graph, which means that the graph structure is completely destroyed by the random noise, rendering the resulting smoothed classifier useless.[1] On the other hand, p has to be sufficiently high to obtain any guarantees, since higher p's lead to higher certified radii. Similarly, the node attributes are also often sparse vectors, e.g. corresponding to

[1]For example, the Cora-ML dataset has $n = 2810$ nodes, so random sampling introduces $pn^2 = 0.01 \cdot 2810^2 = 78961$ random edges in expectation, i.e. around 28 random edges per node, which is significantly higher than the average node degree of 6.

bag-of-words representations of text, and suffer from the same issue. None of the existing discrete certificates are sparsity-aware. The core idea of our approach is to incorporate sparsity in the randomization scheme by perturbing non-zeros/edges and zeros/non-edges separately in a way that preservers the structure of the data.

Besides the common issue with sparsity, Lee et al. [220]'s and Jia et al. [223]'s certificates are tight but computationally expensive, while Levine and Feizi [221] and Dvijotham et al. [222]'s certificates sacrifice tightness to obtain improved runtime. We overcome these limitations and propose a certificate which is at the same time tight, efficient to compute, and sparsity-aware. In summary, we make contributions on two fronts:

- **GNN Certificates**: Our certificates handle both structure and attribute perturbations and can be applied to any GNN, including graph classification models.
- **Discrete Certificates**: (i) We generalize previous work by explicitly accounting for sparsity; (ii) We obtain tight certificates with a dramatically reduced computational complexity, independent of the input size.

The key observation behind these contributions is that we can partition the space of binary vectors into a *small* number of regions of constant likelihood ratio. The certificate is obtained by traversing these regions and keeping track of the PMF w.r.t. the clean input and the adversarial example. For example, for binary data the number of regions in our partitioning equals the size of the (certified) radius, i.e. grows linearly, and *does not* depend on the input size. This is in stark contrast to previous work where the number of regions is quadratic w.r.t. the input size. Considering that the adjacency matrix of a graph with n nodes has n^2 entries, this reduction in complexity from up to $(n^2)^2 = n^4$ to r regions (where r is the radius) is necessary for feasibility. Furthermore, by drawing connections between our randomization and the Poisson-Binomial distribution for binary data (product of Multinomials for discrete data) we develop an algorithm to efficiently traverse and compute these regions.

8.2 Background and Preliminaries

Let $\boldsymbol{x} \in \mathcal{X} = \{0,1\}^d$ be an observed binary vector. For simplicity we keep the main exposition w.r.t. binary data and we discuss the general discrete case in Sec. 8.5. In Sec. 8.6 we show how to instantiate our framework for GNNs, where $\boldsymbol{x}$ corresponds to the (flattened) adjacency and/or attribute matrix of a graph.

Given a classifier $g(\cdot)$ the goal of the attacker is to find an adversarial example $\tilde{\boldsymbol{x}} \in \mathcal{B}(\boldsymbol{x})$ in the perturbation set such that $\tilde{\boldsymbol{x}}$ is misclassified[2], i.e. $g(\boldsymbol{x}) \neq g(\tilde{\boldsymbol{x}})$, corresponding to an evasion attack. Our goal is to verify whether such an adversarial example exists, i.e. verify whether $g(\boldsymbol{x}) \overset{?}{=} g(\tilde{\boldsymbol{x}})$ for all $\tilde{\boldsymbol{x}} \in \mathcal{B}(\boldsymbol{x})$. We defer all proofs to the appendix (Sec. G.1).

[2]Or classified as some chosen target class other than $g(x)$.

8.2.1 Randomized Smoothing Framework

Let $f : \mathcal{X} \to \mathcal{Y}$ denote a (deterministic or random) function corresponding to a base classifier which takes a vector $\boldsymbol{x} \in \mathcal{X}$ as input and outputs a single class $f(\boldsymbol{x}) = y \in \mathcal{Y}$ with $\mathcal{Y} = \{1, \dots, C\}$. We construct a smoothed classifier $g : \mathcal{X} \to \mathcal{Y}$ from f as follows:

$$g(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \Pr(f(\phi(\boldsymbol{x})) = y) \tag{8.1}$$

where ϕ is a randomization scheme to be specified (e.g. adding Gaussian noise to $\boldsymbol{x}$), which assigns probability mass $\Pr(\phi(\boldsymbol{x}) = \boldsymbol{z})$ for each randomized outcome $\boldsymbol{z} \in \mathcal{X}$. In other words, $g(\boldsymbol{x})$ returns the most likely class (the majority vote) if we first randomly perturb the input $\boldsymbol{x}$ using ϕ and then classify the resulting vector $\phi(\boldsymbol{x})$ with the base classifier f. To simplify notation let $p_y(\boldsymbol{x}) = \Pr(f(\phi(\boldsymbol{x})) = y)$ and $y^* = \arg\max_{y \in \mathcal{Y}} p_y(\boldsymbol{x})$. Let $p^* = p_{y^*}(\boldsymbol{x})$ be the probability of the most likely class. Following Lee et al. [220] we define:

$$\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}(p, y) = \min_{\substack{h \in \mathcal{H}: \\ \Pr(h(\phi(\boldsymbol{x}))=y)=p}} \Pr(h(\phi(\tilde{\boldsymbol{x}})) = y) \tag{8.2}$$

where $\tilde{\boldsymbol{x}} \in \mathcal{X}$ is a given neighboring point, and $\mathcal{H}$ is the set of measurable classifiers with respect to ϕ. We have that $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}(p, y) \leq \Pr(f(\phi(\tilde{\boldsymbol{x}})) = y)$ is a *tight* lower bound on the probability that a neighboring point $\tilde{\boldsymbol{x}}$ is assigned to class y using the smoothed classifier g. The bound is tight in the sense that the base classifier f satisfies the constraint. For an input $\boldsymbol{x}$ and a perturbation set $\mathcal{B}(\boldsymbol{x})$ specifying a threat model (e.g. l_0-ball of radius r) if

$$\min_{\tilde{\boldsymbol{x}} \in \mathcal{B}(\boldsymbol{x})} \rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}(p^*, y^*) > 0.5 \tag{8.3}$$

then we can guarantee that $\Pr(f(\phi(\tilde{\boldsymbol{x}})) = y^*) > 0.5$, for all $\tilde{\boldsymbol{x}} \in \mathcal{B}(\boldsymbol{x})$. This implies that $g(\boldsymbol{x}) = g(\tilde{\boldsymbol{x}}) = y^*$ for any input within the ball, i.e. $\boldsymbol{x}$ is certifiably robust.

Computing $p_y(\boldsymbol{x})$ exactly is difficult, so similar to previous work [217] we compute a lower bound $\underline{p_y(\boldsymbol{x})}$ based on the Clopper-Pearson Bernoulli confidence interval [224] with confidence level α using Monte Carlo samples from $\phi(\cdot)$. Since $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}(p)$ is an increasing function of p [220], a lower bound entails a valid certificate. The certificate is probabilistic and holds with probability $1 - \alpha$. Eq. 8.3 is tight for two classes and provides a sufficient condition to guarantee robustness for more classes ($|\mathcal{Y}| > 2$). In Sec. G.2 we show how to obtain better guarantees for multi-class classification by computing confidence intervals that hold *simultaneously* for all classes using Bonferroni correction.

8.2.2 Solving the Optimization Problem in Eq. 8.2

Assume we can partition $\mathcal{X} = \bigcup_i^I \mathcal{R}_i, \mathcal{R}_i \cap \mathcal{R}_j = \emptyset$ into regions $\mathcal{R}_i$ of constant likelihood ratio, such that for every $\boldsymbol{z} \in \mathcal{R}_i$ it holds $\Pr(\phi(\boldsymbol{x}) = \boldsymbol{z}) / \Pr(\phi(\tilde{\boldsymbol{x}}) = \boldsymbol{z}) = c_i$ for some constant c_i. Then, Eq. 8.2 is equivalent to the following Linear Program (LP) [220]:

$$\min_{\boldsymbol{h}} \boldsymbol{h}^T \tilde{\boldsymbol{r}} \quad \text{s.t.} \quad \boldsymbol{h}^T \boldsymbol{r} = p, \quad 0 \leq \boldsymbol{h} \leq 1 \tag{8.4}$$

where $\boldsymbol{h} \in [0,1]^I$ is the vector we are optimizing over corresponding to the classifier h, and $\boldsymbol{r}$ is a vector where $\boldsymbol{r}_i = \Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_i)$ for each region, and similarly for $\tilde{\boldsymbol{r}}_i$. The exact solution to this LP can be easily obtained with a greedy algorithm: first sort the regions such that $c_1 \geq c_2 \geq \cdots \geq c_I$, then iteratively assign $\boldsymbol{h}_i = 1$ for all regions $\mathcal{R}_i$ until the budget constraint is met (except for the final region which we "consume" partially). See Sec. G.1 for more details. Therefore, how efficiently we can compute the certificate depends on the number of regions and how difficult it is to compute $\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_i)$ for a given $\mathcal{R}_i$ and $\boldsymbol{x}$. This is why reducing the number of regions is crucial.

We show that the optimization problem for the multi-class certificate is also a simple LP and can be exactly solved with a similar greedy algorithm (Sec. G.2). Another interpretation of Eq. 8.2 is that it corresponds to likelihood ratio testing with significance level p between two different hypotheses: $\Pr(\phi(\boldsymbol{x}) = \boldsymbol{z})$ vs. $\Pr(\phi(\tilde{\boldsymbol{x}}) = \boldsymbol{z})$ [225]. We show in Sec. 8.4.3 that given our choice of randomization ϕ, the problem is equivalent to hypothesis testing of two Poisson-Binomial distributions with different parameters.

8.3 Threat Model

We assume that an adversary can perturb $\boldsymbol{x}$ by flipping some of its bits. We define the ball centered at the clean input $\boldsymbol{x}$:

$$\mathcal{B}_{r_a,r_d}(\boldsymbol{x}) = \left\{ \tilde{\boldsymbol{x}} : \tilde{\boldsymbol{x}} \in \mathcal{X}, \sum_{i=1}^{d} \mathbb{I}(\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i - 1) \leq r_d, \sum_{i=1}^{d} \mathbb{I}(\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i + 1) \leq r_a \right\} \tag{8.5}$$

which contains all binary vectors $\tilde{\boldsymbol{x}}$ which can be obtained from $\boldsymbol{x}$ by deleting at most r_d bits (flipping from 1 to 0) and adding at most r_a bits (flipping from 0 to 1). Analogously, we define the sphere $\mathcal{S}_{r_a,r_d}(\boldsymbol{x})$ where the inequalities in Eq. 8.5 are replaced by equalities. The minimum over $\mathcal{B}_{r_a,r_d}(\boldsymbol{x})$ in Eq. 8.3 is always attained at some $\tilde{\boldsymbol{x}} \in \mathcal{S}_{r_a,r_d}(\boldsymbol{x})$.

Intuitively, the radii r_a and r_d control the global budget of the attacker, i.e. the overall number of additions or deletions they can make. This is in contrast to other threat models for binary/graph data which do not distinguish between addition and deletion. Threat models for graphs often specify additional local budget constraints, e.g. at most a given number of perturbations per node. We focus on global constraints which correspond to more powerful attacks. We can also provide l_0-ball guarantees, i.e. to certify w.r.t. $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_0 \leq r$ we can simply certify w.r.t. all balls $\mathcal{B}_{r_a,r_d}(\boldsymbol{x})$ where $r_a + r_d = r$.

8.4 Sparsity-aware Certificate

8.4.1 Data-dependent Sparsity-aware Randomization

We define the noise distribution with two parameters $p_-, p_+ \in [0,1]$ independently per dimension i:

$$\Pr(\phi(\boldsymbol{x})_i \neq \boldsymbol{x}_i) = p_-^{\boldsymbol{x}_i} p_+^{(1-\boldsymbol{x}_i)} \tag{8.6}$$

The randomization scheme ϕ flips the bit $\boldsymbol{x}_i = 1$ to 0 (e.g. deletes an existing edge) with probability p_-, and similarly flips the bit $\boldsymbol{x}_i = 0$ to 1 (e.g. adds a new edge) with

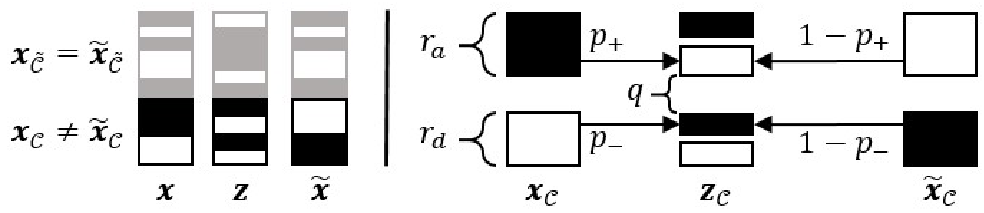


Figure 8.2: The vector $\tilde{\boldsymbol{x}}$ is obtained from $\boldsymbol{x}$ by adding exactly r_a bits and deleting exactly r_d bits. Any vector $\boldsymbol{z}$ in the region $\mathcal{R}_q^{r_a,r_d}$ is obtained by flipping q bits in $\boldsymbol{x}_\mathcal{C}$ and not flipping (retaining) q bits in $\tilde{\boldsymbol{x}}_\mathcal{C}$. Solid boxes denote ones and empty boxes denote zeros.

probability p_+. This allows us to control the amount of smoothing separately for the ones and zeros (edges and non-edges). In other words, the noise distribution is *data-dependent*, which is in contrast to all previous randomized smoothing certificates. Moreover, we say that ϕ is sparsity-aware since often, for real-world data, the number of ones in $\boldsymbol{x}$ is significantly smaller than the number of zeros, i.e. $\|\boldsymbol{x}\|_0 \ll d$. As we will show in Sec. 8.8.2 sparsity-awareness is crucial for obtaining non-trivial certificates. The randomization scheme defined in Lee et al. [220] is a special case which flips the i-th bit $\boldsymbol{x}_i$ with a single probability $p = p_- = p_+$ regardless of its value. For the general discrete case see Sec. 8.5.

8.4.2 Regions of Constant Likelihood Ratio

We can partition $\mathcal{X}$ into a *small* number of regions of constant likelihood which enables us to use the greedy algorithm for solving Eq. 8.2 specified in Sec. 8.2.2 to obtain an efficient certificate. Given any $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}} \in \mathcal{S}_{r_a,r_d}(\boldsymbol{x})$, let $\mathcal{C} = \{i : \boldsymbol{x}_i \neq \tilde{\boldsymbol{x}}_i\}$ be the set of dimensions where $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ disagree, and let $\tilde{\mathcal{C}} = \{1, \dots, d\} \setminus \mathcal{C}$ be its complement. Let $\boldsymbol{x}_\mathcal{C}, \tilde{\boldsymbol{x}}_\mathcal{C} \in \{0,1\}^{|\mathcal{C}|}$ denote the vectors $\boldsymbol{x}, \tilde{\boldsymbol{x}}$ considering only the dimensions specified in $\mathcal{C}$.

We define the region $\mathcal{R}_q^{r_a,r_d}$ containing all binary vectors $\boldsymbol{z}$ which can be obtained by flipping exactly q bits in $\boldsymbol{x}_\mathcal{C}$ and which have *any* configuration of ones and zeros in $\boldsymbol{x}_{\tilde{\mathcal{C}}}$:

$$\mathcal{R}_q^{r_a,r_d} = \{\boldsymbol{z} \in \mathcal{X} : \|\boldsymbol{x}_\mathcal{C} - \boldsymbol{z}_\mathcal{C}\|_0 = q, \|\mathbf{1} - \boldsymbol{x}_\mathcal{C}\|_0 = r_a, \|\boldsymbol{x}_\mathcal{C}\|_0 = r_d\}$$

The region $\mathcal{R}_q^{r_a,r_d}$ contains at the same time all vectors $\boldsymbol{z}$ which can be obtained by retaining (not flipping) q bits in $\tilde{\boldsymbol{x}}_\mathcal{C}$, i.e. $\|\tilde{\boldsymbol{x}}_\mathcal{C} - \boldsymbol{z}_\mathcal{C}\|_0 = r_d + r_a - q$ for all $\boldsymbol{z} \in \mathcal{R}_q^{r_a,r_d}$. To see this, note that from the definition of $\mathcal{S}_{r_a,r_d}(\boldsymbol{x})$, $\tilde{\boldsymbol{x}}_\mathcal{C}$ is the complement to $\boldsymbol{x}_\mathcal{C}$, and we can obtain $\tilde{\boldsymbol{x}}_\mathcal{C}$ from $\boldsymbol{x}_\mathcal{C}$ by flipping exactly r_d bits from 1 to 0, and flipping exactly r_a bits from 0 to 1 (see Fig. 8.2). We can partition $\mathcal{X}$ in exactly $r_a + r_d + 1$ such regions.

Proposition 8.1. *The set* $\{\mathcal{R}_0^{r_a,r_d}, \dots, \mathcal{R}_{r_a+r_d}^{r_a,r_d}\}$ *partitions the entire space of binary vectors* $\mathcal{X}$ *into disjoint regions, i.e.* $\mathcal{X} = \bigcup_{q=0}^{q=r_a+r_d} \mathcal{R}_q^{r_a,r_d}$ *and* $\mathcal{R}_i^{r_a,r_d} \cap \mathcal{R}_j^{r_a,r_d} = \emptyset, \forall i \neq j$.

Since the smoothing is independent per dimension we can restrict our attention only to those dimensions where $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ disagree, otherwise $\Pr(\phi(\boldsymbol{x})_i \neq \boldsymbol{x}_i) = \Pr(\phi(\tilde{\boldsymbol{x}})_i \neq \tilde{\boldsymbol{x}}_i)$ for $i \in \tilde{\mathcal{C}}$ which does not change the ratio c_q for any region $\mathcal{R}_q^{r_a,r_d}$. This implies that the number of regions is independent of the dimension d. Furthermore, by definition $|\mathcal{C}| = r_a + r_d$, thus we can make between 0 and $r_a + r_d$ flips in total counting only w.r.t. the dimensions in $\mathcal{C}$, and any given $\boldsymbol{z}$ vector belongs only to a single region.

8.4.3 Poisson-Binomial View of the Regions

Before we state further results, it is helpful to consider a different view of the randomization scheme ϕ and how it influences the regions. The scheme $\phi(\cdot)$ is equivalent to first drawing a noise sample $\epsilon_i \sim \mathrm{Ber}(p = p_-^{\boldsymbol{x}_i} p_+^{(1-\boldsymbol{x}_i)})$ from a Bernoulli distribution with probability $p = p_-$ if $\boldsymbol{x}_i = 1$ or $p = p_+$ otherwise, and setting $\phi(\boldsymbol{x})_i = \boldsymbol{x}_i \oplus \epsilon_i$, where $\oplus$ is the XOR. ϕ is data-dependent and sparsity-aware since we can specify e.g. a relatively large p_- for the ones and relatively small p_+ for the zeros to avoid introducing too many noisy bits.

Proposition 8.2. *Given any $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{S}_{r_a,r_d}(\boldsymbol{x})$ and any region $\mathcal{R}_q^{r_a,r_d}$, the probability $\phi(\boldsymbol{x})$ to land in the region is* $\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_q^{r_a,r_d}) = \Pr(Q = q)$ *where* $Q \sim \mathrm{PB}([p_+, r_a], [p_-, r_d]) = \mathrm{PB}(\underbrace{p_+, \ldots, p_+}_{r_a \text{ times}}, \underbrace{p_-, \ldots, p_-}_{r_d \text{ times}})$ *is a Poisson-Binomial random variable on* $\{0, \ldots, r_a + r_d\}$.

Intuitively, all vectors $\boldsymbol{z} \in \mathcal{R}_q^{r_a,r_d}$ correspond to observing q "successes" where a "success" is interpreted as successfully flipping the bit of $\boldsymbol{x}_\mathcal{C}$, which happens with probability p_- or p_+. At the same time, "success" is interpreted as retaining (not flipping) the bit of $\tilde{\boldsymbol{x}}_\mathcal{C}$ with probability $(1 - p_-)$ or $(1 - p_+)$. The probability distribution for the number of successes is a sum of d independent, but not identical (since $p_i = p_+$ or $p_j = p_-$) Bernoulli random variables which is a Poisson-Binomial random variable.

Since ϵ_i are independent $\Pr(\phi(\boldsymbol{x}) = \boldsymbol{z}) = \prod_{i \in \tilde{\mathcal{C}}} \Pr(\phi(\boldsymbol{x})_i = \boldsymbol{z}_i) \prod_{j \in \mathcal{C}} \Pr(\phi(\boldsymbol{x})_j = \boldsymbol{z}_j)$. By definition $\mathcal{R}_q^{r_a,r_d}$ contains all vectors $\boldsymbol{z}$ that have any configuration of ones and zeros in $\tilde{\mathcal{C}}$ so when we sum over all $\boldsymbol{z} \in \mathcal{R}_q^{r_a,r_d}$ the first product equals 1. Therefore, we can equivalently consider a sum of only $|\mathcal{C}|$ non-identical Bernoulli random variables which is a Poisson-Binomial random variable, i.e. $\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_q^{r_a,r_d}) = \mathrm{PB}([p_+, r_a], [p_-, r_d])$ is a $|\mathcal{C}|$-dimensional Poisson-Binomial distribution with two groups of distinct probabilities.

Alternatively, Eq. 8.2 can be seen as performing likelihood ratio testing where the two hypotheses correspond to two Poisson-Binomial distributions with different parameters, $\mathrm{PB}([p_+, r_a], [p_-, r_d])$ vs. $\mathrm{PB}([1 - p_-, r_a], [1 - p_+, r_d])$ relating to $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ respectively.

For the special case $p_+ = p_- = p$, the Poisson-Binomial distribution reduces to a standard Binomial distribution, i.e. $Q \sim \mathrm{Bin}(p, r_a + r_d)$. Analogously, for discrete data the probability for $\phi(\boldsymbol{x})$ to land in the respective regions is a Multinomial distribution (see Sec. 8.5 and Sec. G.11). We obtain the same certificate as in Lee et al. [220] for a significantly reduced cost, highlighting that the improved region partitioning is crucial.

Proposition 8.3. *For all $\boldsymbol{z} \in \mathcal{R}_q^{r_a,r_d}$, the likelihood ratio is*

$$\eta_q^{r_a,r_d} = \frac{\Pr(\phi(\boldsymbol{x}) = \boldsymbol{z})}{\Pr(\phi(\tilde{\boldsymbol{x}}) = \boldsymbol{z})} = \left[\frac{p_+}{1 - p_-}\right]^{q - r_d} \left[\frac{p_-}{1 - p_+}\right]^{q - r_a}$$

and is constant in the region $\mathcal{R}_q^{r_a,r_d}$. Moreover, for a fixed r_a and r_d, the ratio $\eta_q^{r_a,r_d}$ is a monotonically decreasing function of q if $(p_- + p_+) < 1$, constant if $(p_- + p_+) = 1$, or monotonically increasing function of q if $(p_- + p_+) > 1$.

Linear number of regions. With Proposition 8.1 and Proposition 8.3 we can partition $\mathcal{X}$ into exactly $(r_a + r_d + 1)$ number of regions with constant likelihood ratio. The number of regions grows *linearly* with the radii. Crucially, this implies that the number of regions is *independent* of the input size d. In the special case when $p_- = 0$ and $p_+ > 0$ (or similarly $p_- > 0$ and $p_+ = 0$) there are only three non-empty regions (see Sec. G.3).

Data (size) independence. From Proposition 8.2 and Proposition 8.3 it follows that the value of $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}(p, y)$, and hence the certificate, is exactly the same for any $p, y, \boldsymbol{x}$ and $\tilde{\boldsymbol{x}} \in \mathcal{S}_{r_a,r_d}(\boldsymbol{x})$. In other words, as long as $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ differ in exactly r_d zeros and r_a ones, the solution to Eq. 8.3 is the same. Moreover, the certificate does not depend on the configuration of ones and zeros in the dimensions $\tilde{\mathcal{C}}$ where $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ agree since neither the probability $\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_q^{r_a,r_d})$ nor the ratio $\eta_q^{r_a,r_d}$ depend on the values of $\boldsymbol{x}_i$ and $\tilde{\boldsymbol{x}}_i$ for $i \in \tilde{\mathcal{C}}$. Altogether this means that w.l.o.g. we can compute the certificate based on the following two canonical vectors: $\boldsymbol{x}_{\text{ca}} = (1, \dots, 1, 0, \dots, 0)$ and $\tilde{\boldsymbol{x}}_{\text{ca}} = (0, \dots, 0, 1, \dots, 1)$, where $\|\boldsymbol{x}_{\text{ca}}\|_0 = r_d$ and $\|\tilde{\boldsymbol{x}}_{\text{ca}}\|_0 = r_a$. If several inputs have the same $\underline{p_y(\boldsymbol{x})}$, which holds in practice, we only need to compute the certificate once to certify all of them.

No sorting. Since $\eta_q^{r_a,r_d}$ is monotonic in q we do not need to construct all regions in advance and afterwards sort them in a decreasing order. We can completely avoid the sorting required for the greedy algorithm outlined in Sec. 8.2.2 and directly visit the regions one by one, increasing q (or decreasing when $p_+ + p_- > 1$) each time until we reach $\underline{p_y(\boldsymbol{x})}$. For more details and pseudo-code see Sec. G.4.

8.4.4 Efficiently Computing the Regions

Since $\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_q^{r_a,r_d}) = \text{PB}(q; [p_+, r_a], [p_-, r_d])$ we need to compute the PMF of a Poisson-Binomial distribution. If done naively we need to sum $r!/[r!(r-q)!]$ terms where $r = r_a + r_d$. Fortunately, there is a recursive formula that requires only $\mathcal{O}(qr)$ operations [226]. Since we only have two distinct flip probabilities we can further simplify to obtain:

$$T_{r_a,r_d}(i) = r_a \cdot (p_+/(1-p_+))^i + r_d \cdot (p_-(1-p_-))^i$$

$$R_{r_a,r_d}(q) = \frac{1}{q} \sum_{i=1}^{q} (-1)^{i+1} \cdot T_{r_a,r_d}(i) \cdot R_{r_a,r_d}(q-i)$$

Now $\text{PB}(q; \cdot) = R_{r_a,r_d}(q) \cdot (1-p_+)^{r_a} \cdot (1-p_-)^{r_d}$. To avoid unnecessary computations we additionally unroll the recursion with dynamic programming.[3] Compared to previous discrete certificates [220, 221] we do not need to compute Binomial coefficients.

8.5 General Certificate for Discrete Data

Let $\boldsymbol{x} \in \mathcal{X}_K = \{0, \dots, K-1\}^d$ be a d-dimensional vector where each $\boldsymbol{x}_i$ belongs to one of K different categories. We define the sparsity-aware randomization scheme $\phi(\cdot)$:

$$\Pr(\phi(\boldsymbol{x}_i) = k) = \begin{cases} [\frac{p_+}{k-1}]^{(\boldsymbol{x}_i \neq k)} (1-p_+)^{(\boldsymbol{x}_i = k)}, & \boldsymbol{x}_i = 0 \\ [\frac{p_-}{k-1}]^{(\boldsymbol{x}_i \neq k)} (1-p_-)^{(\boldsymbol{x}_i = k)}, & \boldsymbol{x}_i \neq 0 \end{cases}$$

[3]For multiple-precision arithmetic we use the gmpy2 library: `https://pypi.org/project/gmpy2/`. An alternative approach is to compute the PMF via the Discrete Fourier Transform [227].

That is, we flip zeros with probability p_+, and non-zeros with probability p_-, uniformly to any of the other values. For the special case $p_+ = p_-$ we recover the certificate from Lee et al. [220], and for $K = 2$ we recover our certificate for binary data.

As before, we can partition $\mathcal{X}_K$ into disjoint regions of constant likelihood ratio and efficiently solve the problem defined in Eq. 8.2. We show that the number of regions does not depend on the number of discrete categories K or the dimension of the input d. Specifically, for $p_+ = p_-$ we have exactly $2r + 1$ regions where r is the certified radius, i.e. $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_0 = r$. For $p_+ \neq p_-$ the number of regions is upper bounded by $(r + 1)^2$. Here the key insight is that again $\Pr(\phi(\boldsymbol{x})_i \neq \boldsymbol{x}_i) = \Pr(\phi(\tilde{\boldsymbol{x}})_i \neq \tilde{\boldsymbol{x}}_i)$ if $\boldsymbol{x}_i = \tilde{\boldsymbol{x}}_i$ so w.l.o.g. we can consider only the dimensions where $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ disagree. For a detailed analysis of the regions and how to efficiently compute them see Sec. G.11 in the appendix.

8.5.1 Comparison with Existing Discrete Certificates

There are up to $(d + 1)^2$ non-empty regions for the partitioning in Lee et al. [220], i.e. quadratic w.r.t. input size. Since their certificate is a special case ($p_+ = p_-$) our partitioning provides a dramatic reduction of complexity. For example, to certify perturbations to the binary adjacency matrix where $d = n^2$ we have to traverse up to $\mathcal{O}(n^4)$ regions which is infeasible even for small graphs. With our certificate we have to examine at most $r_a + r_d + 1$ regions regardless of the graph size. Beyond this, in Sec. 8.8.2 we show that our sparsity-aware randomization yields a higher certified ratio. Other certificates for discrete data which are based on f-divergences [222] or randomized ablation [221] sacrifice tightness to gain computational efficiency and provide looser guarantees.

8.6 Instantiating the Certificate for GNNs

Let $G = (\mathcal{V}, \mathcal{E})$ be an attributed graph with $n = |\mathcal{V}|$ nodes. We denote with $\boldsymbol{A} \in \{0, 1\}^{n \times n}$ the adjacency matrix and $\boldsymbol{F} \in \{0, 1\}^{n \times m}$ the matrix of m-dimensional binary features for each node. We consider three different scenarios: (i) the adversary can only perturb the graph structure: $\boldsymbol{x} = \mathrm{vec}(\boldsymbol{A})$, (ii) only the node attributes: $\boldsymbol{x} = \mathrm{vec}(\boldsymbol{F})$, (iii) or both: $\boldsymbol{x} = [\mathrm{vec}(\boldsymbol{A}), \mathrm{vec}(\boldsymbol{F})]$. Here $\mathrm{vec}(\cdot)$ "flattens" a matrix into a vector, and $[\cdot, \cdot]$ denotes concatenation. When the graph is undirected, $\mathrm{vec}(\boldsymbol{A})$ considers only the lower (or upper)-triangular part of $\boldsymbol{A}$. The base classifier $f(\cdot)$ can be any GNN. Perturbing a single given graph can potentially change the predictions for *all* nodes. To certify a given target node t we simply focus on its own predictions (its own distribution over node-level classes) which in general could be computed based on the entire graph. Note, under our threat model we can apply the perturbation anywhere in the graph/features, e.g. including the neighbors of node t. Here we focus on node classification, however, our certificate can be trivially extended to graph-level classification (see Sec. G.8).

Joint certificates. When perturbing both the graph structure and the node attributes, if we set $\boldsymbol{x} = [\mathrm{vec}(\boldsymbol{A}), \mathrm{vec}(\boldsymbol{F})]$ we have to share a single set of radii (r_a, r_d) and flip probabilities (p_+, p_-) for both $\boldsymbol{A}$ and $\boldsymbol{F}$. However, it may be beneficial to specify different flip probabilities/radii. To achieve this we first independently calculate the set of regions

$\mathcal{R}^{\boldsymbol{A}} = \{\dots, \mathcal{R}_q^{r_a^{\boldsymbol{A}}, r_d^{\boldsymbol{A}}}, \dots\}$ and $\mathcal{R}^{\boldsymbol{F}} = \{\dots, \mathcal{R}_q^{r_a^{\boldsymbol{F}}, r_d^{\boldsymbol{F}}}, \dots\}$for $\boldsymbol{x} = \operatorname{vec}(\boldsymbol{A})$ and $\boldsymbol{x} = \operatorname{vec}(\boldsymbol{F})$ respectively using different $(r_a^{\boldsymbol{A}}, r_d^{\boldsymbol{A}}, r_a^{\boldsymbol{F}}, r_d^{\boldsymbol{F}})$. Then we form the regions $\mathcal{R}_{q,q'}$, where

$$\Pr\left(\phi(\boldsymbol{x}) \in \mathcal{R}_{q,q'}\right) = \Pr\left(\phi(\boldsymbol{x}) \in \mathcal{R}_q^{r_a^{\boldsymbol{A}}, r_d^{\boldsymbol{A}}}\right) \cdot \Pr\left(\phi(\boldsymbol{x}) \in \mathcal{R}_{q'}^{r_a^{\boldsymbol{F}}, r_d^{\boldsymbol{F}}}\right)$$

The total number of $\mathcal{R}_{q,q'}$ regions is thus $(r_a^{\boldsymbol{A}} + r_d^{\boldsymbol{A}} + 1)(r_a^{\boldsymbol{F}} + r_d^{\boldsymbol{F}} + 1)$. Therefore, we pay only a small price in terms of complexity for the flexibility of specifying different radii. The size of the balls we can certify in practice is relatively small, e.g. the four radii are typically below 100 so the certificate is feasible. Note that this can be trivially extended to certify arbitrary groupings of $\boldsymbol{x}$ into subspaces with different radii/flip probabilities per subspace. However, the complexity quickly increases. For more details see Sec. G.5.

8.6.1 Comparison with Existing Certificates for GNNs

There are only few certificates for GNNs: Zügner and Günnemann [215] can only handle attribute attacks, while Bojchevski and Günnemann [5] and Zügner and Günnemann [216] only handle graph attacks. All three certificates apply only to node classification and a small class of models. Since their certificates hold for certain (base) classifiers, e.g. GCN [19] or PPNP [10], which tend to be less robust than their smoothed counterparts, we cannot make a fair comparison. Moreover, they rely on local budget constraints (at most a given number of perturbations per node), and provide looser guarantees when using global budget only (since e.g. the global budget certificate for PPNP is NP-Hard). Nonetheless, we compare our certificate with these approaches in the appendix, and show that it provides comparable or better guarantees (see Sec. G.6). Jia et al. [223]'s certificate is neither sparsity-aware nor efficient, and does not apply to GNNs.

8.7 Training

Our certificates hold regardless of how the base classifier f is trained. However, in order to classify the labeled example $(\boldsymbol{x}, y)$ correctly and robustly, g needs to consistently classify the noisy $\phi(\boldsymbol{x})$ as y. To ensure this, similar to previous work [217], we train the base classifier with perturbed inputs, that is we apply $\phi(\cdot)$ during training which is akin to data augmentation with noise. We also investigated the approach suggested by Salman et al. [228], where one directly trains the smoothed classifier g, rather than f. When the base classifier f is a GNN and the task is node-level classification, unlike Salman et al. [228] we did not observe performance improvements with this strategy (see Sec. G.7).

Adversarial training. Even though adversarial training [229, 230] is a *heuristic* defense adding adversarial examples during training tends to also improve the *certifiable* robustness [215, 231]. This has also been demonstrated for smoothed classifiers [228], especially given access to additional unlabeled data [232]. However, adversarial training is useful only with a sufficiently powerful attack. For continuous data we can maximize the loss w.r.t. $\boldsymbol{x}$ via projected gradient descent to find an adversarial example, but this is not well suited for discrete data [30]. We leave it as future work to develop suitable techniques for finding adversarial examples of g so we can employ adversarial training.

8.8 Experimental Evaluation

Our goal is to answer the following questions: (i) What are the trade-offs for different flip probabilities? (ii) What is the benefit of sparsity-awareness? (iii) How robust are different GNNs for different threat models? (iv) How large is the efficiency gain?

8.8.1 Graph Neural Networks

Setup. We evaluate the certifiable robustness of three GNNs: GCN [19], GAT [233] and APPNP [10]. We focus on the node classification task and the three scenarios we outlined in Sec. 8.6. We demonstrate our claims on two datasets: Cora-ML ($n = 2995, e = 8416$) and PubMed ($n = 19717, e = 44324$) [234]. The graphs are sparse, i.e. their number of edges $e \ll n^2$. See Sec. A.1 for further details about the data. For all experiments we set the confidence level $\alpha = 0.01$ and the number of samples for certification to 10^6 (10^5 for MNIST and ImageNet). For hyperparameters and implementation details see Sec. G.9.

In Fig. 8.3 we show the *certified ratio* w.r.t. attribute perturbations for GCN on Cora-ML, i.e. the ratio of nodes which can be certified given the provided radii. The heatmap shows the trade-offs for certifying addition vs. deletion for $p_+ = 0.01, p_- = 0.6$. Since p_- is significantly higher we can certify a larger r_d radius. To ensure the model is robust to a few *worst-case* deletions, we need to ensure it is robust to many randomly deleted bits. The contour lines show the radii for which the certified ratio is at least 0.3 (0.5), i.e. at least 30 % (50 %) of all nodes can be certified.

We repeat the experiment and compare the binary-class certificate $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}$ vs. the multi-class certificate $\mu_{\boldsymbol{x},\tilde{\boldsymbol{x}}}$. Fig. 8.4 shows that the multi-class certificate is better, i.e. achieves a higher certified ratio for the majority of (smaller) radii, while the binary-class certificate performs better for higher radii. In general, the absolute difference is relatively small, with the multi-class certificate being better by 0.012 on average across different radii.

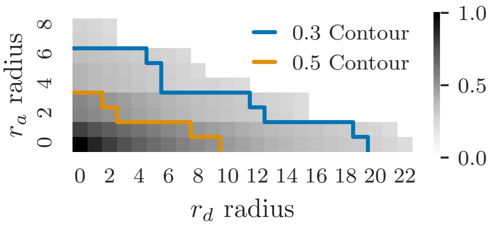


Figure 8.3: Certifying attribute perturbations for GCN on Cora-ML. We show the ratio of certified nodes (darker cells correspond to higher ratio) for different radii for $p_+ = 0.01, p_- = 0.6$. We also show the contour lines under which at least 30 % (and 50 %) of all nodes are certifiably robust.

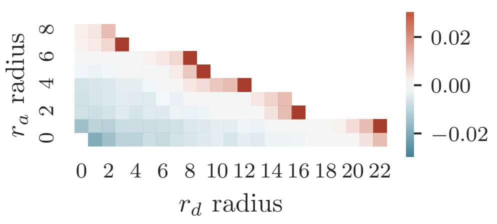


Figure 8.4: Binary-class vs. multi-class certificate comparison. Cells with blue (red) colors show the radii for which the multi-class (respectively binary-class) certificate obtains a higher certificated ratio. The darkest red cells in the corners exceed the color map and have value of around 0.15. Same setup as in Fig. 8.3.

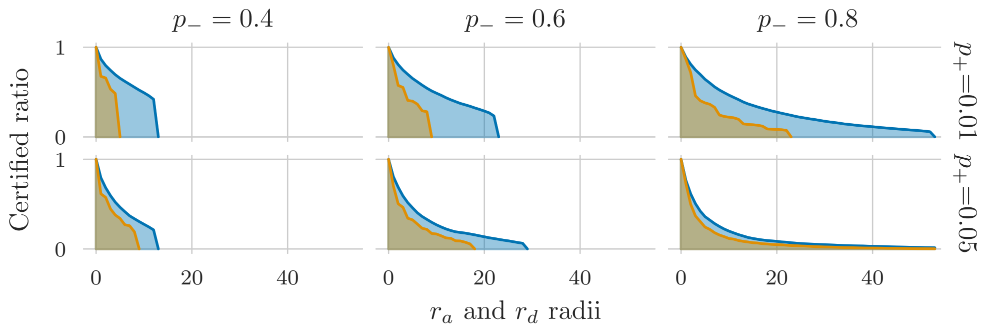
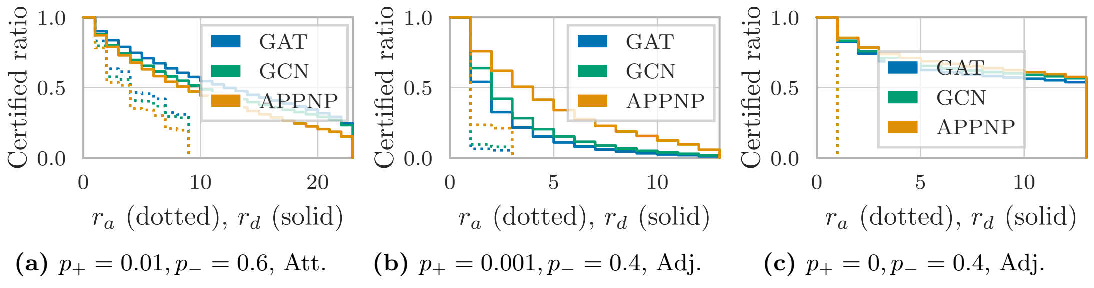


Figure 8.5: Trade-offs for different configurations of flip probabilities. We show the x and y-axis of the heatmap on Fig. 8.3 for different flip probabilities, i.e. $r_a = 0$, r_d varies (blue histogram) and $r_d = 0$, r_a varies (orange histogram) respectively.

In Fig. 8.5 we investigate the trade-offs for different degrees of smoothing. The y-axis shows the ratio of certified nodes. By decreasing the flip probabilities we can certify a larger portion of nodes but at lower radii, while increasing the probabilities allows for larger certified radii overall at the price of smaller ratios. This implies that in practice we can choose a suitable smoothing degree depending on the threat model since the difference in clean accuracy is at most 2 % for all cases (see Sec. 8.8.1).

In Fig. 8.6 we compare the ratio of certified nodes for different GNNs and threat models. We can see that when perturbing the attributes (Fig. 8.6a) GAT is more robust than GCN and APPNP. On the other hand when perturbing the graph structure (Fig. 8.6b) the order is inverted, now APPNP is more robust than GCN and GAT. This highlights that different models have different robustness trade-offs. Furthermore, certifying the attributes is in general easier compared to certifying the graph structure and edge addition is most challenging. Intuitively, since most nodes have a low degree (e.g. average degree on Cora-ML is 6) the attacker can easily misclassify them by adding a few edges.

Interestingly, if we consider the special case where ϕ only deletes edges (by setting $p_+ = 0$) the certified ratio for r_d is significantly improved (Fig. 8.6c). In practice, the observed graph $\boldsymbol{x}$ might already be corrupted. The certificate verifies that all $\tilde{\boldsymbol{x}}$ in the ball, including the unobserved clean graph, have the same prediction. From this point of



(a) $p_+ = 0.01, p_- = 0.6$, Att. **(b)** $p_+ = 0.001, p_- = 0.4$, Adj. **(c)** $p_+ = 0, p_- = 0.4$, Adj.

Figure 8.6: Certifiable robustness for different models. Solid lines denote r_d (with $r_a = 0$) and dotted lines denote r_a (with $r_d = 0$).

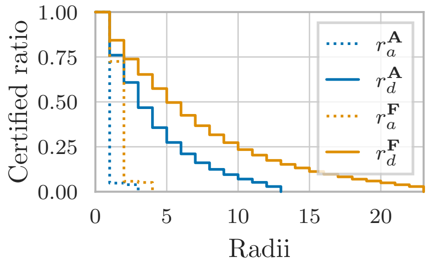


Figure 8.7: Certifying joint perturbations to the graph and attributes on Cora-ML.

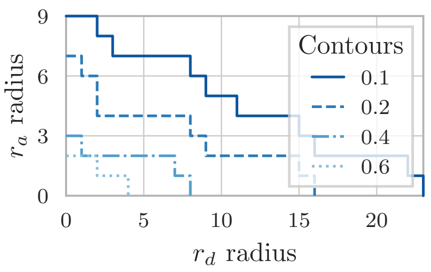


Figure 8.8: Certifying attribute perturbations on PubMed, $p_+ = 0.01, p_- = 0.6$.

view, by randomly deleting edges we are reducing the influence of adversarial edges which were potentially added. Since for many applications it is more feasible for the attacker to add rather than remove edges, certifying r_d is exactly the goal. In general, we see that none of the graph models are really robust, especially w.r.t. structure perturbations.

Next, in Fig. 8.7 we show our method's ability to certify robustness against *joint* perturbations to both the graph structure and the node attributes. We set $p_+^{\boldsymbol{A}} = 2 \cdot 10^{-5}$, $p_-^{\boldsymbol{A}} = 0.4$ for the graph, and $p_+^{\boldsymbol{F}} = 2 \cdot 10^{-5}$, $p_-^{\boldsymbol{F}} = 0.6$ for the attributes. This combined scenario yields slightly worse certificates compared to perturbations w.r.t. one input. Similar to before, we observe that certificates w.r.t. addition are especially hard to obtain.

Efficiency. The overall runtime to compute our certificate for *all* test nodes from the Cora-ML dataset using a GCN model is less than 25 minutes, or around 0.54 seconds per node. Most of the time is spent on $\underline{p_y(\boldsymbol{x})}$ and can be trivially reduced. Finally, to demonstrate that our certificates scales to large graphs we certify w.r.t. the attributes on the PubMed dataset which has over 19.5k nodes (results shown in Fig. 8.8).

Sparsity. Sparsity is crucial when certifying graphs. To show this we certify the attributes and set $p_+ = p_- = 0.1$ since $p_+ = 0.1$ is the largest value such that the clean accuracy is still reasonably high. We further compare with the randomized ablation certificate by Levine and Feizi [221] which also does not consider sparsity. Their certificate depends on the number of retained pixels k, or in our case retained entries of the feature

Table 8.1: Maximum certified radius averaged across nodes for attribute perturbations on GCN. The certificates from [220] and [221] are not sparsity-aware.

	$\overline{r_d}$	$\overline{r_a}$
$k = 0.2d$ [221]	2.01	2.01
$p_+ = p_- = 0.1$ [220]	2.03	2.03
$p_+ = 0.01$, $p_- = 0.6$	9.99	3.38
$p_+ = 0.01$, $p_- = 0.8$	12.65	4.94
$p_+ = 0.00$, $p_- = 0.8$	18.66	2.14

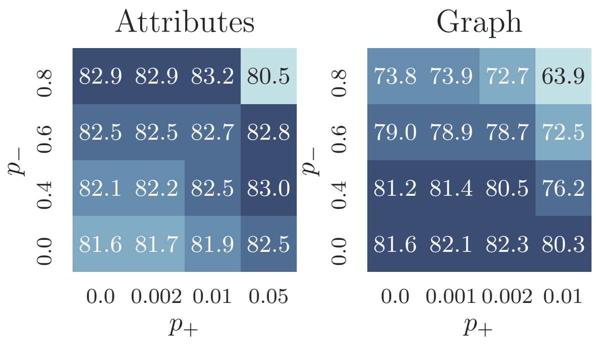


Figure 8.9: Accuracy for different flip probabilities.

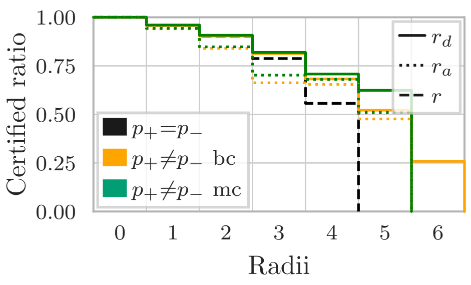


Figure 8.10: The benefit of sparsity awareness on binarized MNIST.

Table 8.2: Certified accuracy for different radii on ImageNet. We show only the time to compute the certificate given $p_y(\boldsymbol{x})$ since $\phi(\cdot)$ is the same for all certificates. The numbers for the baselines are from the respective papers.

Cert.	Time	$r=1$	$r=3$	$r=5$	$r=7$
[222]	28 ms	0.36	0.22	0.14	0
[220]	4 days	0.54	0.34	0.24	0.18
Ours	2.5 ms	0.54	0.34	0.24	0.18

(adjacency) matrix. There is an inherent trade-off: a lower value of k equals a higher certified radius but worse classification accuracy. We set $k = 0.2d$ to the lowest value that still maintains reasonable accuracy. For all certificates we compute the maximum certified radius averaged across all nodes which we denote with $\bar{r}$. We can see in Table 8.1 that our sparsity-aware certificate is significantly better. The performance gap widens even further for graph perturbations. We can conclude that sparsity-awareness is essential.

Clean accuracy. Next, we investigate the clean accuracy for different configurations of smoothing probabilities. In general, we should set the flip probabilities to be as high as possible while maintaining good clean accuracy for the smoothed classifier. On Fig. 8.9 we show the accuracy averaged across 10 different random train/validation/test splits for a GCN base classifier on Cora-ML. Interestingly, when perturbing the attributes increasing p_- and p_+ improves over the accuracy of the base classifier (bottom-left corner, $p_- = 0, p_+ = 0$). We can interpret the perturbation as dropout (except applied during both training and evaluation) which has been shown to improve performance [10, 233]. Similar to the conclusions in previous experiments we see that the graph structure is more sensitive to perturbations and the accuracy decreases as we increase the flip probabilities.

8.8.2 Discretized Images

To show the general applicability of our method and the importance of sparsity and efficiency we certify a CNN model on discretized images (see Sec. G.9 for details).

Sparsity. In Fig. 8.10 we compare our certificate with Lee et al. [220] on binarized MNIST images. Since they have a single radius ($r_a = r_d$) we compare our radii by setting $r_d = 0$ and varying $r_a \geq 0$ (and similarly for $r_d \geq 0$). Their certificate is not sparsity-aware and is a special case of ours (we set $p_+ = p_- = 0.2$). For our certificates we can specify different flip probabilities (we set $p_+ = 0.1, p_- = 0.2$) which results in a significant increase in the certified ratio w.r.t. r_d and matching ratio w.r.t. r_a. We also compare our binary-class (bc) with our multi-class (mc) certificate and we see that the tighter multi-class certificate tends to provide better guarantees.

Efficiency. In Table 8.2 we show the certified accuracy for discretized ImageNet data ($K = 256$) and $p_+ = p_- = 0.8$. We see that our certificate matches Lee et al. [220]'s but at a dramatically improved runtime, from 4 days to under a second. Dvijotham et al.

[222]'s certificate is efficient at the expense of tightness and obtains worse guarantees. Even though $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}$ can be precomputed once and reused for different test inputs, without our improvement it would still be infeasible if d is slightly larger or varies (e.g. sequences).

8.9 Related Work

GNNs are a fundamental part of the modern machine learning landscape and have been successfully used for a variety of tasks from node-level classification [19, 233, 235] to graph-level classification and regression [24, 236] across many domains. However, GNNs are highly sensitive to small adversarial perturbations [3, 30, 210, 211] – a common phenomenon observed for machine learning models in general [16, 43].

Beyond heuristic defenses [212, 213, 229, 230], which can be easily broken in practice [237], certifiable robustness techniques provide provable guarantees [231, 238, 239]. Most certificates either have scalability issues or rely on conservative relaxations. In contrast, the recently proposed randomized smoothing technique [217–220] is a general approach which is relatively inexpensive, yet provides good (probabilistic) guarantees.

Most work on randomized smoothing focuses on continuous data with a few exceptions that can tackle binary/discrete data. In contrast to our approach, these certificates are not sparsity-aware and are either computationally intractable or provide loose guarantees (see Sec. 8.5.1). Moreover, we are the first to apply randomized smoothing to GNNs. There are only few certificates for graphs [5, 215, 216] and as we discussed in Sec. 8.1 and in Sec. 8.6.1 they have serious limitations that we overcome.

8.10 Conclusion

We propose the first sparsity-aware certificate for discrete data based on the randomized smoothing framework. Our certificate can be efficiently computed and the complexity does not depend on the input size or the number of discrete categories. The sparsity-awareness and the drastically improved efficiency significantly broaden its applicability compared to previous work. We apply our certificate to study the robustness of different Graph Neural Networks and show that there are clear trade-offs across GNNs models.

8.11 Retrospective

This retrospective is limited since we finished this work close to the thesis submission, nonetheless we make some observations. First, it is interesting that smoothing significantly improves the clean accuracy for graphs (Fig. 8.9), especially since this does not hold for images [240, 241]. Why do we not observe a large trade-off between clean accuracy and adversarial robustness as we often see for other methods? The relations to local Lipschitzness [242] and data augmentation [243] may provide some answers. Second, we did not fully exploit the generality of the certificate, and the focus on GNNs was mostly for historical reasons. Since the certificate is directly applicable to e.g. text data, additional experiments with NLP models can further increase the impact of the publication.

Finally, it is worth highlighting that we can discretize continuous data to arbitrary precision since the computational complexity is independent of the number of categories, greatly increasing the applicability.[4] In Lee et al. [220] the certificate for discretized images even outperformed its continuous counterpart.

[4]The caveat is that continuous data is implicitly "ordered", while the discrete categories are not.

Part V
Concluding Remarks

9 Conclusions and Outlook

9.1 Summary

In this thesis we study the impact of noise and adversarial perturbations on three types of machine learning models for graph data: unsupervised, generative, and semi-supervised. We design adversarial attacks to investigate the lack of robustness, we develop robust models which are resilient to corruptions, and we derive provable robustness guarantees. In each case we show the importance of robustness for ensuring that our models are reliable. Along the way we tackle recurring challenges such as handling the non i.i.d. nature of graph data and optimizing over a discrete and combinatorial graph domain.

First, in Part II we tackle unsupervised representation learning. We discuss two different notions of robustness for node embeddings. Namely, in Chapter 3 we assume the data is corrupted and the goal is to derive a robust spectral embedding by neutralizing the corruptions. We achieve this by jointly learning the corruptions and the embedding of the latent clean graph. In Chapter 4 we take a different approach and we achieve robustness by accounting for uncertainty. Each node is represented as a Gaussian distribution with some mean and a covariance matrix. Rather than explicitly removing corruptions, their effects can be absorbed by increasing the uncertainty of the representation (i.e. increasing the variance). In Chapter 5, taking the opposite role, we initiate the first study of adversarial perturbations for node embeddings. We show that node embeddings are highly sensitive to adversarial attacks by analyzing a family of methods based on random walks. The adversarial perturbations can be efficiently computed, and they generalize to other (unsupervised and semi-supervised) models.

Next, In Part III we design a robust generative model for graphs and a scalable inference algorithm. The key idea is to explicitly consider partial anomalies in the generative process. The main benefit is that if a node is only partially corrupt we can still infer something meaningful about it (e.g. its cluster), rather than discarding it as an anomaly.

Finally, in Part IV we study semi-supervised learning focusing on certifiable robustness. The goal is to provide provable guarantees. In Chapter 7 we tackle a specific class of models – which includes label propagation and some graph neural networks – where the predictions are a linear function of (personalized) PageRank. The main benefit of specializing to this class of models it that we can compute the certificate exactly (assuming a local budget for the attacker) by taking advantage of the structure of the problem. In Chapter 8 we instead derive a certificate which is model-agnostic, i.e. it does not rely on any assumptions about the model. With this approach we can certify any graph-based classifier (or in fact any classifier by discretizing the input), however, this generality comes at a price since we can only compute a lower bound on the true certified radius. This bound is tight and cannot be improved unless we make additional assumptions.

9.2 Concluding Retrospective

Collectively looking at the different approaches in this thesis we observe an interesting general principle: "joint or shared is better than separate". In Chapter 3 we argue for jointly learning the corruptions and the embedding. In Chapter 4 we show that sharing the learnable parameters that generate the embeddings is effective and has the benefit of being inductive. Similarly, as we discuss in Sec. 6.7, learning a joint inference model for the variational parameters often performs better in practice. The PPNP model is a notable exception, where the separation into two phases (predict then propagate) enables superior scalability [8] and efficient certificates (Chapter 7).

Another observation is that proper evaluation is both more difficult and more essential than it seems. In Shchur et al. [11] we examine the pitfalls of graph neural network evaluation. We show that using fixed train/validation/test splits and differences in the training procedure (e.g. early stopping criteria) lead to flawed comparisons. Our findings show that simpler GNNs outperform more sophisticated ones if the hyperparameters and the training procedure are fairly tuned for all models. More generally, it is critical to include and tune all of the relevant baselines. For example, in Sec. 4.4.1 we show that a simple logistic regression has a surprisingly strong link prediction performance, even outperforming some of the more advanced methods. Due to sparsity, link prediction is inherently difficult to evaluate [244] and using the wrong evaluation metric might be misleading. Therefore, we spent a lot of time thinking about the design of a fair evaluation scheme.[1] Questions on how to choose the number of labeled nodes[2], or whether we should (roughly) match the number of trainable parameters for the baselines, are difficult to answer and depend on the context and the application. Similarly, as we elaborate in Sec. 9.3, assuming the right threat model is critical for evaluating adversarial robustness.

Evaluation is important not only for the final model but during development as well. For example, since our NetGAN model [7] was the first implicit graph generative model there were no established techniques for assessing the quality of the generated graphs making it difficult to evaluate different architectures. In general, the lack of (large-scale) graphs with ground-truth clusters (or anomalies, or labels) can impede progress.

In (writing about) research there is a challenge in balancing generality vs. having a single focused message, and balancing simplicity vs. including additional modifications that yield small improvements. For example, we discovered several small improvements (consistency regularization, adding additional f-divergence constraints, etc.) for our sparse smoothing certificate (Chapter 8). Ultimately however, we decided not to include them in the publication (and the thesis) since they would have diluted the main message. Similarly, generalizing the certificate to discrete data increases the impact, but makes is more likely for the method to be overlooked in the graph mining community. Overall, we strived to keep the methods as simple as possible, and I believe we struck a good balance.

[1] Ideally, we should train on graphs with edges observed up to a given point in time, and evaluate on future edges. However, for the standard benchmarks the temporal information is absent so we have to randomly split the data. This requires some care since some baselines cannot deal with singleton nodes and disconnected components, others only work well for undirected graphs, etc.

[2] In Bojchevski et al. [8] we argue that the sparsely-labeled scenario is most relevant in practice.

9.3 Open Questions

A central open question, concerning almost all of the work in this thesis, is how to design realistic yet tractable threat models. Balancing the practical relevance of the assumptions about the goals, knowledge, and capabilities of the adversary vs. the implied computational feasibility is challenging. For example, so far we have not accounted for the unequal cost of different perturbations[3] – perturbing a node corresponding to a high-profile or security-aware user is likely more difficult, and thus more expensive, compared to a regular user. Similarly, not every perturbation is deterministically executable but rather there is a certain probability that the attacker will be able to e.g. successfully flip a certain edge. The attacker is also not likely to have access to all nodes in the network but rather a certain subset, adding another optimization constraint. Moreover, the defender and the attacker dynamically adapt to each other's actions, e.g. the defender can immunize the most vulnerable or influential nodes, forcing the attacker to change their strategy, which in turn changes the defender's optimal strategy. The list of relevant aspects goes on, but the work in this thesis provides a solid foundation to address them.

So far our certifiable robustness estimates are too pessimistic since we allow the attacker to find a potentially different perturbed graph for different target nodes. However, in reality the attacker can only realize a single perturbation, i.e. a given edge can either be present or not, which limits the amount of damage they can inflict. Deriving certificates which account for this aspect of the threat model is an open problem.

From a theoretical perspective, another open question is to study adversarially robust learning from the viewpoint of generalization. Can the inclusion of additional relational information help close the generalization gap between standard and robust accuracy which has been (empirically and theoretically) demonstrated for i.i.d. data? There is evidence for example, that unlabeled data, which is implicitly considered in semi-supervised node classification, improves adversarial robustness [232].

Typically, we assume that the graph is explicitly given (e.g. a social network), with the robust spectral embedding from Chapter 3 as a notable exception. However, in many of the interesting practical applications of e.g. GNNs the graph is inferred or jointly learned with the rest of the model parameters during training. Developing robust graph-based models where the graph is implicit or learned from data is an interesting future direction.

Beyond robustness, there are many challenging and impactful open problems in the broader area of trustworthy machine learning for graph data. From criminal justice and employment to healthcare and autonomous driving, in high-stakes applications we have to consider questions about privacy, fairness, interpretability, and bias to ensure model reliability in practice. Given the close connection between adversarial robustness and e.g. differential privacy the tools we developed in this thesis can be useful for tackling these questions. Similarly, we can exploit recently established connections between randomized smoothing and fairness [245]. Specifically, an interesting line of research is to build upon our approach from Chapter 8 to provably ensure individual fairness, e.g. ensure that similar nodes are treated similarly.

[3]Although for our PageRank-based certificate described in Chapter 7 this is a trivial extension.

9.4 Broader Impact

In this section we discuss potential societal impacts of our research and related ethical considerations. Inspired by NeurIPS which recently introduced[4] a requirement that submissions must include a similar discussion of broader impact, this exercise can help inform policymakers to build policies that amplify the benefits and mitigate the risks.

At a first glance improving robustness seems like it should mostly have a positive impact. The list of disadvantages for ensuring that our models are reliable and behave as expected seems like it must be short, barring any potentially inherent clean accuracy trade-offs. Upon reflection we find that depending on the setting and the sensitivity of the data, improving (adversarial) robustness might be problematic. Since we highlight positive impacts throughout the thesis, here we discuss some potential negative impacts.

Take for example the robust clustering model PAICAN which we develop in Chapter 6, and assume we apply it to a social network. A person might be obfuscating some of their attributes (age, political affiliation) on purpose due to privacy concerns or to avoid being placed in a certain cluster. Such adversarial corruption might fool a classical model, but a robust model such as PAICAN is more likely to correctly infer the user's cluster and thereby also their sensitivity attributes. One could argue that in the wrong hands, e.g. a dictator persecuting their opposition, the robust model is more harmful. Similar arguments can be made for allowing attacks on face recognition or surveillance systems. A more direct negative impact of our work on adversarial attacks is the possibility that malicious actors may use the techniques we develop to damage real-world systems.

Overall, we believe that studying robustness has a net positive impact. More broadly however, to asses the true impact we have to engage and integrate the perspectives of the communities which are most affected by machine learning systems. These (often marginalized) communities are left out of the conversation when it comes to the design and deployment of these systems: starting from the dataset collection, to the trade-offs implied by different algorithmic choices, and ultimately the feedback loops which are created and reinforced. The general goal of designing systems which are trustworthy cannot be achieved without listening to their voices.

[4]See the call for papers at `https://neurips.cc/Conferences/2020/CallForPapers`

Bibliography

[1] Aleksandar Bojchevski, Yves Matkovic, and Stephan Günnemann. Robust spectral clustering for noisy data: Modeling sparse corruptions improves latent embeddings. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2017.

[2] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations, ICLR*, 2018.

[3] Aleksandar Bojchevski and Stephan Günnemann. Adversarial attacks on node embeddings via graph poisoning. In *International Conference on Machine Learning, ICML*, 2019.

[4] Aleksandar Bojchevski and Stephan Günnemann. Bayesian robust attributed graph clustering: Joint learning of partial anomalies and group structure. In *Conference on Artificial Intelligence, AAAI*, 2018.

[5] Aleksandar Bojchevski and Stephan Günnemann. Certifiable robustness to graph perturbations. In *Neural Information Processing Systems, NeurIPS*, 2019.

[6] Aleksandar Bojchevski, Johannes Klicpera, and Stephan Günnemann. Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more. In *International Conference on Machine Learning, ICML*, 2020.

[7] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. NetGAN: Generating graphs via random walks. In *International Conference on Machine Learning, ICML*, 2018.

[8] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate PageRank. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2020.

[9] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Martin Blais, Amol Kapoor, Michal Lukasik, and Stephan Günnemann. Is PageRank all you need for scalable graph neural networks? In *International Workshop on Mining and Learning with Graphs, MLG*, 2019.

[10] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized PageRank. In *International Conference on Learning Representations, ICLR*, 2019.

[11] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. In *Relational Representation Learning Workshop, R2L*, 2018.

[12] Oleksandr Shchur, Aleksandar Bojchevski, Mohamed Farghal, Stephan Günnemann, and Yusuf Saber. Anomaly detection in car-booking graphs. In *International Conference on Data Mining Workshops, ICDM*, 2018.

[13] Eugenio Angriman, Alexander van der Grinten, Aleksandar Bojchevski, Daniel Zügner, Stephan Günnemann, and Henning Meyerhenke. Group centrality maximization for large-scale graphs. In *Symposium on Algorithm Engineering and Experiments, ALENEX*, 2020.

[14] Federico Monti, Oleksandr Shchur, Aleksandar Bojchevski, Or Litany, Stephan Günnemann, and Michael M Bronstein. Dual-primal graph convolutional networks. In *Graph Embedding and Mining Workshop, GEM*, 2019.

[15] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *International Conference on Machine Learning, ICML*, 2012.

[16] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations, ICLR*, 2014.

[17] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *Neural Information Processing Systems, NIPS*, 2017.

[18] Jianyu Wang, Rui Wen, Chunming Wu, Yu Huang, and Jian Xion. Fdgars: Fraudster detection via graph convolutional networks in online app review system. In *International World Wide Web Conference, WWW*, 2019.

[19] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations, ICLR*, 2017.

[20] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Conference on Artificial Intelligence, AAAI*, 2019.

[21] SungMin Rhee, Seokjun Seo, and Sun Kim. Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification. In *International Joint Conference on Artificial Intelligence, IJCAI*, 2018.

[22] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *International Joint Conference on Neural Networks, IJCNN*, 2005.

[23] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *Transactions on Neural Networks*, 2008.

[24] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning, ICML*, Proceedings of Machine Learning Research, 2017.

[25] Thomas N. Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning, ICML*, 2018.

[26] Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. Efficient probabilistic logic reasoning with graph neural networks. In *International Conference on Learning Representations, ICLR*, 2020.

[27] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations, ICLR*, 2019.

[28] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *Transactions on Knowledge and Data Engineering, TKDE*, 2018.

[29] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 2008.

[30] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2018.

[31] Christian Von Mering, Lars J Jensen, Berend Snel, Sean D Hooper, Markus Krupp, Mathilde Foglierini, Nelly Jouffre, Martijn A Huynen, and Peer Bork. String: known and predicted protein–protein associations, integrated and transferred across organisms. *Nucleic Acids Research*, 2005.

[32] Michael W Mahoney. Lecture notes on spectral graph methods. *arXiv preprint arXiv:1608.04845*, 2016.

[33] Lodewijk Kallenberg. Markov decision processes. 2011.

[34] Taher H. Haveliwala. Topic-sensitive pagerank. In *International World Wide Web Conference, WWW*, 2002.

[35] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *International World Wide Web Conference, WWW*, 2003.

[36] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Neural Information Processing Systems, NIPS*, 2003.

[37] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *International Conference on Machine Learning, ICML*, 2005.

[38] Dengyong Zhou and Christopher J. C. Burges. Spectral clustering and transductive learning with multiple views. In *International Conference on Machine Learning, ICML*, 2007.

[39] Gilbert W Stewart. *Matrix perturbation theory.* 1990.

[40] Yuji Nakatsukasa. Absolute and relative weyl theorems for generalized eigenvalue problems. *Linear Algebra and its Applications*, 2010.

[41] Chi-Kwong Li and Roy Mathias. The lidskii-mirsky-wielandt theorem–additive and multiplicative versions. *Numerische Mathematik*, 1999.

[42] Zoltán Gyöngyi and Hector Garcia-Molina. Link spam alliances. In *International Conference on Very Large Data Bases, VLDB*, 2005.

[43] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations, ICLR*, 2015.

[44] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Conference on Computer Vision and Pattern Recognition, CVPR*, 2017.

[45] Charu C Aggarwal and Chandan K Reddy. *Data clustering: Algorithms and applications.* 2013.

[46] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 2007.

[47] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *Conference on Computer Vision and Pattern Recognition, CVPR*, 1997.

[48] Desmond J Higham, Gabriela Kalna, and Milla Kibble. Spectral clustering and its use in bioinformatics. *Journal of computational and applied mathematics*, 2007.

[49] Mingguang Shi and Guofu Xu. Spectral clustering using nyström approximation for the accurate identification of cancer molecular subtypes. *Scientific Reports*, 2017.

[50] Scott White and Padhraic Smyth. A spectral clustering approach to finding communities in graphs. In *International Conference on Data Mining, ICDM*, 2005.

[51] Yehuda Koren. Drawing graphs by eigenvectors: theory and practice. *Computers & Mathematics with Applications*, 2005.

[52] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Neural Information Processing Systems, NIPS*, 2002.

[53] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection.* 2005.

[54] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM, JACM*, 2011.

[55] Stephan Günnemann, Nikou Günnemann, and Christos Faloutsos. Detecting anomalies in dynamic rating data: A robust probabilistic model for rating evolution. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2014.

[56] Nikou Günnemann, Stephan Günnemann, and Christos Faloutsos. Robust multivariate autoregression for anomaly detection in dynamic product ratings. In *International World Wide Web Conference, WWW*, 2014.

[57] Zhenguo Li, Jianzhuang Liu, Shifeng Chen, and Xiaoou Tang. Noise robust spectral clustering. In *International Conference on Computer Vision, ICCV*, 2007.

[58] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Neural Information Processing Systems, NIPS*, 2004.

[59] Wanzeng Kong, Sanqing Hu, Jianhai Zhang, and Guojun Dai. Robust and smart spectral clustering from normalized cut. *Neural Computing and Applications*, 2013.

[60] Hao Huang, Shinjae Yoo, Hong Qin, and Dantong Yu. A robust clustering algorithm based on aggregated heat kernel mapping. In *International Conference on Data Mining, ICDM*, 2011.

[61] Francis R. Bach and Michael I. Jordan. Learning spectral clustering. *Neural Information Processing Systems, NIPS*, 2003.

[62] Xiatian Zhu, Chen Loy, and Shaogang Gong. Constructing robust affinity graphs for spectral clustering. In *Conference on Computer Vision and Pattern Recognition, CVPR*, 2014.

[63] Stephan Günnemann, Ines Färber, Sebastian Raubach, and Thomas Seidl. Spectral subspace clustering for graphs with feature vectors. In *International Conference on Data Mining, ICDM*, 2013.

[64] Xi Li, Weiming Hu, Chunhua Shen, Anthony Dick, and Zhongfei Zhang. Context-aware hypergraph construction for robust spectral clustering. *Transactions on Knowledge and Data Engineering, TKDE*, 2014.

[65] Hong Chang and Dit-Yan Yeung. Robust path-based spectral clustering. *Pattern Recognition*, 2008.

[66] Xiaoqian Wang, Feiping Nie, and Heng Huang. Structured doubly stochastic matrix for graph based clustering. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2016.

[67] Benjamin A Miller, Michelle S Beard, Patrick J Wolfe, and Nadya T Bliss. A spectral framework for anomalous subgraph detection. *Transactions on Signal Processing, TSP*, 2015.

[68] Chandler Davis and William Morton Kahan. The rotation of eigenvectors by a perturbation. iii. *Journal on Numerical Analysis, SINUM*, 1970.

[69] Leting Wu, Xiaowei Ying, Xintao Wu, and Zhi-Hua Zhou. Line orthogonality in adjacency eigenspace with application to community partition. In *International Joint Conference on Artificial Intelligence, IJCAI*, 2011.

[70] Jella Pfeiffer and Franz Rothlauf. Analysis of greedy heuristics and weight-coded eas for multidimensional knapsack problems and multi-unit combinatorial auctions. In *Genetic and Evolutionary Computation Conference, GECCO*, 2007.

[71] Daniel Lehmann, Liadan Ita Oćallaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM, JACM*, 2002.

[72] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Computational and applied mathematics*, 1987.

[73] Dheeru Dua and Casey Graff. "UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[74] Jonathan J. Hull. A database for handwritten text recognition research. *Transactions on Pattern Analysis and Machine Intelligence, TPAMI*, 1994.

[75] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2010.

[76] Anne Condon and Richard M Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 2001.

[77] Quanquan Gu, Charu Aggarwal, and Jiawei Han. Unsupervised link selection in networks. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, 2013.

[78] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Review*, 1996.

[79] Stephen Boyd. Convex optimization of graph laplacian eigenvalues. In *Proceedings of the International Congress of Mathematicians*, 2006.

[80] Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová, and Pan Zhang. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences*, 2013.

[81] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *International Joint Conference on Artificial Intelligence IJCAI*, 2015.

[82] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. Tri-party deep network representation. In *International Joint Conference on Artificial Intelligence, IJCAI*, 2016.

[83] Soumyajit Ganguly and Vikram Pudi. Paper2vec: Combining graph and text information for scientific paper representation. In *European Conference on Information Retrieval, ECIR*, 2017.

[84] W. L. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[85] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 2018.

[86] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *Conference on Knowledge Discovery and Data Mining, KDD*, 2014.

[87] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Conference on Knowledge Discovery and Data Mining, KDD*, 2016.

[88] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations, ICLR, Workshop Track*, 2013.

[89] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: Large-scale information network embedding. In *International World Wide Web Conference, WWW*, 2015.

[90] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2016.

[91] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *International on Conference on Information and Knowledge Management, CIKM*, 2015.

[92] Xiaofei Sun, Jiang Guo, Xiao Ding, and Ting Liu. A general framework for content-enhanced network representation learning. *arXiv preprint arXiv:1610.02906*, 2016.

[93] Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2015.

[94] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. Heterogeneous network embedding via deep architectures. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2015.

[95] Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In *International Conference on Web Search and Data Mining, WSDM*, 2017.

[96] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Neural Information Processing Systems, NIPS*, 2017.

[97] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Neural Information Processing Systems, NIPS*, 2016.

[98] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Conference on Computer Vision and Pattern Recognition, CVPR*, 2018.

[99] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning, ICML*, 2016.

[100] Trang Pham, Truyen Tran, Dinh Q Phung, and Svetha Venkatesh. Column networks for collective classification. In *Conference on Artificial Intelligence, AAAI*, 2017.

[101] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[102] Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. In *International Conference On Learning Representations, ICLR*, 2014.

[103] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. Learning to represent knowledge graphs with gaussian embedding. In *International Conference on Information and Knowledge Management, CIKM*, 2015.

[104] Ludovic Dos Santos, Benjamin Piwowarski, and Patrick Gallinari. Multilabel classification on heterogeneous graphs with gaussian embeddings. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases, ECML-PKDD*, 2016.

[105] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 2006.

[106] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, ICLR*, 2015.

[107] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 2000.

[108] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Conference on Digital Libraries, DL*, 1998.

[109] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research, JMLR*, 2008.

[110] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. Verse: Versatile graph embeddings from similarity measures. In *International World Wide Web Conference, WWW*, 2018.

[111] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2019.

[112] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Neural Information Processing Systems, NIPS*, 2017.

[113] Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. Poincaré glove: Hyperbolic word embeddings. In *International Conference on Learning Representations, ICLR*, 2019.

[114] Marten Lienen. Uniform edge sampling from complete k-partite graphs, 2019. URL `https://martenlienen.com/blog/sampling-k-partite-graph-edges/`.

[115] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Conference on Artificial Intelligence, AAAI*, 2015.

[116] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *Symposium on Security and Privacy*, 2017.

[117] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. In *International Joint Conference on Artificial Intelligence IJCAI*, 2018.

[118] Moustapha Cissé, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured prediction models. *arXiv preprint arXiv:1707.05373*, 2017.

[119] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. In *International Joint Conference on Artificial Intelligence IJCAI*, 2017.

[120] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[121] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2018.

[122] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International Conference on Machine Learning, ICML*, 2018.

[123] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations, ICLR*, 2019.

[124] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. Sybilguard: defending against sybil attacks via social networks. In *Special Interest Group on Data Communications, SIGCOMM*, 2006.

[125] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *Neural Information Processing Systems, NIPS*, 2016.

[126] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Neural Information Processing Systems, NIPS*, 2016.

[127] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations, ICLR*, 2014.

[128] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial examples for malware detection. In *European Symposium on Research in Computer Security, ESORICS*, 2017.

[129] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *European Symposium on Security and Privacy, EuroS&P*, 2016.

[130] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *Transactions on Knowledge and Data Engineering, TKDE*, 2013.

[131] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C. Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Workshop on Artificial Intelligence and Security, AISec*, 2017.

[132] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning, ICML*, 2017.

[133] Elias Boutros Khalil, Bistra N. Dilkina, and Le Song. Scalable diffusion-aware optimization of network topology. In *Conference on Knowledge Discovery and Data Mining, KDD*, 2014.

[134] Chen Chen, Hanghang Tong, B. Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. Eigen-optimization on large graphs by edge manipulation. *Transactions on Knowledge Discovery from Data, TKDD*, 2016.

[135] Vineet Chaoji, Sayan Ranu, Rajeev Rastogi, and Rushi Bhatt. Recommendations to boost content spread in social networks. In *International World Wide Web Conference, WWW*, 2012.

[136] Victor Amelkin and Ambuj K. Singh. Fighting opinion control in social networks via link recommendation. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2019.

[137] Cynthia A. Phillips. The network inhibition problem. In *Symposium on Theory of Computing, STOC*, 1993.

[138] Eitan Israeli and R. Kevin Wood. Shortest-path network interdiction. *Networks*, 2002.

[139] Balázs Csanád Csáji, Raphaël M. Jungers, and Vincent D. Blondel. Pagerank optimization by edge selection. *Discrete Applied Mathematics*, 2014.

[140] Hau Chan, Leman Akoglu, and Hanghang Tong. Make it or break it: Manipulating robustness in large networks. In *International Conference on Data Mining, ICDM*, 2014.

[141] Yizheng Chen, Yacin Nadji, Athanasios Kountouras, Fabian Monrose, Roberto Perdisci, Manos Antonakakis, and Nikolaos Vasiloglou. Practical attacks against graph-based clustering. In *Special Interest Group on Security, Audit and Control SIGSAC*, 2017.

[142] Mengchen Zhao, Bo An, Yaodong Yu, Sulin Liu, and Sinno Jialin Pan. Data poisoning attacks on multi-task relationship learning. In *Conference on Artificial Intelligence, AAAI*, 2018.

[143] Daniel Zügner and Stephan Günnemann. Certifiable robustness and robust training for graph convolutional networks. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2019.

[144] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. Adversarial network embedding. In *Conference on Artificial Intelligence, AAAI*, 2018.

[145] Cheng Yang and Zhiyuan Liu. Comprehend deepwalk as matrix factorization. *arXiv preprint arXiv:1501.00358*, 2015.

[146] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *International Conference on Web Search and Data Mining WSDM*, 2018.

[147] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 2007.

[148] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations, ICLR*, 2018.

[149] Lada A. Adamic and Natalie S. Glance. The political blogosphere and the 2004 U.S. election: divided they blog. In *International Workshop on Link Discovery, LinkKDD*, 2005.

[150] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, 2002.

[151] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning, ICML*, 2019.

[152] Heng Chang, Yu Rong, Tingyang Xu, Wenbing Huang, Honglei Zhang, Peng Cui, Wenwu Zhu, and Junzhou Huang. A restricted black-box adversarial framework towards attacking graph embedding models. In *Conference on Artificial Intelligence, AAAI*, 2020.

[153] Cécile Bothorel, Juan David Cruz Gomez, Magnani Matteo, and Barbora Micenkova. Clustering attributed graphs: models, measures and methods. *Network Science*, 2015.

[154] Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. Focused clustering and outlier detection in large attributed graphs. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2014.

[155] Jing Gao, Feng Liang, Wei Fan, Chi Wang, Yizhou Sun, and Jiawei Han. On community outliers and their efficient detection in information networks. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2010.

[156] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery, DMKD*, 2015.

[157] John Wright, Arvind Ganesh, Shankar R. Rao, YiGang Peng, and Yi Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Neural Information Processing Systems, NIPS*, 2009.

[158] Liang Xiong, Xi Chen, and Jeff Schneider. Direct robust matrix factorizatoin for anomaly detection. In *International Conference on Data Mining, ICDM*, 2011.

[159] Leman Akoglu, Hanghang Tong, Brendan Meeder, and Christos Faloutsos. PICS: parameter-free identification of cohesive subgroups in large attributed graphs. In *International Conference on Data Mining, ICDM*, 2012.

[160] Jaewon Yang, Julian J. McAuley, and Jure Leskovec. Community detection in networks with node attributes. In *International Conference on Data Mining, ICDM*, 2013.

[161] Mark EJ Newman and Aaron Clauset. Structure and inference in annotated networks. *Nature Communications*, 2016.

[162] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. GBAGC: A general bayesian framework for attributed graph clustering. *Transactions on Knowledge Discovery from Data, TKDD*, 2014.

[163] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. A model-based approach to attributed graph clustering. In *Special Interest Group on Management of Data, SIGMOD*, 2012.

[164] Darko Hric, Tiago P Peixoto, and Santo Fortunato. Network structure, metadata, and the prediction of missing nodes and annotations. *Physical Review X*, 2016.

[165] Tomoharu Iwata and Makoto Yamada. Multi-view anomaly detection via robust probabilistic latent variable models. In *Neural Information Processing Systems, NIPS*, 2016.

[166] Stephan Günnemann, Brigitte Boden, and Thomas Seidl. Finding density-based subspace clusters in graphs with feature vectors. *Data Mining and Knowledge Discovery, DMKD*, 2012.

[167] Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 2011.

[168] Xiaoran Yan, Cosma Shalizi, Jacob E Jensen, Florent Krzakala, Cristopher Moore, Lenka Zdeborová, Pan Zhang, and Yaojia Zhu. Model selection for degree-corrected block models. *Journal of Statistical Mechanics*, 2014.

[169] Christopher M Bishop. Pattern recognition. *Machine Learning*, 2006.

[170] Béla Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *Electronic Journal of Combinatorics, EJC*, 1980.

[171] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys (CSUR)*, 2006.

[172] Emmanuel Lazega et al. *The collegial phenomenon: The social mechanisms of cooperation among peers in a corporate law partnership*. 2001.

[173] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[174] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, 2014.

[175] Dustin Tran, Matthew D. Hoffman, Rif A. Saurous, Eugene Brevdo, Kevin Murphy, and David M. Blei. Deep probabilistic programming. In *International Conference on Learning Representations, ICLR*, 2017.

[176] Oleksandr Shchur and Stephan Günnemann. Overlapping community detection with graph neural networks. In *International Workshop on Deep Learning on Graphs, DLG*, 2019.

[177] Carlos Castillo and Brian D. Davison. Adversarial web search. *Foundations and Trends in Information Retrieval*, 2010.

[178] Bryan Hooi, Neil Shah, Alex Beutel, Stephan Günnemann, Leman Akoglu, Mohit Kumar, Disha Makhija, and Christos Faloutsos. BIRDNEST: bayesian inference for ratings-fraud detection. In *International Conference on Data Mining, ICDM*, 2016.

[179] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2018.

[180] Eliav Buchnik and Edith Cohen. Bootstrapped graph diffusions: Exposing the power of nonlinearity. In *International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS*, 2018.

[181] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Neural Information Processing Systems, NIPS*, 2017.

[182] Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning, ICML*, 2018.

[183] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations, ICLR*, 2014.

[184] Aditi Raghunathan, Jacob Steinhardt, and Percy S. Liang. Semidefinite relaxations for certifying robustness adversarial examples. In *Neural Information Processing Systems, NeurIPS*, 2018.

[185] Jinyin Chen, Yangyang Wu, Xiang Lin, and Qi Xuan. Can adversarial network attack be defended? *arXiv preprint arXiv:1903.05994*, 2019.

[186] Fuli Feng, Xiangnan He, Jie Tang, and Tat-Seng Chua. Graph adversarial training: Dynamically regularizing based on graph structure. *arXiv preprint arXiv:1902.08226*, 2019.

[187] Ke Sun, Hantao Guo, Zhanxing Zhu, and Zhouchen Lin. Virtual adversarial training on graph convolutional networks in node classification. *Pattern Recognition and Computer Vision, PRCV*, 2019.

[188] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. In *International Joint Conference on Artificial Intelligence, IJCAI*, 2019.

[189] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: Deep insights into attack and defense. In *International Joint Conference on Artificial Intelligence, IJCAI*, 2019.

[190] Shen Wang, Zhengzhang Chen, Jingchao Ni, Xiao Yu, Zhichun Li, Haifeng Chen, and Philip S Yu. Adversarial defense framework for graph neural network. *arXiv preprint arXiv:1905.03679*, 2019.

[191] Xianfeng Tang, Yandong Li, Yiwei Sun, Huaxiu Yao, Prasenjit Mitra, and Suhang Wang. Robust graph neural network against poisoning attacks via transfer learning. *arXiv preprint arXiv:1908.07558*, 2019.

[192] Benjamin A Miller, Mustafa Çamurcu, Alexander J Gomez, Kevin Chan, and Tina Eliassi-Rad. Improving robustness to attacks against vertex classification. In *International Workshop on Mining and Learning with Graphs, MLG*, 2018.

[193] Yingxue Zhang, S Khan, and Mark Coates. Comparing and detecting adversarial attacks for graph deep learning. In *Representation Learning on Graphs and Manifolds, RLGM*, 2019.

[194] Kai Zhou, Tomasz P Michalak, and Yevgeniy Vorobeychik. Adversarial robustness of similarity-based link prediction. In *International Conference on Data Mining, ICDM*, 2019.

[195] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Üstebay. Bayesian graph convolutional neural networks for semi-supervised classification. In *Conference on Artificial Intelligence, AAAI*, 2019.

[196] NT Hoang, Jun Jin Choong, and Tsuyoshi Murata. Learning graph neural networks with noisy labels. *arXiv preprint arXiv:1905.01591*, 2019.

[197] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. In *Representation Learning on Graphs and Manifolds, RLGM*, 2019.

[198] Konstantin Avrachenkov and Nelly Litvak. The effect of new links on google pagerank. *Stochastic Models*, 2006.

[199] Balázs Csanád Csáji, Raphaël M. Jungers, and Vincent D. Blondel. Pagerank optimization in polynomial time by stochastic shortest path reformulation. In *International Conference on Algorithmic Learning Theory, ALT*, 2010.

[200] Cristobald de Kerchove, Laure Ninove, and Paul Van Dooren. Maximizing pagerank via outlinks. *Linear Algebra and its Applications*, 2008.

[201] Olivier Fercoq, Marianne Akian, Mustapha Bouhtou, and Stephane Gaubert. Ergodic control and polyhedral approaches to pagerank optimization. *Transactions on Automatic Control, TAC*, 2013.

[202] Romain Hollanders, Jean-Charles Delvenne, and Raphaël Jungers. Policy iteration is well suited to optimize pagerank. *arXiv preprint arXiv:1108.3779*, 2011.

[203] Martin Olsen. Maximizing pagerank with new backlinks. In *International Conference on Algorithms and Complexity, CIAC*, 2010.

[204] Olivier Fercoq. *Optimization of Perron eigenvectors and applications: from web ranking to chronotherapeutics.* PhD thesis, Ecole Polytechnique X, 2012.

[205] Martin L Puterman. *Markov decision processes: Discrete stochastic dynamic programming.* 1994.

[206] Hanif D Sherali and Cihan H Tuncbilek. A reformulation-convexification approach for solving nonconvex quadratic programming problems. *Journal of Global Optimization*, 1995.

[207] Eitan Altman. *Constrained Markov decision processes.* 1999.

[208] John M Danskin. *The theory of max-min and its application to weapons allocation problems.* 1967.

[209] Shuchang Tao, Huawei Shen, Qi Cao, Liang Hou, and Xueqi Cheng. Adversarial immunization for improving certifiable robustness on graphs. *arXiv preprint arXiv:2007.09647*, 2020.

[210] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International Conference on Machine Learning, ICML*, Proceedings of Machine Learning Research, 2018.

[211] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations, ICLR*, 2019.

[212] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. In *International Joint Conference on Artificial Intelligence, IJCAI*, 2019.

[213] Negin Entezari, Saba A. Al-Sayouri, Amirali Darvishzadeh, and Evangelos E. Papalexakis. All you need is low (rank): Defending against adversarial attacks on graphs. In *International Conference on Web Search and Data Mining WSDM*, 2020.

[214] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Workshop on Artificial Intelligence and Security, AISec*, 2017.

[215] Daniel Zügner and Stephan Günnemann. Certifiable robustness and robust training for graph convolutional networks. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2019.

[216] Daniel Zügner and Stephan Günnemann. Certifiable robustness of graph convolutional networks under structure perturbations. In *International Conference on Knowledge Discovery and Data Mining, KDD*, New York, NY, USA, 2020.

[217] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning, ICML*, Proceedings of Machine Learning Research, 2019.

[218] Mathias Lécuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *Symposium on Security and Privacy*, 2019.

[219] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Second-order adversarial attack and certifiable robustness. *CoRR*, 2018.

[220] Guang-He Lee, Yang Yuan, Shiyu Chang, and Tommi S. Jaakkola. Tight certificates of adversarial robustness for randomly smoothed classifiers. In *Neural Information Processing Systems, NeurIPS*, 2019.

[221] Alexander Levine and Soheil Feizi. Robustness certificates for sparse adversarial attacks by randomized ablation. In *Conference on Artificial Intelligence, AAAI*, 2020.

[222] Krishnamurthy (Dj) Dvijotham, Jamie Hayes, Borja Balle, Zico Kolter, Chongli Qin, Andras Gyorgy, Kai Xiao, Sven Gowal, and Pushmeet Kohli. A framework for robustness certification of smoothed classifiers using f-divergences. In *International Conference on Learning Representations, ICLR*, 2020.

[223] Jinyuan Jia, Binghui Wang, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified robustness of community detection against adversarial structural perturbation via randomized smoothing. In *International World Wide Web Conference, WWW*, 2020.

[224] Charles J Clopper and Egon S Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 1934.

[225] Keith D Tocher. Extension of the neyman-pearson theory of tests to discontinuous variates. *Biometrika*, 1950.

[226] Sean X Chen and Jun S Liu. Statistical applications of the poisson-binomial and conditional bernoulli distributions. *Statistica sinica*, 1997.

[227] Manuel Fernández and Stuart Williams. Closed-form expression for the poisson-binomial probability density function. *Transactions on Aerospace and Electronic Systems, TAES*, 2010.

[228] Hadi Salman, Jerry Li, Ilya P. Razenshteyn, Pengchuan Zhang, Huan Zhang, Sébastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In *Neural Information Processing Systems, NeurIPS*, 2019.

[229] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations, ICLR*, 2017.

[230] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations, ICLR*, 2018.

[231] Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning, ICML*, Proceedings of Machine Learning Research, 2018.

[232] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C. Duchi, and Percy Liang. Unlabeled data improves adversarial robustness. In *Neural Information Processing Systems, NeurIPS*, 2019.

[233] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations, ICLR*, 2018.

[234] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 2008.

[235] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Neural Information Processing Systems, NIPS*, 2016.

[236] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations, ICLR*, 2020.

[237] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning, ICML*, Proceedings of Machine Learning Research, 2018.

[238] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Neural Information Processing Systems, NIPS*, 2017.

[239] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Neural Information Processing Systems, NeurIPS*, 2018.

[240] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations, ICLR*, 2019.

[241] Jongheon Jeong and Jinwoo Shin. Consistency regularization for certified robustness of smoothed classifiers. *arXiv preprint arXiv:2006.04062*, 2020.

[242] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. *arXiv preprint arXiv:2003.02460*, 2020.

[243] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, and Bryan Hooi. Nodeaug: Semi-supervised node classification with data augmentation. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2020.

[244] Yang Yang, Ryan N Lichtenwalter, and Nitesh V Chawla. Evaluating link prediction methods. *Knowledge and Information Systems*, 2015.

[245] Samuel Yeom and Matt Fredrikson. Individual fairness revisited: Transferring techniques from adversarial robustness. In Christian Bessiere, editor, *International Joint Conference on Artificial Intelligence, IJCAI*, 2020.

[246] Euan Adie and William Roe. Altmetric: enriching scholarly content with article-level discussion and metrics. *Learned Publishing*, 2013.

[247] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition, CVPR*, 2009.

[248] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan

Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 2020.

[249] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[250] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems, NIPS*, 2019.

[251] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research, JMLR*, pages 1–5, 2016.

[252] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

[253] Marina Sokol, Konstantin Avrachenkov, Paulo Gonçalves, and Alexey Mishenin. Generalized optimization framework for graph-based semi-supervised learning. In *International Conference on Data Mining, ICDM*, 2012.

[254] Martin Olsen, Anastasios Viglas, and Ilia Zvedeniouk. A constant-factor approximation algorithm for the link building problem. In *International Conference on Combinatorial Optimization and Applications, COCOA*, 2010.

[255] Leizhen Cai. Parameterized complexity of cardinality constrained optimization problems. *The Computer Journal*, 2008.

[256] Jinyuan Jia, Xiaoyu Cao, Binghui Wang, and Neil Zhenqiang Gong. Certified robustness for top-k predictions against adversarial perturbations via randomized smoothing. In *International Conference on Learning Representations, ICLR*, 2020.

Appendices

A Data and Code

A.1 Datasets

A.1.1 Graph Datasets

Here we provide a description of the real-world datasets which we frequently use for experimental evaluation in this thesis. Cora-ML [107], Cora (Cora-Full) [107], Citeseer [108], DBLP [82], and PubMed [234] are citation networks. Here, the nodes correspond to papers, the edges correspond to citations (references) between the papers, and the node features correspond to bag-of-words representation of the paper's abstract. The node labels correspond to the paper's field of study (e.g. formal languages or probabilistic methods). Since citation graphs are commonly used as benchmark datasets in the literature you will encounter them throughout this thesis.

Several slightly different versions of the Cora dataset are publicly available. Since the metadata (e.g. which feature dimensions corresponds to which word) was not available we manually processed the original raw data from McCallum et al. [107] to obtain the two versions we use in this thesis. The larger graph contains papers from many different computer science fields, while Cora-ML contains only a subset of machine learning papers.

In the PolBlogs [149] graph each node is a political blog, the edges correspond to links between the blogs, and there are no node features available. Each blog is categorized based on its political leaning as either blue or red which serves as the node label.

We created the SocialPapers dataset to evaluate the approach described in Chapter 6. Here nodes represent biomedical papers forming edges if they are frequently mentioned by the same users on social media. The attributes designate the paper's subjects (e.g. psychology, neurology), and the in which the papers are published are considered as ground-truth labels. The data was collected using the Altmetric API [246]. Based on product review data [173] we also created an Amazon co-purchase attributed graph. The edges indicate that two products are often purchased together, and the attributes are binary product category indicators (e.g. books, jewelry). Here labels are not available.

The graphs above all have medium or large number of nodes. Since the baselines discussed in Chapter 6 lack scalability we also use several small graphs. In the Parliament[1] dataset the nodes are French parliament members having an edge if they cosigned a bill together, while their attributes indicate their constituency. Here we consider political parties as ground-truth clusters. The Lazega Lawyers [172] dataset contains different networks, where edges indicate e.g. a strong coworker or friendship relation among attorneys. We also have categorical attributes (e.g. seniority, formal status) for each node. We use the friendship network and binarize the attributes. HVR [161] is a dataset

[1]Raw data obtained from `https://github.com/briatte/parlement`

Table A.1: Statistics about the graph datasets used in this thesis. Each column shows a different number associated with the graph: n nodes, e edges, d features, e_d non-zero elements in the sparse feature matrix, k classes, c connected components. n_s and e_s show the number of nodes and edges after standardization (selecting the largest connected component and making the graph undirected) and "-" indicates no changed compared to n and e.

Dataset	n	e	d	e_d	k	c	n_s	e_s	type
Cora-ML	2995	8416	2879	151 171	7	61	2810	15 962	citation
Cora	19 793	65 311	8710	1 125 610	70	364	18 800	18 800	citation
Citeseer	3312	4715	3703	105 165	6	438	2110	7336	citation
DBLP	17 716	105 734	1639	92 192	4	589	16 191	103 826	citation
PubMed	19 717	88 648	500	988 031	3	1	-	-	citation
PolBlogs	1490	19 025	n/a	n/a	2	268	1222	33 428	blog links
SocialPapers	20 007	2 088 048	96	24 615	836	1	-	-	co-mention
Amazon	29 618	850 348	4643	96 709	-	1	-	-	co-purchase
Parliament	451	11 646	108	451	7	1	-	-	co-sign
Lazega Lawyers	71	575	70	426	2	1	-	798	friendship
HVR	307	3263	6	307	2	1	-	6502	genes

consisting of several networks of highly recombinant malaria parasite genes. The edges are based on the length of the longest substring shared between the genes sequences.

In Table A.1 we report detailed statistics about each graph. We see that most of them are sparse. Since we often standardize the graphs as a preprocessing steps, i.e. we select only the nodes that belong to the largest connected component and we make the graphs undirected and unweighted, we also show the statistics after standardization. We note that most graphs exhibit strong homophily, that is the edge density between nodes from the same class is significantly larger compared to the edge density between nodes from different classes (not shown).

A.1.2 Other Datasets

MNIST [75] ($n = 70\,000, d = 784, k = 10$), USPS [74] ($n = 9298, d = 256, k = 10$), and Pendigits [73] ($n = 7494, d = 16, k = 10$) are dataset of handwritten digits. Specifically, for MNIST and USPS we have 8-bit grayscale images of size 28×28 and 16×16 respectively of the digits 0 through 9.[2] For Pendigits the features correspond to the (x, y) coordinate information from a pressure sensitive tablet.

Iris [73] ($n = 150, d = 4, k = 3$) is a dataset about the morphologic variation of Iris flowers. The Banknote authentication data [73] ($n = 1372, d = 4, k = 3$) is extracted by applying the wavelet transform to images that were taken from genuine and forged banknotes. The ImageNet [247] datasets ($n = 14\,197\,122, d = 65\,536, k = 21\,841$) has over 14 million color images scaled to 256×256 pixels.

[2]Obtained from `https://cs.nyu.edu/~roweis/data.html`

A.2 Code

We provide an open-source implementation of all the methods discussed in this thesis. All implementations are in Python. We often rely on the following packages: Numpy/Scipy [248], Tensorflow [249], PyTorch [250] and CVXPY [251].

The code for each method can be found at https://github.com/abojchevski/[method]/ where '[method]' is specified below:

- /rsc/ implements the RSC method discussed in Chapter 3
- /graph2gauss/ implements the Graph2Gauss method discussed in Chapter 4
- /node_embedding_attack/ implements the attack discussed in Chapter 5
- /paican/ implements the PAICAN method discussed in Chapter 6
- /graph_cert/ implements the certificate discussed in Chapter 7
- /sparse_smoothing/ implements the certificate discussed in Chapter 8

B Robust Spectral Clustering

B.1 Proofs

Proof of Lemma 3.1. Note that

$$\boldsymbol{L}(\boldsymbol{A}^g) = \boldsymbol{D}(\boldsymbol{A}^g) - \boldsymbol{A}^g = (\boldsymbol{D}(\boldsymbol{A}) - \boldsymbol{D}(\boldsymbol{A}^c)) - (\boldsymbol{A} - \boldsymbol{A}^c) = \boldsymbol{L}(\boldsymbol{A}) - \boldsymbol{L}(\boldsymbol{A}^c) \tag{B.1}$$

Thus, $\mathrm{Tr}(\boldsymbol{H}^T\boldsymbol{L}(\boldsymbol{A}^g)\boldsymbol{H})$ can equivalently be written as

$$\mathrm{Tr}(\boldsymbol{H}^T \cdot (\boldsymbol{L}(\boldsymbol{A}) - \boldsymbol{L}(\boldsymbol{A}^c)) \cdot \boldsymbol{H}) = \mathrm{Tr}(\boldsymbol{H}^T \cdot \boldsymbol{L}(\boldsymbol{A}) \cdot \boldsymbol{H}) - \mathrm{Tr}(\boldsymbol{H}^T \cdot \boldsymbol{L}(\boldsymbol{A}^c) \cdot \boldsymbol{H}) \tag{B.2}$$

Given $\boldsymbol{H}$, the term $\mathrm{Tr}(\boldsymbol{H}^T \cdot \boldsymbol{L}(\boldsymbol{A}) \cdot \boldsymbol{H})$ is constant. Thus, minimizing the previous term is equivalent to maximizing $\mathrm{Tr}(\boldsymbol{H}^T \cdot \boldsymbol{L}(\boldsymbol{A}^c) \cdot \boldsymbol{H})$.

Let $\boldsymbol{y}_k$ be a column vector of $\boldsymbol{H}$, noticing that (see [46])

$$\boldsymbol{y}_k^T\boldsymbol{L}(\boldsymbol{A}^c)\boldsymbol{y}_k = \sum_{ij} \frac{1}{2} \cdot \boldsymbol{A}^c_{ij} \cdot (\boldsymbol{y}_{ki} - \boldsymbol{y}_{kj})^2 \tag{B.3}$$

and exploiting the orthogonality of $\boldsymbol{H}$ it follows:

$$\mathrm{Tr}(\boldsymbol{H}^T \cdot \boldsymbol{L}(\boldsymbol{A}^c) \cdot \boldsymbol{H}) = \sum_k \sum_{ij} \frac{1}{2} \cdot \boldsymbol{A}^c_{ij} \cdot (\boldsymbol{y}_{ki} - \boldsymbol{y}_{kj})^2 = \sum_{ij} \frac{1}{2} \cdot \boldsymbol{A}^c_{ij} \cdot \|\boldsymbol{h}_i - \boldsymbol{h}_j\|_2^2 \tag{B.4}$$

where the last step used $\boldsymbol{y}_{ki} = \boldsymbol{h}_{ik}$. To ensure that $\boldsymbol{A}^c$ as well as $\boldsymbol{A}^g$ are non-negative, it holds $0 \leq \boldsymbol{A}^c_{ij} \leq \boldsymbol{A}_{ij}$. Thus, if $\boldsymbol{A}_{ij} = 0$ then $\boldsymbol{A}^c_{ij} = 0$. Exploiting this fact and the symmetry of the graph leads to $\sum_{ij} \frac{1}{2} \cdot \boldsymbol{A}^c_{ij} \cdot \|\boldsymbol{h}_i - \boldsymbol{h}_j\|_2^2 = \sum_{(i,j)\in\mathcal{E}} \boldsymbol{A}^c_{ij} \cdot \|\boldsymbol{h}_i - \boldsymbol{h}_j\|_2^2$.

Next, we show that there exists a solution where each $\boldsymbol{A}^c_{ij} \in \{0, \boldsymbol{A}_{ij}\}$. As known, $0 \leq \boldsymbol{A}^c_{ij} \leq \boldsymbol{A}_{ij}$. Let $M = [\boldsymbol{A}^c_e]_{e\in\mathcal{E}}$ be a maximum of Eq. 3.7 where some $\boldsymbol{A}^c_{ij} > 0$ but $\boldsymbol{A}^c_{ij} < \boldsymbol{A}_{ij}$. Let M' be the solution where this entry is replaced by $\boldsymbol{A}^c_{ij} = \boldsymbol{A}_{ij}$. Since only $\|.\|_0$ constraints are used, M and M' fulfill the same constraints. Since $\|\boldsymbol{h}_i - \boldsymbol{h}_j\|_2^2$ is non-negative, $f_1(M') \geq f_1(M)$. It follows, that a solution minimizing Eq. 3.6 can be found by investigating $\boldsymbol{A}^c_{ij} = 0$ or $\boldsymbol{A}^c_{ij} = \boldsymbol{A}_{ij}$ only. □

Proof of Lemma 3.2. The goal is to find a matrix $\boldsymbol{A}^g$ whose sum of the first k eigenvalues is minimal (and fulfills the given constraints). Since, however, $\boldsymbol{A}^g$ is not known, we refer to the principle of eigenvalue perturbation. Let $\boldsymbol{A}^t$ be the matrix obtained in the previous iteration of the alternating optimization and let $\boldsymbol{y}_i$ be the i-th generalized eigenvector of $\boldsymbol{L}(\boldsymbol{A}^t)$ (these are the *columns* of the matrix $\boldsymbol{H}$ from above, i.e. $\boldsymbol{y}_{ij} = \boldsymbol{h}_{ji}$). Furthermore, denote the corresponding eigenvalues with λ_i. We define $\boldsymbol{L}(\boldsymbol{A}^g) - \boldsymbol{L}(\boldsymbol{A}^t) =: \Delta\boldsymbol{L}$ and

$\boldsymbol{D}(\boldsymbol{A}^g)-\boldsymbol{D}(\boldsymbol{A}^t) = \Delta\boldsymbol{D}$. Based on the theory of eigenvalue perturbation [39], the eigenvalue λ_i^g of $\boldsymbol{L}(\boldsymbol{A}^g)$ can be approximated by

$$\begin{aligned}\lambda_i^g &\approx \lambda_i + \boldsymbol{y}_i^T \cdot (\Delta\boldsymbol{L} - \lambda_i \cdot \Delta\boldsymbol{D}) \cdot \boldsymbol{y}_i \\ &= \lambda_i + \boldsymbol{y}_i^T \cdot ((\boldsymbol{L}(\boldsymbol{A}^g) - \boldsymbol{L}(\boldsymbol{A}^t)) - \lambda_i \cdot (\boldsymbol{D}(\boldsymbol{A}^g) - \boldsymbol{D}(\boldsymbol{A}^t))) \cdot \boldsymbol{y}_i\end{aligned} \tag{B.5}$$

Using the fact that $\boldsymbol{L}(\boldsymbol{A}^g) = \boldsymbol{L}(\boldsymbol{A}) - \boldsymbol{L}(\boldsymbol{A}^c)$ and $\boldsymbol{D}(\boldsymbol{A}^g) = \boldsymbol{D}(\boldsymbol{A}) - \boldsymbol{D}(\boldsymbol{A}^c)$, and after rearranging the terms, we obtain

$$\begin{aligned}\lambda_i^g \approx\ & \underbrace{\lambda_i + \boldsymbol{y}_i^T \cdot ((\boldsymbol{L}(\boldsymbol{A}) - \boldsymbol{L}(\boldsymbol{A}^t)) - \lambda_i \cdot (\boldsymbol{D}(\boldsymbol{A}) - \boldsymbol{D}(\boldsymbol{A}^t))) \cdot \boldsymbol{y}_i}_{=:c_i} \\ & - \underbrace{\boldsymbol{y}_i^T \cdot ((\boldsymbol{L}(\boldsymbol{A}^c)) - \lambda_i \cdot (\boldsymbol{D}(\boldsymbol{A}^c)) \cdot \boldsymbol{y}_i}_{=:g_i}\end{aligned} \tag{B.6}$$

Since c_i is constant, minimizing λ_i^g is equivalent to maximizing g_i. Simplifying yields:

$$g_i = \boldsymbol{y}_i^T \cdot \boldsymbol{L}(\boldsymbol{A}^c) \cdot \boldsymbol{y}_i - \lambda_i \cdot \boldsymbol{y}_i^T \cdot \boldsymbol{D}(\boldsymbol{A}^c) \cdot \boldsymbol{y}_i = \sum_{jj'} \frac{1}{2} \boldsymbol{A}^c_{jj'} (\boldsymbol{y}_{ij} - \boldsymbol{y}_{ij'})^2 - \lambda_i \sum_j \boldsymbol{y}_{ij}^2 \cdot d_j^c \tag{B.7}$$

where $d_j^c = [\boldsymbol{D}(\boldsymbol{A}^c)]_{jj} = \sum_{j'} \boldsymbol{A}^c_{jj'}$. Thus

$$g_i = \sum_{jj'} \frac{1}{2} \boldsymbol{A}^c_{jj'} (\boldsymbol{y}_{ij} - \boldsymbol{y}_{ij'})^2 - \lambda_i \boldsymbol{y}_{ij}^2 \boldsymbol{A}^c_{jj'} = \sum_{jj'} \boldsymbol{A}^c_{jj'} \left(\frac{1}{2} (\boldsymbol{y}_{ij} - \boldsymbol{y}_{ij'})^2 - \lambda_i \boldsymbol{y}_{ij}^2 \right) \tag{B.8}$$

and exploiting the symmetry of the graph, we obtain

$$g_i = \sum_{(j,j')\in\mathcal{E}} \boldsymbol{A}^c_{jj'} \left((\boldsymbol{y}_{ij} - \boldsymbol{y}_{ij'})^2 - \lambda_i \boldsymbol{y}_{ij}^2 - \lambda_i \boldsymbol{y}_{ij'}^2 \right)$$

Since the overall goal is to minimize $\sum_{i=1}^k \lambda_i^g$, we aim at maximizing

$$\begin{aligned}\sum_{i=1}^k g_i &= \sum_{i=1}^k \sum_{(j,j')\in\mathcal{E}} \boldsymbol{A}^c_{jj'} \left((\boldsymbol{y}_{ij} - \boldsymbol{y}_{ij'})^2 - \lambda_i \boldsymbol{y}_{ij}^2 - \lambda_i \boldsymbol{y}_{ij'}^2 \right) \\ &= \sum_{(j,j')\in\mathcal{E}} \boldsymbol{A}^c_{jj'} \left(\sum_{i=1}^k (\boldsymbol{y}_{ij} - \boldsymbol{y}_{ij'})^2 - \sum_{i=1}^k \lambda_i \boldsymbol{y}_{ij}^2 - \sum_{i=1}^k \lambda_i \boldsymbol{y}_{ij'}^2 \right)\end{aligned}$$

By noticing that $\boldsymbol{y}_{ij} = h_{ji}$ we obtain

$$= \sum_{(j,j')\in\mathcal{E}} \boldsymbol{A}^c_{jj'} \left(\underbrace{\|\boldsymbol{h}_j - \boldsymbol{h}_{j'}\|_2^2 - \|\sqrt{\boldsymbol{\lambda}} \circ \boldsymbol{h}_j\|_2^2 - \|\sqrt{\boldsymbol{\lambda}} \circ \boldsymbol{h}_{j'}\|_2^2}_{x} \right)$$

Note that some of the terms x might be negative. Clearly, since we aim to maximize the equation – and since $\boldsymbol{A}^c_{ij} \geq 0$ – for these terms we have to choose $\boldsymbol{A}^c_{ij} = 0$. For the remaining (non-negative) terms, the same arguments apply as in the proof of Lemma 3.1: i.e. they are either 0 or $\boldsymbol{A}_{ij}$. Thus, overall, for each term we have $\boldsymbol{A}^c_e \in \{0, \boldsymbol{A}_e\}$. □

Proof of Lemma 3.3. Note that $\boldsymbol{A}_{ij} = \boldsymbol{A}_{ij} - \boldsymbol{A}^c_{ij}$ and $d^g_i = d_i - d^c_i$. Let $\boldsymbol{y}_k$ be a column vector of $\boldsymbol{H}$. It holds

$$\boldsymbol{y}_k^T \cdot \boldsymbol{L}_{\text{sym}}(A^g) \cdot \boldsymbol{y}_k = \sum_{ij} \frac{1}{2} \boldsymbol{A}_{ij} \Big(\frac{\boldsymbol{y}_{ki}}{\sqrt{d^g_i}} - \frac{\boldsymbol{y}_{kj}}{\sqrt{d^g_j}}\Big)^2 = \sum_{ij} \frac{1}{2} \boldsymbol{A}_{ij} \Big(\frac{\boldsymbol{y}^2_{ki}}{d^g_i} + \frac{\boldsymbol{y}^2_{kj}}{d^g_j} - \frac{2 \cdot \boldsymbol{y}_{ki}\boldsymbol{y}_{kj}}{\sqrt{d^g_i}\sqrt{d^g_j}}\Big)$$

$$= \sum_i \frac{1}{2}\boldsymbol{y}^2_{ki} + \sum_j \frac{1}{2}\boldsymbol{y}^2_{kj} - \sum_{ij} \frac{\boldsymbol{A}_{ij}\boldsymbol{y}_{ki}\boldsymbol{y}_{kj}}{\sqrt{d^g_i}\sqrt{d^g_j}} \tag{B.9}$$

Since $\boldsymbol{y}_k$ is given, the first two terms are constant. Furthermore, due to orthogonality it holds $Tr(\boldsymbol{H}^T \boldsymbol{L}_{\text{sym}}(A^g)\boldsymbol{H}) = \sum_k \boldsymbol{y}_k^T \cdot \boldsymbol{L}_{\text{sym}}(A^g) \cdot \boldsymbol{y}_k$. Thus, minimizing the trace is equivalent to maximizing

$$\sum_k \sum_{ij} \frac{\boldsymbol{A}_{ij}\boldsymbol{y}_{ki}\boldsymbol{y}_{kj}}{\sqrt{d^g_i}\sqrt{d^g_j}} = \sum_{ij} \frac{\boldsymbol{A}_{ij}}{\sqrt{d^g_i}\sqrt{d^g_j}} \boldsymbol{h}_i \cdot \boldsymbol{h}_j^T \tag{B.10}$$

noticing that $\boldsymbol{y}_{ki} = h_{ij}$. Exploiting the graph's symmetry concludes the proof. □

Proof of Corollary 3.2. Adding $e = (i, j)$ to $\mathcal{X}$ has the following effects: the term a^c_e changes from 0 to a_e; the degree of the two incident nodes becomes $d^{\mathcal{X}\cup\{e\}}_i = d^{\mathcal{X}}_i - a_e$. Therefore,

$$f_3(\boldsymbol{v}^{\mathcal{X}\cup\{e\}}) = f_3(\boldsymbol{v}^{\mathcal{X}}) - \frac{p_e}{\sqrt{d^{\mathcal{X}}_i} \cdot \sqrt{d^{\mathcal{X}}_j}} - \sum_{\substack{(x,y)\in(\mathcal{E}_i\cup\mathcal{E}_j)\setminus\mathcal{X} \\ (x,y)\neq(i,j)}} \frac{p_{xy}}{\sqrt{d^{\mathcal{X}}_x} \cdot \sqrt{d^{\mathcal{X}}_x}} \tag{B.11}$$

$$+ \sum_{\substack{x\neq j \\ (i,x)\in\mathcal{E}_i\setminus\mathcal{X} \\ \vee(x,i)\in\mathcal{E}_i\setminus\mathcal{X}}} \frac{p_{ix}}{\sqrt{d^{\mathcal{X}}_i - a_e}\sqrt{d^{\mathcal{X}}_x}} + \sum_{\substack{x\neq i \\ (j,x)\in\mathcal{E}_j\setminus\mathcal{X} \\ \vee(x,j)\in\mathcal{E}_j\setminus\mathcal{X}}} \frac{p_{xj}}{\sqrt{d^{\mathcal{X}}_j - a_e}\sqrt{d^{\mathcal{X}}_x}}$$

$$= f_3(\boldsymbol{v}^{\mathcal{X}}) + s(i, a_e, \mathcal{X}) + s(j, a_e, \mathcal{X}) + \delta(e, \mathcal{X}) = f_3(\boldsymbol{v}^{\mathcal{X}}) + \Delta(e, \mathcal{X})$$

Since $\mathcal{X}$ is given, $f_3(\boldsymbol{v}^{\mathcal{X}})$ is constant. Thus, the edge $e \in \mathcal{E}'$ maximizing $f_3(\boldsymbol{v}^{\mathcal{X}\cup\{e\}})$ is found by maximizing $\Delta(e, \mathcal{X})$. □

C Deep Gaussian Embedding of Graphs

C.1 Proofs

Proof (Proposition 4.1). Since both $\mathcal{L}$ and $\mathcal{L}_s$ are summing over i it is sufficient to show that the losses are equal in expectation for a single node i. Denoting with $\mathcal{L}_s^{(i)}$ the loss for a single node i and with $E_{i,k,k'} = E_{ij_k}^2 + \exp^{-E_{ij_{k'}}}$ for notational convenience we have:

$$
\begin{aligned}
\mathcal{L}_s^{(i)} =& \mathbb{E}_{(j_1,\dots,j_K)\sim(\mathcal{V}_{i1},\dots,\mathcal{V}_{iK})} \sum_{k<k'} |\mathcal{V}_{ik}| \cdot |\mathcal{V}_{ik'}| \cdot E_{i,k,k'} \\
\overset{(1)}{=}& \mathbb{E}_{(j_1,\dots,j_K)\sim(\mathcal{V}_{i1},\dots,\mathcal{V}_{iK})} |\mathcal{V}_{i1}| \cdot |\mathcal{V}_{i2}| \cdot E_{i,1,2} \\
& + \dots + \mathbb{E}_{(j_1,\dots,j_K)\sim(\mathcal{V}_{i1},\dots,\mathcal{V}_{iK})} |\mathcal{V}_{iK-1}| \cdot |\mathcal{V}_{iK}| \cdot E_{i,K-1,K} \\
\overset{(2)}{=}& \mathbb{E}_{(j_1,j_2)\sim(\mathcal{V}_{i1},\mathcal{V}_{i2})} |\mathcal{V}_{i1}| \cdot |\mathcal{V}_{i2}| \cdot E_{i,1,2} \\
& + \dots + \mathbb{E}_{(j_{K-1},j_K)\sim(\mathcal{V}_{iK-1},\mathcal{V}_{iK})} |\mathcal{V}_{iK-1}| \cdot |\mathcal{V}_{iK}| \cdot E_{i,K-1,K} \\
\overset{(3)}{=}& \sum_{j_1\in\mathcal{V}_{i1}} \sum_{j_2\in\mathcal{V}_{i2}} p(j_1)p(j_2)|\mathcal{V}_{i1}| \cdot |\mathcal{V}_{i2}| \cdot E_{i,1,2} \\
& + \dots + \sum_{j_{K-1}\in\mathcal{V}_{iK-1}} \sum_{j_K\in\mathcal{V}_{iK}} p(j_{K-1})p(j_K)|\mathcal{V}_{iK-1}| \cdot |\mathcal{V}_{iK}| \cdot E_{i,K-1,K} \qquad \text{(C.1)} \\
\overset{(4)}{=}& \frac{1}{|\mathcal{V}_{i1}|}\frac{1}{|\mathcal{V}_{i2}|}|\mathcal{V}_{i1}||\mathcal{V}_{i2}| \sum_{j_1\in\mathcal{V}_{i1}} \sum_{j_2\in\mathcal{V}_{i2}} \cdot E_{i,1,2} \\
& + \dots + \frac{1}{|\mathcal{V}_{iK-1}|}\frac{1}{|\mathcal{V}_{iK}|}|\mathcal{V}_{iK-1}||\mathcal{V}_{iK}| \sum_{j_{K-1}\in\mathcal{V}_{iK-1}} \sum_{j_K\in\mathcal{V}_{iK}} \cdot E_{i,K-1,K} \\
=& \sum_{j_1\in\mathcal{V}_{i1}} \sum_{j_2\in\mathcal{V}_{i2}} \cdot E_{i,1,2} + \dots + \sum_{j_{K-1}\in\mathcal{V}_{iK-1}} \sum_{j_K\in\mathcal{V}_{iK}} \cdot E_{i,K-1,K} \\
=& \sum_{k<k'} \sum_{j\in\mathcal{V}_{ik}} \sum_{j'\in\mathcal{V}_{ik'}} \left({E_{ij}}^2 + \beta \cdot \exp^{-E_{ij'}} \right) = \mathcal{L}
\end{aligned}
$$

In step (1) we have expanded the sum over $k < k'$ in independent terms. In step (2) we have marginalized the expectation over the variables that do not appear in the expression, e.g. for the term $\mathbb{E}_{(j_1,\dots,j_K)\sim(\mathcal{V}_{i1},\dots,\mathcal{V}_{iK})}|\mathcal{V}_{i1}| \cdot |\mathcal{V}_{i2}| \cdot E_{i12}$ we can marginalize over j_p where $p \neq 1$ and $p \neq 2$ since the term does not depend on them. In step (3) we have expanded the expectation term. In step (4) we have substituted $p(j_p)$ with $\frac{1}{|\mathcal{V}_{ij_p}|}$ since we are sampling uniformly at random. Since $\mathcal{L}_s^{(i)}$ is equal to $\mathcal{L}^{(i)}$ it follows that $\nabla\mathcal{L}_s$ based on a set of samples is an unbiased estimate of $\nabla\mathcal{L}$. $\square$

C.2 Implementation Details

Architecture and hyperparameters. We observed that Graph2Gauss is not sensitive to the choice of hyperparameters such as number and size of hidden layers. Better yet, as shown in Sec. 4.4.4, Graphs2Gauss is also not sensitive to the size of the embedding size L. Thus, for a new graph, one can simply pick a relatively large embedding size and if required prune it later similarly to the analysis performed in Fig. 4.7c.

As a sensible default we recommend an encoder with a single hidden layer of size $s_1 = 512$. More specifically, to obtain the embeddings for a node i we have

$$\boldsymbol{h}_i = \mathrm{relu}(\boldsymbol{X}_i \boldsymbol{W} + \boldsymbol{b}) \qquad \boldsymbol{\mu}_i = \boldsymbol{h}_i \boldsymbol{W}_\mu + \boldsymbol{b}_\mu \qquad \boldsymbol{\sigma}_i = \mathrm{elu}(\boldsymbol{h}_i \boldsymbol{W}_\Sigma + \boldsymbol{b}_\Sigma) + 1$$

where $\boldsymbol{X}_i$ are node attributes, relu and elu are the rectified linear unit and exponential linear unit respectively. In practice, we found that the softplus works equally well as the elu for making sure that $\boldsymbol{\sigma}_i$ are positive and in turn $\boldsymbol{\Sigma}_i$ is positive-definite. We used Xavier initialization [252] for the weight matrices $\boldsymbol{W} \in \mathbb{R}^{D \times s_1}$, $\boldsymbol{b} \in \mathbb{R}^{s_1}$, $\boldsymbol{W}_\mu \in \mathbb{R}^{s_1 \times L/2}$, $\boldsymbol{b}_\mu \in \mathbb{R}^{L/2}$, $\boldsymbol{W}_\Sigma \in \mathbb{R}^{s_1 \times L/2}$, $\boldsymbol{b}_\Sigma \in \mathbb{R}^{L/2}$. As discussed in Sec. 4.3.4, multiple hidden layers, or other architectures such as CNNs/RNNs can also be used based on the specific problem.

Unlike other approaches using Gaussian embeddings [102–104] we do not explicitly regularize the norm of the means and we do not clip the covariance matrices. Given the self-regularizing nature of the KL divergence this is unnecessary, as was confirmed in our experiments. The parameters are optimized using Adam [106] with a fixed learning rate of 0.001 and no learning rate annealing/decay.

Edge cover. Some of the methods such as node2vec [87] are not able to produce an embedding for nodes that have not been seen during training. Therefore, it is important to make sure that during the train/validation/test split of the edge set, every node appears at least once in the train set. Random sampling of the edges does not guarantee this, especially when allocating a low percentage of edges in the train set during the split. To guarantee that every node appears at least once in the train set we have to find an edge cover. An edge cover of a graph is a set of edges such that every node of the graph is incident to at least one edge of the set. The minimum edge cover problem is the problem of finding an edge cover of minimum size. The dashed line in Fig. 4.4c indicates exactly the size of the minimum edge cover. This condition had to be satisfied for the competing methods, however, since Graph2Gauss is inductive, it does not require that every node is in the train set.

D Adversarial Attacks on Node Embeddings

D.1 Proofs

Proof. **Theorem 5.1.** Applying eigenvalue perturbation theory we obtain that $\lambda'_p = \lambda_p + \boldsymbol{u}_p^T(\Delta\hat{\boldsymbol{M}})\boldsymbol{u}_p$ where λ'_p is the eigenvalue of $\hat{\boldsymbol{M}}'$ based on $\boldsymbol{A}'$ obtained after perturbing the graph. Using the fact that $\lambda_p = \boldsymbol{u}_p^T\hat{\boldsymbol{M}}\boldsymbol{u}_p$, and the fact that singular values are equal to the absolute value of the corresponding eigenvalues we obtain the desired result. □

Proof. **Theorem 5.2.** Denote with $\boldsymbol{e}_i$ the vector of all zeros and a single one at position i. Then, we have $\Delta\boldsymbol{A} = \Delta w_{ij}(\boldsymbol{e}_i\boldsymbol{e}_j^T + \boldsymbol{e}_j\boldsymbol{e}_i^T)$ and $\Delta\boldsymbol{D} = \Delta w_{ij}(\boldsymbol{e}_i\boldsymbol{e}_i^T + \boldsymbol{e}_j\boldsymbol{e}_j^T)$. From eigenvalue perturbation theory [39], we get: $\lambda'_y \approx \lambda_y + \boldsymbol{u}_y^T(\Delta\boldsymbol{A} - \lambda_y\Delta\boldsymbol{D})\boldsymbol{u}_y$. Substituting $\Delta\boldsymbol{A}$ and $\Delta\boldsymbol{D}$ concludes the proof. □

We include an immediate result to prove Theorem 5.3.

Lemma D.1. *Consider the generalized eigenvalue problem $\boldsymbol{A}\boldsymbol{u} = \lambda\boldsymbol{D}\boldsymbol{u}$ and suppose that we have the changes in the respective matrices/vectors: $\Delta\boldsymbol{A}, \Delta\boldsymbol{D}$ and $\Delta\lambda$, then the approximate change in the eigenvectors $\Delta\boldsymbol{u}$ can be expressed as:*

$$\Delta\boldsymbol{u} = -(\boldsymbol{A} - \lambda\boldsymbol{D})^+\Big(\Delta\boldsymbol{A} - \Delta\lambda\boldsymbol{D} - \lambda\Delta\boldsymbol{D}\Big)\boldsymbol{u}$$

Proof. **Lemma D.1.** We start from $(\boldsymbol{A}+\Delta\boldsymbol{A})(\boldsymbol{u}+\Delta\boldsymbol{u}) = (\lambda+\Delta\lambda)(\boldsymbol{D}+\Delta\boldsymbol{D})(\boldsymbol{u}+\Delta\boldsymbol{u})$. Ignoring second-order terms and simplifying using $\boldsymbol{A}\boldsymbol{u} = \lambda\boldsymbol{D}\boldsymbol{u}$ we get

$$\boldsymbol{A}\Delta\boldsymbol{u} + \Delta\boldsymbol{A}\boldsymbol{u} = \lambda\Delta\boldsymbol{D}\boldsymbol{u} + \Delta\lambda\boldsymbol{D}\boldsymbol{u} + \lambda\boldsymbol{D}\Delta\boldsymbol{u}$$

Collecting the terms for $\boldsymbol{u}$ and $\Delta\boldsymbol{u}$ we get

$$(\boldsymbol{A} - \lambda\boldsymbol{D})\Delta\boldsymbol{u} = -(\Delta\boldsymbol{A} - \Delta\lambda\boldsymbol{D} - \lambda\Delta\boldsymbol{D})\boldsymbol{u}$$

Since the matrix $(\boldsymbol{A} - \lambda\boldsymbol{D})$ is singular we multiply with its Moore-Penrose pseudoinverse $(\boldsymbol{A} - \lambda\boldsymbol{D})^+$ to obtain the final result. Here $(\boldsymbol{A} - \lambda\boldsymbol{D})^+(\boldsymbol{A} - \lambda\boldsymbol{D})$ is not the identity matrix, but the projection matrix onto the orthogonal complement of $\ker(\boldsymbol{A} - \lambda\boldsymbol{D})$. Thus, since the respective eigenvector is not uniquely determined, we also choose $\Delta\boldsymbol{u}$ to be orthogonal to the eigenspace $\ker(\boldsymbol{A} - \lambda\boldsymbol{D})$. □

Proof. **Theorem 5.3.** Let $\Delta\boldsymbol{A}$ and $\Delta\boldsymbol{D}$ be defined as in Theorem 5.2 and let $\Delta\lambda$ be the change in the eigenvalues as computed in Theorem 5.2. Plugging these terms in Lemma D.1 and simplifying we obtain the result. □

We include an intermediate result to prove Lemma 5.2 and Lemma 5.3.

Lemma D.2. *λ is an eigenvalue of $\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2} := \boldsymbol{A}_{norm}$ with eigenvector $\hat{\boldsymbol{u}} = \boldsymbol{D}^{1/2}\boldsymbol{u}$ if and only if λ and $\boldsymbol{u}$ solve the generalized eigen-problem $\boldsymbol{A}\boldsymbol{u} = \lambda\boldsymbol{D}\boldsymbol{u}$.*

Proof. **Lemma D.2.** We have $\boldsymbol{A}\boldsymbol{z} = \lambda\boldsymbol{D}\boldsymbol{z} \implies (\boldsymbol{Q}^{-1}\boldsymbol{A}\boldsymbol{Q}^{-T})(\boldsymbol{Q}^T\boldsymbol{z}) = \lambda(\boldsymbol{Q}^T\boldsymbol{z})$ for any real symmetric $\boldsymbol{A}$ and any positive-definite $\boldsymbol{D}$, where $\boldsymbol{D} = \boldsymbol{Q}\boldsymbol{Q}^T$ using the Cholesky factorization. Substituting the adjacency/degree matrix and using $\boldsymbol{Q} = \boldsymbol{Q}^T = \boldsymbol{D}^{1/2}$ we obtain the result. □

Proof. **Lemma 5.2.** From Qiu et al. [146] we have $\boldsymbol{S} = \boldsymbol{D}^{-1/2}\big(\hat{\boldsymbol{U}}\big(\sum_{r=1}^{T}\hat{\boldsymbol{\Lambda}}^r\big)\hat{\boldsymbol{U}}^T\big)\boldsymbol{D}^{-1/2}$ where $\hat{\boldsymbol{U}}\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{U}}^T = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2} =: \boldsymbol{A}_{\text{norm}}$ is the eigenvalue decomposition of $\boldsymbol{A}_{\text{norm}}$. From Lemma D.2 we have that λ is an eigenvalue of $\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ with eigenvector $\hat{\boldsymbol{u}} = \boldsymbol{D}^{1/2}\boldsymbol{u}$ if and only if λ and $\boldsymbol{u}$ solve the generalized eigen-problem $\boldsymbol{A}\boldsymbol{u} = \lambda\boldsymbol{D}\boldsymbol{u}$. Substitute $\hat{\boldsymbol{\Lambda}} = \boldsymbol{\Lambda}$ and $\hat{\boldsymbol{U}} = \boldsymbol{D}^{1/2}\boldsymbol{U}$ in $\boldsymbol{S}$, and use the fact that $\boldsymbol{D}$ is diagonal. □

Proof. **Lemma 5.3.** Following Qiu et al. [146], the singular values of $\boldsymbol{S}$ can be bounded by $\sigma_p(\boldsymbol{S}) \leq \frac{1}{d_{\min}}\big|\sum_{r=1}^{T}(\hat{\mu}_{\pi(p)})^r\big|$ where μ are the (standard) eigenvalues of $\boldsymbol{A}_{\text{norm}}$. Using Lemma D.2, the same bound applies using the generalized eigenvalues λ_p of $\boldsymbol{A}$. Now using Theorem 5.2, we obtain $\tilde{\lambda}'_p$ an approximation of the p-th *generalized* eigenvalue of $\boldsymbol{A}'$. Plugging it into the singular value bound we obtain: $\sigma_p(\boldsymbol{S}) \leq \frac{1}{d_{\min}}\big|\sum_{r=1}^{T}(\tilde{\lambda}'_{\pi(p)})^r\big|$ which concludes the proof. □

Note that the permutation π does not need be computed explicitly. In practice, for every $\tilde{\lambda}'_p$, we compute the term $\big|\sum_{r=1}^{T}(\tilde{\lambda}'_p)^r\big|$. Afterwards, these terms are simply sorted.

D.2 Analysis of Spectral Embedding Methods

Attacking spectral embedding. Finding the spectral embedding is equivalent to the following trace minimization problem:

$$\min_{\boldsymbol{Z}\in\mathbb{R}^{|V|\times K}} Tr(\boldsymbol{Z}^T\boldsymbol{L}_{\text{xy}}\boldsymbol{Z}) = \sum_{i=1}^{K}\lambda_i(\boldsymbol{L}_{\text{xy}}) = \mathcal{L}_{\text{SC}} \tag{D.1}$$

subject to orthogonality constraints, where $\boldsymbol{L}_{\text{xy}}$ is the graph Laplacian. The solution is obtained via the eigen-decomposition of $\boldsymbol{L}$, with $\boldsymbol{Z}^* = \boldsymbol{U}_K$ where $\boldsymbol{U}_K$ are the K-first eigenvectors corresponding to the K-smallest eigenvalues λ_i. The Laplacian is typically defined in three different ways: the unnormalized Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$, the normalized random walk Laplacian $\boldsymbol{L}_{\text{rw}} = \boldsymbol{D}^{-1}\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A}$ and the normalized symmetric Laplacian $\boldsymbol{L}_{\text{sym}} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2} = \boldsymbol{I} - \boldsymbol{A}_{\text{norm}}$, where $\boldsymbol{A}, \boldsymbol{D}, \boldsymbol{A}_{\text{norm}}$ are defined as before.

Lemma D.3 ([147])**.** *λ is an eigenvalue of $\boldsymbol{L}_{rw}$ with eigenvector $\boldsymbol{u}$ if and only if λ is an eigenvalue of $\boldsymbol{L}_{sym}$ with eigenvector $\boldsymbol{w} = \boldsymbol{D}^{1/2}\boldsymbol{u}$. Furthermore, λ is an eigenvalue of $\boldsymbol{L}_{rw}$ with eigenvector $\boldsymbol{u}$ if and only if λ and $\boldsymbol{u}$ solve the generalized eigen-problem $\boldsymbol{L}\boldsymbol{u} = \lambda\boldsymbol{D}\boldsymbol{u}$.*

From Lemma D.3 we see that we can attack both normalized versions of the graph Laplacian with a single attack strategy since they have the same eigenvalues. It also helps us to do that efficiently similar to our previous analysis (Theorem 5.3).

Theorem D.1. *Let $\boldsymbol{L}_{rw}$ (or equivalently $\boldsymbol{L}_{sym}$) be the initial graph Laplacian before performing a flip and λ_y and $\boldsymbol{u}_y$ be any eigenvalue and eigenvector of $\boldsymbol{L}_{rw}$. The eigenvalue λ'_y of $\boldsymbol{L}'_{rw}$ obtained after flipping a single edge (i,j) is*

$$\lambda'_y \approx \lambda_y + \Delta w_{ij}((\boldsymbol{u}_{yi} - \boldsymbol{u}_{yj})^2 - \lambda_y(\boldsymbol{u}_{yi}^2 + \boldsymbol{u}_{yj}^2)) \tag{D.2}$$

where $\boldsymbol{u}_{yi}$ is the i-th entry of the vector $\boldsymbol{u}_y$.

Proof. From Lemma D.3 we can estimate the change in $\boldsymbol{L}_{\mathrm{rw}}$ (or equivalently $\boldsymbol{L}_{\mathrm{sym}}$) by estimating the eigenvalues solving the generalized eigen-problem $\boldsymbol{L}\boldsymbol{u} = \lambda \boldsymbol{D}\boldsymbol{u}$. Let $\Delta \boldsymbol{L} = \boldsymbol{L}' - \boldsymbol{L}$ be the change in the unnormalized graph Laplacian after performing a single edge flip (i,j) and $\Delta \boldsymbol{D}$ be the corresponding change in the degree matrix. Let $\boldsymbol{e}_i$ be defined as before. Then $\Delta \boldsymbol{L} = (1 - 2\boldsymbol{A}_{ij})(\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T$ and $\Delta \boldsymbol{D} = (1 - 2\boldsymbol{A}_{ij})(\boldsymbol{e}_i\boldsymbol{e}_i^T + \boldsymbol{e}_j\boldsymbol{e}_j^T)$. Based on the theory of eigenvalue perturbation we have $\lambda'_y \approx \lambda_y + \boldsymbol{u}_y^T(\Delta \boldsymbol{L} - \lambda_y \Delta \boldsymbol{D})\boldsymbol{u}_y$. Finally, we substitute $\Delta \boldsymbol{L}$ and $\Delta \boldsymbol{D}$. □

Using now Theorem D.1 and Eq. D.1 we finally estimate the loss of the spectral embedding after flipping an edge $\mathcal{L}_{\mathrm{SC}}(\boldsymbol{L}'_{\mathrm{rw}}, \boldsymbol{Z}) \approx \sum_{p=1}^{K} \lambda'_p$. Note that here we are summing over the K-first *smallest* eigenvalues. We see that spectral embedding and the random-walk-based approaches are indeed very similar.

Theorem D.2. *Let $\boldsymbol{L}$ be the initial unnormalized graph Laplacian before performing a flip and λ_y and $\boldsymbol{u}_y$ be any eigenvalue and eigenvector of $\boldsymbol{L}$. The eigenvalue λ'_y of $\boldsymbol{L}'$ obtained after flipping a single edge (i,j) can be approximated by:*

$$\lambda'_y \approx \lambda_y - (1 - 2\boldsymbol{A}_{ij})(\boldsymbol{u}_{yi} - \boldsymbol{u}_{yj})^2 \tag{D.3}$$

Proof. Let $\Delta \boldsymbol{A} = \boldsymbol{A}' - \boldsymbol{A}$ be the change in the adjacency matrix after performing a single edge flip (i,j) and $\Delta \boldsymbol{D}$ be the corresponding change in the degree matrix. Let $\boldsymbol{e}_i$ be defined as before. Then $\Delta \boldsymbol{L} = \boldsymbol{L}' - \boldsymbol{L} = (\boldsymbol{D} + \Delta \boldsymbol{D}) - (\boldsymbol{A} + \Delta \boldsymbol{A}) - (\boldsymbol{D} - \boldsymbol{A}) = \Delta \boldsymbol{D} - \Delta \boldsymbol{A} = (1 - 2\boldsymbol{A}_{ij})(\boldsymbol{e}_i\boldsymbol{e}_i^T + \boldsymbol{e}_j\boldsymbol{e}_j^T - (\boldsymbol{e}_i\boldsymbol{e}_j^T + \boldsymbol{e}_j\boldsymbol{e}_i^T))$. Based on the theory of eigenvalue perturbation we have $\lambda'_y \approx \lambda_y + \boldsymbol{u}_y^T(\Delta \boldsymbol{L})\boldsymbol{u}_y$. Substituting $\Delta \boldsymbol{L}$ and rearranging we get the above results. □

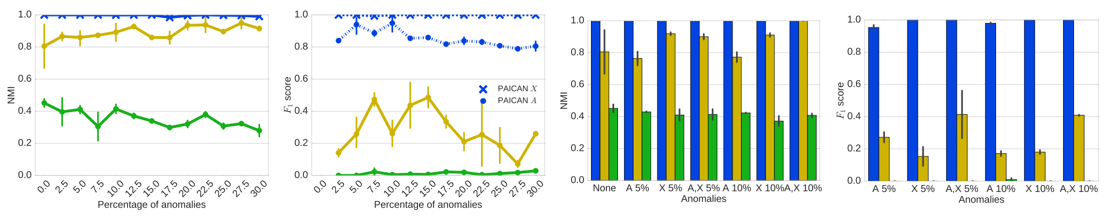
E Robust Attributed Graph Clustering

E.1 Further Experiments and Details

Experimental setup. For all methods we provide the true number K of clusters to detect. CODA is a nondeterministic algorithm that is highly sensitive to initialization and parameter choice. Following [154], we tried the values of $\{0.05, 0.1, 0.5\}$ for λ, perform several restarts for each of them and report only the highest NMI achieved. Moreover, CODA requires the percentage of anomalous nodes, for which we provide the true values. FocusCO is a semi-supervised approach; we have to provide example nodes that belong to the same single cluster. We pick the best possible scenario, i.e. we run FocusCO K number of times and we provide all nodes from a given cluster. We then pick the run which gave us the highest NMI. SIAN and LSBM are also executed multiple times and we pick the solution achieving highest NMI. For our approach, we simply perform several restarts with random initialization and pick the one that gives us the highest likelihood. PICS and BAGC are deterministic.

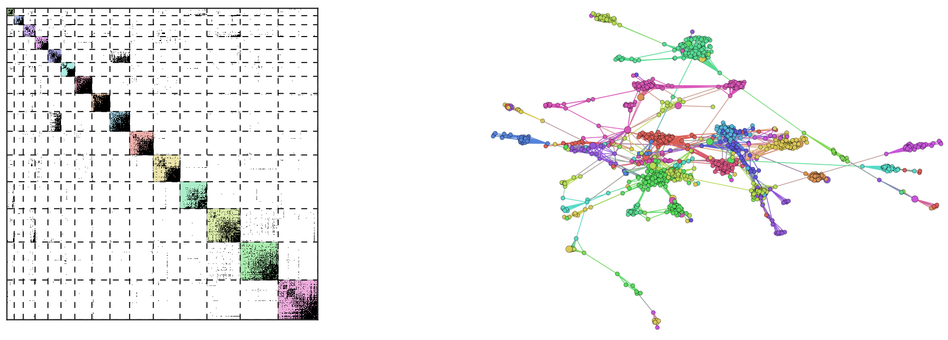
Blocky clusters: Robustness and anomaly detection. In Fig. E.1a we see that CODA performs relatively well achieving high NMI score for blocky clusters even though it can not detect all the anomalies as shown in Fig. E.1b. FocusCO although performs better compared to the power-law degree distributed graphs still has poor performance in comparison. PAICAN consistently outperforms both methods. We can draw similar conclusions for the case when we are generating either only graph anomalies (A), attribute anomalies (X), or complete anomalies (A, X) shown in Fig. E.1c and Fig. E.1d.

Case study: Connectivity patterns. We run our method on the SocialPapers and inspect the inferred block connectivity patterns $\boldsymbol{\eta}$. The nature of this dataset yields non-trivial block structure where some clusters have significant number of edges between each other. In other words we have relatively high values on the off-diagonals of $\boldsymbol{\eta}$ rather than just on the diagonal. For example our method detects a cluster where most of the



(a) NMI **(b)** F_1 score **(c)** NMI (Partial) **(d)** F_1 score (Partial)

Figure E.1: Clustering and anomaly detection performance when increasing the percentage of anomalies on synthetic data with blocky clusters.



(a) Adjacency matrix **(b)** Graph

Figure E.2: Clustering in the Amazon co-purchase graph. Colors indicate clusters. Bigger nodes indicate partial graph anomalies.

papers are from the "neurology" subject. We observe that this cluster connects to two other clusters with main subjects 'diagnostic imaging' and "psychiatry" respectively. These connection patterns are coherent and indicative of users who tend to mention papers from the broader area of neuropsychiatry. Similarly we discover a cluster of papers about "audiology" with preferred connections to a cluster about "speech language pathology".

Case study: Clustering. The adjacency matrix in Fig. E.2a reveals that the network has mostly typical clusters – with most edges within the clusters and few edges between the clusters. Notable is one off-diagonal entry between clusters $C9$ and $C5$. Interestingly, this entry describes co-purchase behavior between the cluster with topic 'playsets, toys, actionfigures' and the cluster 'clothing, bags', indicating co-buying behavior of families where products for their kids are bought together with other products. PAICAN can easily detect such kind of network topology. The graph embedding in Fig. E.2b visually confirms the good clustering structure and shows that PAICAN can easily handle non-trivial degree distributions. We also observe partial graph anomalies – marked bigger in size – connecting to several unrelated (according to $\boldsymbol{\eta}$) clusters.

Case study: Partial anomalies. We run our method on the DBLP dataset. Since the data is too large to visualize and we have no anomaly ground truth to evaluate the validity of our results we pick some of the detected partial anomalous nodes and inspect their ego network and attributes. In Fig. E.3 we show the ego network for a node that has been marked partially anomalous in attribute space corresponding to Srinivasan Parthasarathy. As we can see from the ego-network he fits nicely in graph space since most of his neighbors belong to the same cluster. However, as we discussed in Sec. 6.5 he is an obvious anomaly w.r.t. attribute space. Overall, all these case studies indicate that PAICAN is able to extract interesting patterns from attributed graphs.

E.2 Proofs and Derivations

Terms of the ELBO. Given our model, the ELBO decomposes as follows:

$$\mathcal{L} = \underbrace{\mathbb{E}_q[\log p(A|\boldsymbol{z}, \boldsymbol{c}, \boldsymbol{\eta}, \eta_{\text{bg}}, \eta_{\text{bb}}, \boldsymbol{\theta}, \widetilde{\boldsymbol{\theta}})]}_{:=\mathcal{L}_A} + \underbrace{\mathbb{E}_q[\log p(X|\boldsymbol{z}, \boldsymbol{c}, \boldsymbol{T})]}_{:=\mathcal{L}_X}$$
$$+\mathbb{E}_q[\log p(\boldsymbol{z}|\boldsymbol{c}, \boldsymbol{\pi})] + \mathbb{E}_q[\log p(\boldsymbol{c}|\boldsymbol{\rho})] - \mathbb{E}_q[\log q(\boldsymbol{z}, \boldsymbol{c})]$$

The last four terms are straightforward and can all be evaluated in linear time w.r.t. the number of nodes and dimensions.

$$\mathbb{E}_q[\log p(\boldsymbol{c}|\boldsymbol{\rho})] = \sum_i \sum_{m=0}^{3} \boldsymbol{\phi}_{im} \log(\rho_m)$$
$$\mathbb{E}_q[\log p(\boldsymbol{z}|\boldsymbol{c}, \boldsymbol{\pi})] = \sum_i \sum_k \boldsymbol{\psi}_{ik}(1 - \boldsymbol{\phi}_{i3}) \log(\pi_k)$$
$$\mathbb{E}_q[\log q(\boldsymbol{z}, \boldsymbol{c})] = \sum_i \sum_k \boldsymbol{\psi}_{ik} \log(\boldsymbol{\psi}_{ik}) + \sum_i \sum_{m=0}^{3} \boldsymbol{\phi}_{im} \log(\boldsymbol{\phi}_{im})$$
$$\mathcal{L}_X = \sum_i \sum_k \boldsymbol{\psi}_{ik} \boldsymbol{\phi}_{i0}^X \left(\sum_d \boldsymbol{X}_{id} \log(\boldsymbol{T}_{dk}) + (1 - \boldsymbol{X}_{id}) \log(1 - \boldsymbol{T}_{dk}) \right) + \sum_i \boldsymbol{\phi}_{i1}^X D \log(0.5)$$

Proof of Eq. 6.7. We will show that $\sum_i \boldsymbol{\phi}_{i0}^A \theta_i \sum_l \boldsymbol{\psi}_{il} \eta_{kl} = 1, \forall k$. We are using this identity only in the E-step of our variational EM. That means we can substitute our MLE solution for the parameters that we got in the M-step. We start by plugging in the solution for η_{kl} and substituting D_k^G, we get:

$$\sum_i \boldsymbol{\phi}_{i0}^A \theta_i \sum_l \boldsymbol{\psi}_{il} \eta_{kl} = \sum_l \boldsymbol{\psi}_{il} \sum_i \boldsymbol{\phi}_{i0}^A \theta_i \frac{m_{kl}}{D_k^G D_l^G} =$$
$$\sum_l \sum_i \boldsymbol{\phi}_{i0}^A \theta_i \boldsymbol{\psi}_{il} \frac{m_{kl}}{\left(\sum_i \boldsymbol{\phi}_{i0}^A \theta_i \boldsymbol{\psi}_{ik} \right) \left(\sum_i \boldsymbol{\phi}_{i0}^A \theta_i \boldsymbol{\psi}_{il} \right)} = \frac{\sum_l m_{kl}}{\sum_i \boldsymbol{\phi}_{i0}^A \theta_i \boldsymbol{\psi}_{ik}}$$

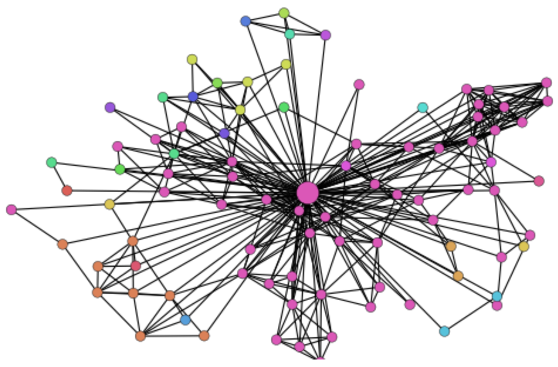


Figure E.3: Ego graph of a partial anomaly in attribute space for DBLP. Colors indicate clusters.

Substituting θ_i and m_{kl} and taking advantage of $\sum_k \boldsymbol{\psi}_k C = C$ for any constant C:

$$\frac{\sum_l m_{kl}}{\sum_i \boldsymbol{\phi}_{i0}^A \theta_i \boldsymbol{\psi}_{ik}} = \frac{\sum_l \sum_{i \neq j} \boldsymbol{A}_{ij} \boldsymbol{\phi}_{i0}^A \boldsymbol{\phi}_{j0}^A \boldsymbol{\psi}_{ik} \boldsymbol{\psi}_{jl}}{\sum_i \boldsymbol{\phi}_{i0}^A (\sum_{j \neq i} \boldsymbol{A}_{ij} \boldsymbol{\phi}_{j0}^A) \boldsymbol{\psi}_{ik}} = \frac{\sum_l \boldsymbol{\psi}_{jl} \sum_{i \neq j} \boldsymbol{A}_{ij} \boldsymbol{\phi}_{i0}^A \boldsymbol{\phi}_{j0}^A \boldsymbol{\psi}_{ik}}{\sum_{i \neq j} \boldsymbol{A}_{ij} \boldsymbol{\phi}_{i0}^A \boldsymbol{\phi}_{j0}^A \boldsymbol{\psi}_{ik}} = \frac{\sum_{i \neq j} \boldsymbol{A}_{ij} \boldsymbol{\phi}_{i0}^A \boldsymbol{\phi}_{j0}^A \boldsymbol{\psi}_{ik}}{\sum_{ij} \boldsymbol{A}_{ij} \boldsymbol{\phi}_{i0}^A \boldsymbol{\phi}_{j0}^A \boldsymbol{\psi}_{ik}} = 1 \tag{E.1}$$

E-Step. Following [169] (Ch. 10) the optimal variational distribution is

$$q^*(z_i) \propto \exp(\mathbb{E}_{q \setminus z_i}[\log p(A, X, \boldsymbol{z}, \boldsymbol{c} \mid \ldots) \tag{E.2}$$

Thus, to derive the optimal variational parameters $\boldsymbol{\psi}_{ik}$ for the cluster assignments, we have to keep only the terms of the ELBO that include $\boldsymbol{\psi}_{ik}$ and disregard the constants. After rearranging we obtain (for the general case of graphs including potential self-loops):

$$\begin{aligned}\boldsymbol{\psi}_{ik}^{\text{new}} \propto \exp\Bigg(\boldsymbol{\phi}_{i0}^A \Bigg[& \sum_{j \in \mathcal{N}_i} \boldsymbol{\phi}_{j0}^A \sum_l \boldsymbol{\psi}_{jl} \log(\theta_i \theta_j \eta_{kl}) \\ & - \theta_i (1 - \theta_i \boldsymbol{\phi}_{i0}^A \sum_l \boldsymbol{\psi}_{il} \eta_{kl}) - \frac{1}{2} \theta_i^2 \eta_{kk} + \boldsymbol{A}_{ii} \log(\frac{1}{2} \theta_i^2 \eta_{kk}) \Bigg] \\ & + \boldsymbol{\phi}_{i0}^X \Bigg[\sum_d \boldsymbol{X}_{id} \log(\boldsymbol{T}_{dk}) + (1 - \boldsymbol{X}_{id}) \log(1 - \boldsymbol{T}_{dk}) \Bigg] \\ & + (1 - \boldsymbol{\phi}_{i3}) \log(\pi_k) \Bigg)\end{aligned} \tag{E.3}$$

When the graph contains no self-loops ($\boldsymbol{A}_{ii} = 0$) we obtain Eq. 6.8. Similarly, for the anomaly assignments $\boldsymbol{\phi}_{im}$, when including the self-loops, the variables $\hat{\boldsymbol{\phi}}_{i0}^A$ and $\hat{\boldsymbol{\phi}}_{i1}^A$ become:

$$\begin{aligned}\hat{\boldsymbol{\phi}}_{i0}^A = & \sum_{j \in \mathcal{N}_i} \boldsymbol{\phi}_{j0}^A \sum_{kl} \boldsymbol{\psi}_{ik} \boldsymbol{\psi}_{jl} \log(\theta_i \theta_j \eta_{kl}) - \theta_i (1 - \theta_i \boldsymbol{\phi}_{i0}^A \sum_{kl} \boldsymbol{\psi}_{ik} \boldsymbol{\psi}_{il} \eta_{kl}) \\ & + \sum_{j \in \mathcal{N}_i} \boldsymbol{\phi}_{j1}^A \log(\widetilde{\theta}_j \eta_{\text{bg}}) - \eta_{\text{bg}} (\widetilde{\theta}^B - \boldsymbol{\phi}_{i1}^A \widetilde{\theta}_i) - \frac{1}{2} \theta_i^2 \sum_k \boldsymbol{\psi}_{ik} \eta_{kk} + \sum_k \boldsymbol{\psi}_{ik} \boldsymbol{A}_{ii} \log(\frac{1}{2} \theta_i^2 \eta_{kk})\end{aligned} \tag{E.4}$$

$$\begin{aligned}\hat{\boldsymbol{\phi}}_{i1}^A = & \log(\widetilde{\theta}_i \eta_{\text{bg}}) \sum_{j \in \mathcal{N}_i} \boldsymbol{\phi}_{j0}^A - \eta_{\text{bg}} \widetilde{\theta}_i (g - \boldsymbol{\phi}_{i0}^A) + \sum_{j \in \mathcal{N}_i} \boldsymbol{\phi}_{j1}^A \log(\widetilde{\theta}_i \widetilde{\theta}_j \eta_{\text{bb}}) \\ & - \widetilde{\theta}_i \eta_{\text{bb}} (\widetilde{\theta}^B - \boldsymbol{\phi}_{i1}^A \widetilde{\theta}_i) - \frac{1}{2} \widetilde{\theta}_i^2 \eta_{\text{bb}} + \boldsymbol{A}_{ii} \log(\frac{1}{2} \widetilde{\theta}_i^2 \eta_{\text{bb}})\end{aligned} \tag{E.5}$$

ELBO Reformulation for the M-Step. We will simplify $\mathcal{L}_A$. From the definition:

$$\begin{aligned}\mathcal{L}_A = &\sum_{i<j}\sum_{kl} \boldsymbol{\psi}_{ik}\boldsymbol{\psi}_{jl}\boldsymbol{\phi}^A_{i0}\boldsymbol{\phi}^A_{j0}\boldsymbol{A}_{ij}\log(\theta_i\theta_j\eta_{kl}) - \boldsymbol{\psi}_{ik}\boldsymbol{\psi}_{jl}\boldsymbol{\phi}^A_{i0}\boldsymbol{\phi}^A_{j0}\theta_i\theta_j\eta_{kl} \\ &+ \sum_{i<j} \boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j0}\boldsymbol{A}_{ij}\log(\widetilde{\theta}_i\eta_{\text{bg}}) - \boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j0}\widetilde{\theta}_i\eta_{\text{bg}} \\ &+ \sum_{i<j} \boldsymbol{\phi}^A_{i0}\boldsymbol{\phi}^A_{j1}\boldsymbol{A}_{ij}\log(\widetilde{\theta}_j\eta_{\text{bg}}) - \boldsymbol{\phi}^A_{i0}\boldsymbol{\phi}^A_{j1}\widetilde{\theta}_j\eta_{\text{bg}} \\ &+ \sum_{i<j} \boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j1}\boldsymbol{A}_{ij}\log(\widetilde{\theta}_i\widetilde{\theta}_j\eta_{\text{bb}}) - \boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j1}\widetilde{\theta}_i\widetilde{\theta}_j\eta_{\text{bb}} \\ &+ \sum_{i}\sum_{k} \boldsymbol{\psi}_{ik}\boldsymbol{\phi}^A_{i0}\boldsymbol{A}_{ii}\log(\frac{1}{2}\theta_i^2\eta_{kk}) - \boldsymbol{\psi}_{ik}\boldsymbol{\phi}^A_{i0}\frac{1}{2}\theta_i^2\eta_{kk} \\ &+ \sum_{i} \boldsymbol{\phi}^A_{i1}\boldsymbol{A}_{ii}\log(\frac{1}{2}\widetilde{\theta}_i^2\eta_{\text{bb}}) - \boldsymbol{\phi}^A_{i1}\frac{1}{2}\widetilde{\theta}_i^2\eta_{\text{bb}}\end{aligned} \tag{E.6}$$

Looking at terms involving $\boldsymbol{\theta}$ in Eq. E.6, and taking advantage of symmetry, we write:

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\theta}} = &\sum_{i\neq j}\sum_{kl} \boldsymbol{\psi}_{ik}\boldsymbol{\psi}_{jl}\boldsymbol{\phi}^A_{i0}\boldsymbol{\phi}^A_{j0}\boldsymbol{A}_{ij}\log(\theta_i) + \frac{1}{2}\sum_{i\neq j}\sum_{kl} \boldsymbol{\psi}_{ik}\boldsymbol{\psi}_{jl}\boldsymbol{\phi}^A_{i0}\boldsymbol{\phi}^A_{j0}\boldsymbol{A}_{ij}\log(\eta_{kl}) \\ -&\frac{1}{2}\sum_{i\neq j}\sum_{kl} \boldsymbol{\psi}_{ik}\boldsymbol{\psi}_{jl}\boldsymbol{\phi}^A_{i0}\boldsymbol{\phi}^A_{j0}\theta_i\theta_j\eta_{kl} + \sum_{i}\sum_{k} \boldsymbol{\psi}_{ik}\boldsymbol{\phi}^A_{i0}\boldsymbol{A}_{ii}\log(\frac{1}{2}\theta_i^2\eta_{kk}) - \boldsymbol{\psi}_{ik}\boldsymbol{\phi}^A_{i0}\frac{1}{2}\theta_i^2\eta_{kk}\end{aligned} \tag{E.7}$$

Which we can rewrite using the definitions of m_{kl} and D^G_k:

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\theta}} = &\sum_{i} \log(\theta_i)\boldsymbol{\phi}^A_{i0}d^G_i + \frac{1}{2}\sum_{kl} m_{kl}\log(\eta_{kl}) - \frac{1}{2}D^G_kD^G_l\eta_{kl} \\ &+ \frac{1}{2}\sum_{i}\sum_{kl} \boldsymbol{\psi}_{ik}\boldsymbol{\psi}_{il}\theta_i^2\boldsymbol{\phi}^A_{i0}(\boldsymbol{\phi}^A_{i0}\eta_{kl} - \eta_{kk}) + \sum_{i} \boldsymbol{\phi}^A_{i0}\boldsymbol{A}_{ii}\log(\frac{1}{2}\theta_i^2)\end{aligned} \tag{E.8}$$

Taking advantage of symmetry we can rewrite terms involving $\widetilde{\boldsymbol{\theta}}$:

$$\begin{aligned}\mathcal{L}_{\widetilde{\boldsymbol{\theta}}} = &\sum_{i\neq j} \boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j0}\boldsymbol{A}_{ij}\log(\widetilde{\theta}_i) + \boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j0}\boldsymbol{A}_{ij}\log(\eta_{\text{bg}}) - \boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j0}\widetilde{\theta}_i\eta_{\text{bg}} \\ &+ \sum_{i\neq j} \boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j1}\boldsymbol{A}_{ij}\log(\widetilde{\theta}_i) + \frac{1}{2}\boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j1}\boldsymbol{A}_{ij}\log(\eta_{\text{bb}}) - \frac{1}{2}\boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j1}\widetilde{\theta}_i\widetilde{\theta}_j\eta_{\text{bb}} \\ &+ \sum_{i} \boldsymbol{\phi}^A_{i1}\boldsymbol{A}_{ii}\log(\frac{1}{2}\widetilde{\theta}_i^2\eta_{\text{bb}}) - \boldsymbol{\phi}^A_{i1}\frac{1}{2}\widetilde{\theta}_i^2\eta_{\text{bb}}\end{aligned} \tag{E.9}$$

Which we can then rewrite using the definitions of m_{bg}, m_{bb} as:

$$\begin{aligned}\mathcal{L}_{\widetilde{\boldsymbol{\theta}}} = &\sum_{i}\sum_{j\in\mathcal{N}_i} \boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j0}\log(\widetilde{\theta}_i) + m_{\text{bg}}\log(\eta_{\text{bg}}) - \sum_{i\neq j}\boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j0}\widetilde{\theta}_i\eta_{\text{bg}}\sum_{i}\sum_{j\in\mathcal{N}_i} \boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j1}\log(\widetilde{\theta}_i) \\ &\frac{1}{2}m_{\text{bb}}\log(\eta_{\text{bb}}) - \frac{1}{2}\sum_{i\neq j}\boldsymbol{\phi}^A_{i1}\boldsymbol{\phi}^A_{j1}\widetilde{\theta}_i\widetilde{\theta}_j\eta_{\text{bb}} + \sum_{i}\boldsymbol{\phi}^A_{i1}\boldsymbol{A}_{ii}\log(\frac{1}{2}\widetilde{\theta}_i^2\eta_{\text{bb}}) - \boldsymbol{\phi}^A_{i1}\frac{1}{2}\widetilde{\theta}_i^2\eta_{\text{bb}}\end{aligned} \tag{E.10}$$

Plugging in D^B and further simplifying we arrive at:

$$\begin{aligned}\mathcal{L}_{\widetilde{\boldsymbol{\theta}}} = &\sum_i \boldsymbol{\phi}^A_{i1} d_i \log(\widetilde{\theta}_i) + m_{\text{bg}} \log(\eta_{\text{bg}}) + \frac{1}{2} m_{\text{bb}} \log(\eta_{\text{bb}}) - \frac{1}{2} D^B D^B \eta_{\text{bb}} \\ &- D^B \sum_i (\sum_j \boldsymbol{\phi}^A_{j0} - \boldsymbol{\phi}^A_{i0}) \eta_{\text{bg}} + \frac{1}{2} \sum_i \widetilde{\theta}^2_i \eta_{\text{bb}} \boldsymbol{\phi}^A_{i1} (\boldsymbol{\phi}^A_{i1} - 1) + \sum_i \boldsymbol{\phi}^A_{i1} \boldsymbol{A}_{ii} \log(\frac{1}{2} \widetilde{\theta}^2_i)\end{aligned} \tag{E.11}$$

Joining the terms regarding $\boldsymbol{\theta}$ and $\widetilde{\boldsymbol{\theta}}$, plugging in the definition of g and rearranging we get the complete log-likelihood w.r.t. to graph space:

$$\begin{aligned}\mathcal{L}_A = &\frac{1}{2} \sum_{kl} m_{kl} \log(\eta_{kl}) - \frac{1}{2} D^G_k D^G_l \eta_{kl} + \sum_i \boldsymbol{\phi}^A_{i0} \log(\theta_i) d^G_i \\ &+ \frac{1}{2} \sum_i \sum_{kl} \boldsymbol{\psi}_{ik} \boldsymbol{\psi}_{il} \theta^2_i \boldsymbol{\phi}^A_{i0} (\boldsymbol{\phi}^A_{i0} \eta_{kl} - \eta_{kk}) + \sum_i \boldsymbol{\phi}^A_{i0} \boldsymbol{A}_{ii} \log(\frac{1}{2} \theta^2_i) \\ &+ \sum_i \boldsymbol{\phi}^A_{i1} d_i \log(\widetilde{\theta}_i) + m_{\text{bg}} \log(\eta_{\text{bg}}) + \frac{1}{2} m_{\text{bb}} \log(\eta_{\text{bb}}) - \frac{1}{2} D^B D^B \eta_{\text{bb}} \\ &- D^B \cdot g \cdot \eta_{\text{bg}} + \sum_i \widetilde{\theta}_i \boldsymbol{\phi}^A_{i1} (1 - \boldsymbol{\phi}^A_{i1}) (\eta_{\text{bg}} - \frac{1}{2} \eta_{\text{bb}}) + \sum_i \boldsymbol{\phi}^A_{i1} \boldsymbol{A}_{ii} \log(\frac{1}{2} \widetilde{\theta}^2_i)\end{aligned} \tag{E.12}$$

For the case that the observed graph has no self-loops (i.e. $\boldsymbol{A}_{ii} = 0$), we obtain the simplified Eq. 6.14.

MLE/MAP of the parameters. Before we solve for the MLE of the parameters we have to include the identifiability constraints

$$D^G_k \stackrel{!}{=} \sum_i (d^G_i + 2\boldsymbol{A}_{ii}) \boldsymbol{\psi}_{ik} \boldsymbol{\phi}^A_{i0}, \qquad D^B \stackrel{!}{=} \sum_i (d_i + 2\boldsymbol{A}_{ii}) \boldsymbol{\phi}^A_{i1} \tag{E.13}$$

Our log-likelihood function has two additional terms

$$\mathcal{L} = \mathcal{L} - \sum_k \lambda_k (D^G_k - \sum_i (d^G_i + 2\boldsymbol{A}_{ii}) \boldsymbol{\phi}_{ik} \boldsymbol{\phi}^A_{i0}) - \lambda (D^B - \sum_i (d_i + 2\boldsymbol{A}_{ii}) \boldsymbol{\phi}^A_{i1}) \tag{E.14}$$

where λ_k and λ are Lagrangian multipliers. Similarly as above, focusing on graphs with $\boldsymbol{A}_{ii} = 0$ we obtain the desired constraints.

Since θ_i are independent of each other we can find the MLE for each of them separately. Taking the terms involving θ_i and setting the derivative to 0 we get: $\frac{\partial \mathcal{L}_{\theta_i}}{\partial \theta_i} = \frac{\boldsymbol{\phi}^A_{i0} d^G_i}{\theta_i} - \sum_k \lambda_k \boldsymbol{\psi}_{ik} \boldsymbol{\phi}^A_{i0}$. Solving the $N + K$ system of equations we get the following solution: $\theta_i = d^G_i, \forall i$ and $\lambda_k = 1, \forall k$. Similarly, for $\widetilde{\theta}_i$ we have $\frac{\partial \mathcal{L}_{\widetilde{\theta}_i}}{\partial \widetilde{\theta}_i} = \frac{\boldsymbol{\phi}^A_{i1} d_i}{\widetilde{\theta}_i} - \lambda \boldsymbol{\phi}^A_{i1}$. Solving the $N + 1$ system of equations we get $\lambda = 1$ and $\widetilde{\theta}_i = d_i, \forall i$.

For the edge generating parameters we have:

$$\frac{\partial \mathcal{L}_{\eta_{kl}}}{\partial \eta_{kl}} = \frac{m_{kl}}{\eta_{kl}} - D_k^G D_l^G = 0 \implies \eta_{kl} = \frac{m_{kl}}{D_k^G D_l^G}$$
$$\frac{\partial \mathcal{L}_{\eta_{\text{bg}}}}{\partial \eta_{\text{bg}}} = \frac{m_{\text{bg}}}{\eta_{\text{bg}}} - D^B g = 0 \implies \eta_{\text{bg}} = \frac{m_{\text{bg}}}{D^B g}$$
$$\frac{\partial \mathcal{L}_{\eta_{\text{bb}}}}{\partial \eta_{\text{bb}}} = \frac{1}{2}\frac{m_{\text{bb}}}{\eta_{\text{bb}}} - \frac{1}{2} D^B D^B = 0 \implies \eta_{\text{bb}} = \frac{m_{\text{bb}}}{D^B D^B}$$

For the topics we get: $\frac{\partial \mathcal{L}_{\boldsymbol{T}_{dk}}}{\partial \boldsymbol{T}_{dk}} = \sum_i \boldsymbol{\psi}_{ik} \boldsymbol{\phi}_{i0}^X \left(\frac{\boldsymbol{X}_{id}}{\boldsymbol{T}_{dk}} - \frac{1-\boldsymbol{X}_{id}}{\boldsymbol{T}_{dk}-1} \right) = 0 \implies \boldsymbol{T}_{dk} = \frac{\sum_i r_{ik} \boldsymbol{X}_{id}}{R_k}$. For the cluster probabilities we introduce Lagrangian multipliers to enforce $\sum_k \pi_k = 1$ we get for $\pi_k = \frac{\sum_i (1-\boldsymbol{\phi}_{i3})\boldsymbol{\psi}_{ik} + \alpha_k}{\sum_i (1-\boldsymbol{\phi}_{i3}) + \sum_k \alpha_k}$. Finally, for ρ_m and also introducing Lagrangian multipliers to enforce $\sum_{m=0}^4 \rho_m = 1$ we have: $\rho_m = \frac{\sum_i \boldsymbol{\phi}_{im} + \beta_m}{N + \sum_m \beta_m}$.

E.3 Limit-case Analysis

In the following we justify the simplification of the term $\mathcal{L}_A$, by considering the limit cases when the graph grows. As we will see, both terms $\sum_i \widetilde{\theta}_i \boldsymbol{\phi}_{i1}^A (1 - \boldsymbol{\phi}_{i1}^A)(\eta_{\text{bg}} - \frac{1}{2}\eta_{\text{bb}})$ and $\frac{1}{2}\sum_i \sum_{kl} \boldsymbol{\psi}_{ik} \boldsymbol{\psi}_{il} \theta_i^2 \boldsymbol{\phi}_{i0}^A (\boldsymbol{\phi}_{i0}^A \eta_{kl} - \eta_{kk})$ become negligible. Recap the definition of $\mathcal{L}_A$:

$$\begin{aligned}
\mathcal{L}_A = &\frac{1}{2}\big(\sum_{kl} m_{kl} \log(\eta_{kl}) - D_k^G D_l^G \eta_{kl} + m_{\text{bb}} \log(\eta_{\text{bb}}) + D^B D^B \eta_{\text{bb}}\big) \\
&+ \frac{1}{2}\sum_i \sum_{kl} \boldsymbol{\psi}_{ik} \boldsymbol{\psi}_{il} \theta_i^2 \boldsymbol{\phi}_{i0}^A (\boldsymbol{\phi}_{i0}^A \eta_{kl} - \eta_{kk}) + m_{\text{bg}} \log(\eta_{\text{bg}}) - g D^B \eta_{\text{bg}} \\
&+ \sum_i \boldsymbol{\phi}_{i0}^A \log(\theta_i) d_i^G + \boldsymbol{\phi}_{i1}^A \log(\widetilde{\theta}_i) d_i + \sum_i \widetilde{\theta}_i \boldsymbol{\phi}_{i1}^A (1 - \boldsymbol{\phi}_{i1}^A)(\eta_{\text{bg}} - \frac{1}{2}\eta_{\text{bb}})
\end{aligned} \tag{E.15}$$

After carefully rearranging the terms η_{kl}, η_{bg}, and η_{bb}, we obtain:

$$\begin{aligned}
\mathcal{L}_A = &\frac{1}{2}\Bigg(\sum_{kl} m_{kl} \log(\eta_{kl}) - \sum_i \boldsymbol{\psi}_{ik} \theta_i \boldsymbol{\phi}_{i0}^A \eta_{kl} (D_l^G - \boldsymbol{\psi}_{il} \theta_i \boldsymbol{\phi}_{i0}^A) \\
&+ m_{\text{bb}} \log(\eta_{\text{bb}}) + \sum_i \widetilde{\theta}_i \boldsymbol{\phi}_{i1}^A \eta_{\text{bb}} \big(D^B - (1 - \boldsymbol{\phi}_{i1}^A)\big)\Bigg) \\
&+ \frac{1}{2}\sum_i \sum_k \boldsymbol{\psi}_{ik} \theta_i^2 \boldsymbol{\phi}_{i0}^A (-\eta_{kk}) + m_{\text{bg}} \log(\eta_{\text{bg}}) - \sum_i \widetilde{\theta}_i \boldsymbol{\phi}_{i1}^A \eta_{\text{bg}} \big(g - (1 - \boldsymbol{\phi}_{i1}^A)\big) \\
&+ \sum_i \boldsymbol{\phi}_{i0}^A \log(\theta_i) d_i^G + \boldsymbol{\phi}_{i1}^A \log(\widetilde{\theta}_i) d_i)
\end{aligned} \tag{E.16}$$

Splitting the $\sum_{kl}$ into $\sum_{k\neq l}$ and $\sum_k$, we can also rearrange the terms η_{kk}:

$$\begin{aligned}\mathcal{L}_A=&\frac{1}{2}\Bigg(\sum_{k\neq l} m_{kl}\log(\eta_{kl}) - \sum_i \boldsymbol{\psi}_{ik}\theta_i\boldsymbol{\phi}^A_{i0}\eta_{kl}(D^G_l - \boldsymbol{\psi}_{il}\theta_i\boldsymbol{\phi}^A_{i0}) + \sum_k m_{kk}\log(\eta_{kk})\\ &-\sum_i \boldsymbol{\psi}_{ik}\theta_i\boldsymbol{\phi}^A_{i0}\eta_{kk}(D^G_k - \boldsymbol{\psi}_{ik}\theta_i\boldsymbol{\phi}^A_{i0} + \theta_i) + m_{\text{bb}}\log(\eta_{\text{bb}}) + \sum_i \widetilde{\theta}_i\boldsymbol{\phi}^A_{i1}\eta_{\text{bb}}\big(D^B - (1-\boldsymbol{\phi}^A_{i1})\big)\Bigg)\\ &+ m_{\text{bg}}\log(\eta_{\text{bg}}) - \sum_i \widetilde{\theta}_i\boldsymbol{\phi}^A_{i1}\eta_{\text{bg}}\big(g - (1-\boldsymbol{\phi}^A_{i1})\big) + \sum_i \boldsymbol{\phi}^A_{i0}\log(\theta_i)d^G_i + \boldsymbol{\phi}^A_{i1}\log(\widetilde{\theta}_i)d_i)\end{aligned}$$

Which can further be written as:

$$\begin{aligned}\mathcal{L}_A=&\frac{1}{2}\Bigg(\sum_{k\neq l} m_{kl}\log(\eta_{kl}) - \sum_i \boldsymbol{\psi}_{ik}\theta_i\boldsymbol{\phi}^A_{i0}\eta_{kl}D^G_l(1 - \frac{\boldsymbol{\psi}_{il}\theta_i\boldsymbol{\phi}^A_{i0}}{D^G_l}) + \sum_k m_{kk}\log(\eta_{kk})\\ &-\sum_i \boldsymbol{\psi}_{ik}\theta_i\boldsymbol{\phi}^A_{i0}\eta_{kk}D^G_k(1 - \frac{\boldsymbol{\psi}_{ik}\theta_i\boldsymbol{\phi}^A_{i0}}{D^G_k} + \frac{\theta_i}{D^G_k}) + m_{\text{bb}}\log(\eta_{\text{bb}}) + \sum_i \widetilde{\theta}_i\boldsymbol{\phi}^A_{i1}\eta_{\text{bb}}D^B(1 - \frac{1-\boldsymbol{\phi}^A_{i1}}{D^B})\Bigg)\\ &+ m_{\text{bg}}\log(\eta_{\text{bg}}) - \sum_i \widetilde{\theta}_i\boldsymbol{\phi}^A_{i1}\eta_{\text{bg}}g(1 - \frac{1-\boldsymbol{\phi}^A_{i1}}{g}) + \sum_i \boldsymbol{\phi}^A_{i0}\log(\theta_i)d^G_i + \boldsymbol{\phi}^A_{i1}\log(\widetilde{\theta}_i)d_i)\end{aligned}$$

Introducing the abbreviation $a_{il} = \boldsymbol{\psi}_{il}\theta_i\boldsymbol{\phi}^A_{i0}$ and noticing that $1-\boldsymbol{\phi}^A_{i1} = \boldsymbol{\phi}^A_{i0}$, we have:

$$\begin{aligned}\mathcal{L}_A =& \frac{1}{2}\Bigg(\sum_{k\neq l} m_{kl}\log(\eta_{kl}) - \sum_i \boldsymbol{\psi}_{ik}\theta_i\boldsymbol{\phi}^A_{i0}\eta_{kl}D^G_l\underbrace{(1 - \frac{a_{il}}{D^G_l})}_{(1)} + \sum_k m_{kk}\log(\eta_{kk})\\ &-\sum_i \boldsymbol{\psi}_{ik}\theta_i\boldsymbol{\phi}^A_{i0}\eta_{kk}D^G_k\underbrace{(1 - \frac{a_{ik}}{D^G_k} + \frac{\theta_i}{D^G_k})}_{(2)} + m_{\text{bb}}\log(\eta_{\text{bb}}) + \sum_i \widetilde{\theta}_i\boldsymbol{\phi}^A_{i1}D^B\eta_{\text{bb}}\underbrace{(1 - \frac{\boldsymbol{\phi}^A_{i0}}{D^B})}_{(3)}\Bigg)\\ &+ m_{\text{bg}}\log(\eta_{\text{bg}}) - \sum_i \widetilde{\theta}_i\boldsymbol{\phi}^A_{i1}\eta_{\text{bg}}g\underbrace{(1 - \frac{\boldsymbol{\phi}^A_{i0}}{g})}_{(4)} + \sum_i \boldsymbol{\phi}^A_{i0}\log(\theta_i)d^G_i + \boldsymbol{\phi}^A_{i1}\log(\widetilde{\theta}_i)d_i)\end{aligned}$$

For the limit case it is sufficient to consider the terms (1)-(4). Note that if we replace each of the terms in the underbraces (1) through (4) with the value of 1 we obtain the simplification. Thus, if we can show that (1)-(4) converge to 1 in the limit case, the approximation error becomes negligible. As we will see this holds for almost all the cases – and in the cases where it does not hold, the two terms $\sum_i \widetilde{\theta}_i\boldsymbol{\phi}^A_{i1}(1-\boldsymbol{\phi}^A_{i1})(\eta_{\text{bg}} - \frac{1}{2}\eta_{\text{bb}})$ and $\frac{1}{2}\sum_i\sum_{kl}\boldsymbol{\psi}_{ik}\boldsymbol{\psi}_{il}\theta_i^2\boldsymbol{\phi}^A_{i0}(\boldsymbol{\phi}^A_{i0}\eta_{kl} - \eta_{kk})$ vanish due to another reason.

We start with the simplest case (4): Since $g = \sum_{i=1}^N \boldsymbol{\phi}^A_{i0}$, the term (4) obviously approaches 1. More formally, we can distinguish two cases: First, if $\lim_{N\to\infty} g = \infty$, then clearly $(\frac{\boldsymbol{\phi}^A_{i0}}{g}) \to 0$ since $\boldsymbol{\phi}^A_{i0}$ is bounded by 1 and the denominator grows faster.

Second, if $\lim_{N\to\infty} g = c$ for some constant c, the series converges. Thus, it has to hold $\lim_{i\to\infty} \phi^A_{i0} = 0$. And therefore again $(\frac{\phi^A_{i0}}{g}) \to 0$.[1]

The exactly same argumentation holds for the term (1): We have $D^G_l = \sum_{i=1}^N a_{il}$. Either it holds $\lim_{N\to\infty} D^G_l = \infty$, in which case, in the fraction the denominator grows faster than the nominator, i.e. the fraction becomes 0. Or we have $\lim_{N\to\infty} D^G_l = c$ for some constant c. The series converges and, thus, $\lim_{i\to\infty} a_{il} = 0$. In this case the nominator approaches 0. In both cases the fraction approaches 0 and therefore (1) approaches 1.

Let us now consider the term (3): Recap that $D^B = \sum_{i=1}^N \widetilde{\theta}_i \phi^A_{i1}$. In the first case, if $\lim_{N\to\infty} D^B = \infty$ then (3) approaches 1. Note again, that this result combined with the above result for (4) means that we can safely drop the term $\sum_i \widetilde{\theta}_i \phi^A_{i1}(1-\phi^A_{i1})(\eta_{\text{bg}} - \frac{1}{2}\eta_{\text{bb}})$. In the second case, $\lim_{N\to\infty} D^B = c$ for some constant c. Then $\lim_{i\to\infty} \widetilde{\theta}_i \phi^A_{i1} = 0$ has to hold. Accordingly, almost all terms in $\sum_i \widetilde{\theta}_i \phi^A_{i1}(1-\phi^A_{i1})(\eta_{\text{bg}} - \frac{1}{2}\eta_{\text{bb}})$ evaluate to zero (note that the variables are all bounded by 1). We can again safely drop the term.

Finally consider the term (2): Recap that $D^G_k = \sum_{i=1}^N a_{ik} = \psi_{ik}\theta_i\phi^A_{i0}$. Again, for the case $\lim_{N\to\infty} D^G_k = \infty$ the overall term clearly approaches 1. Second, for the case $\lim_{N\to\infty} D^G_k = c$, the terms a_{ik} approach 0. The same argument as for (3)+(4) can be used: in combination with the result for (1), the term $\frac{1}{2}\sum_i \sum_{kl} \psi_{ik}\psi_{il}\theta_i^2\phi^A_{i0}(\phi^A_{i0}\eta_{kl} - \eta_{kk})$ can be dropped since almost all elements evaluate to zero.

Overall, in all cases (i.e. independent whether the individual series converge or diverge), the resulting error we make by the approximation approaches zero. Finally, note that is not possible that all series converge at the same time. At least one of the series has to diverge when the graph grows since either infinitely many good or infinitely many anomalous nodes have to be added (or both). This means that the overall term $\mathcal{L}_A$ in the simplified version also has to diverge. Thus, while this term grows, the error gets smaller.

[1]Note that in general the following holds: If $\lim_{N\to\infty} \sum_{i=1}^N x_i = c$ for some constant c (i.e. the series converges), then $\lim_{i\to\infty} x_i = 0$ has to hold.

F Certifiable Robustness to Graph Perturbations

F.1 Label Propagation and Feature Propagation Certificates

Label propagation is a classic method for semi-supervised node classification, and there have been many variants proposed over the years [36–38]. The general idea is to find a classification function $\boldsymbol{F}$ such that the training nodes are predicted correctly and the predicted labels change smoothly over the graph. We can express this formally via the following optimization problem [253]:

$$\min_{\boldsymbol{F}} \left\{ \sum_{i=1}^{N} \sum_{j=1}^{N} \boldsymbol{A}_{ij} \left\| d_i^{\sigma-1} \boldsymbol{F}_{i*} - d_j^{\sigma-1} \boldsymbol{F}_{j*} \right\|^2 + \mu \sum_{i=1}^{N} d_i^{2\sigma-1} \left\| \boldsymbol{F}_{i*} - \boldsymbol{H}_{i*} \right\|^2 \right\} \tag{F.1}$$

where, d_i is the node degree, μ is a regularization parameter trading off smoothness and predicting the labeled nodes correctly, σ is a hyperparameter, and $\boldsymbol{H}$ is a matrix where the rows are one-hot vectors for the training nodes and zero vectors otherwise (i.e. $\boldsymbol{H}_{vc} = 1$ if $\{v \in \mathcal{V}_L \ \wedge \ y_v = c\}$ and $\boldsymbol{H}_{vc} = 0$ otherwise). The resulting matrix $\boldsymbol{F} \in \mathbb{R}^{N \times K}$ is the learned classification function, i.e. the value $\boldsymbol{F}_{vc}$ gives us the (unnormalized) probability that node v belongs to a class c, and we can make predictions by taking the argmax. The problem can be solved in closed form (even though in practice one would use power iteration) and the solution is: $\boldsymbol{F} = (1-\alpha)\left(\boldsymbol{I}_N - \alpha \boldsymbol{D}^{-\sigma} \boldsymbol{A} \boldsymbol{D}^{\sigma-1}\right)^{-1} \boldsymbol{H}$ for $\alpha = 2/(2+\mu)$. We can see that setting $\sigma = 1$, i.e. the standard Laplacian variant [38] we obtain:

$$\boldsymbol{F} = (1-\alpha)\left(\boldsymbol{I}_N - \alpha \boldsymbol{D}^{-1} \boldsymbol{A}\right)^{-1} \boldsymbol{H} = \boldsymbol{\Pi} \boldsymbol{H} \tag{F.2}$$

From Eq. F.2 we have that Label Propagation is very similar to our $\boldsymbol{\pi}$-PPNP: instead of diffusing logits which come from a neural network it propagates the one-hot vectors of the labeled nodes instead. From here onwards we apply our proposed method without any modifications by simply providing a different $\boldsymbol{H}$ matrix in Problem 7.1.

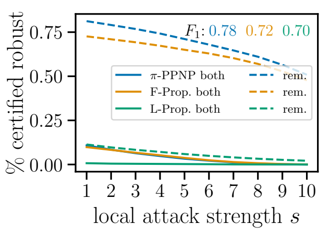
We can also certify the feature propagation (FP) approach of which there are several variants: e.g. the normalized Laplacian FP [180], or a recently proposed equivalent model termed simple graph convolution (SGC) [151]. Feature propagation is carried out in two steps: (i) the node features are diffused to incorporate the graph structure $\boldsymbol{X}^{\text{diff}} = \boldsymbol{\Pi}\boldsymbol{X}$, and (ii) a simple logistic regression model is trained using the diffused features $\boldsymbol{X}^{\text{diff}}$ and subset of labelled nodes. Now, let the $\boldsymbol{W} \in \mathbb{R}^{D \times K}$ be the weights corresponding to a trained logistic regression model. The predictions for all nodes are calculated as $\boldsymbol{Y} = \text{softmax} = (\boldsymbol{X}^{\text{diff}}\boldsymbol{W}) = \text{softmax}(\boldsymbol{\Pi}\boldsymbol{X}\boldsymbol{W}) = \text{softmax}(\boldsymbol{\Pi}\boldsymbol{H})$ with $\boldsymbol{H} = \boldsymbol{X}\boldsymbol{W}$. By simply providing a different matrix $\boldsymbol{H}$ in Problem 7.1 we can certify feature propagation.

F.2 Further Experiments

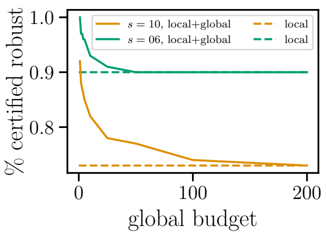
In Fig. F.1a we show the percent of certifiable robust nodes for different local budgets on the Pubmed graph ($N = 19,717, |\mathcal{E}| = 44,324, D = 500, K = 3$) [29] demonstrating that our method scales to large graphs. Similar to before (Fig. 7.3a), the models are more robust to attackers that can only remove edges. In Fig. F.1b we analyze the robustness of Citeseer w.r.t. increasing global budget. The global budget constraints are again able to successfully restrict the attacker. The global budget makes a larger difference when the attacker has a larger local attack strength ($s = 10$). In Fig. F.1c we show that the robust training increases the percent of certifiably robust nodes. Comparing to Fig. 7.4c we conclude that training with a larger local attack strength ($s = 10$ as opposed to $s = 6$) makes the model more robust overall while the predictive performance (F_1 score) is the same in both cases.

We also investigate certifiable accuracy. The ratio of nodes that are both certifiably robust and at the same time have a correct prediction is a lower bound on the overall worst-case classification accuracy since the worst-case perturbation can be different for each node. We plot this ratio in Fig. F.2a for Citeseer and see that the certifiable accuracy is relatively close to the clean accuracy when the budget is restrictive, and it decreases gracefully as we in increase the budget.

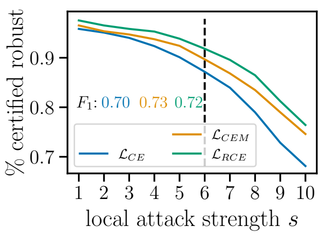
To show how the runtime scales with number of nodes and number of edges we randomly generate SBM graphs of increasing size, and we set all edges in the generated graphs as fragile ($\mathcal{F} = \mathcal{E}$). In Fig. F.2b we see the mean runtime across five runs for local budget (VI algorithm). Even for graphs with more than 10K nodes the certificate runs in a few seconds. Similarly, Fig. F.2c shows the runtime for global budget (RLT relaxation). We see that the runtime scales linearly with the number of edges. Furthermore, the overall runtime can be easily reduced by: (i) stopping early whenever the worst-case margin becomes negative, (ii) using Gurobi's distributed optimization capabilities to reduce solve times, and (iii) having a single shared preprocessing step for all nodes.



(a) Pubmed, local budget

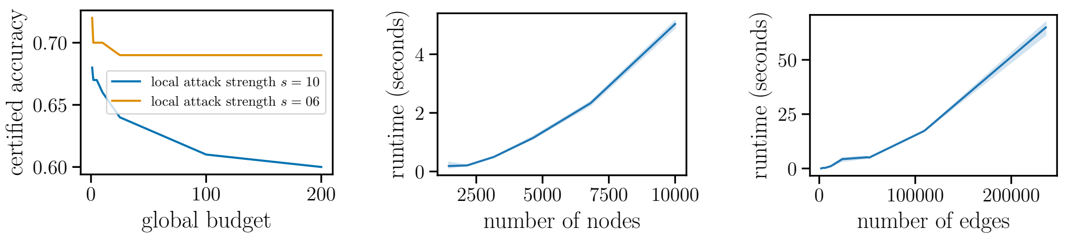


(b) Citeseer, global budget



(c) Citeseer, robust training

Figure F.1: (a,b) The local and global budget successfully restrict the attacker. Models are more robust to removing edges than both removing and adding edges. (c) Our robust training successfully increases the percentage of certifiably robust nodes.



(a) Citeseer, certifiable accuracy. **(b)** Runtime: SBM, local. **(c)** Runtime: SBM, global.

Figure F.2: Further experiments on certifiable accuracy (a) and runtime (b-c).

F.3 Proofs

Proof. Proposition 7.1. Problem 7.2 can be formulated as an average cost infinite horizon Markov decision problem, where at each node v we decide which subset of $\mathcal{F}_v$ edges are active, i.e. $\mathcal{A}_v = \mathcal{P}(\mathcal{F}^v)$ where $\mathcal{P}(\mathcal{F}^v)$ is the power set of $\mathcal{F}^v$ and the reward depends only on the starting state but not on the action and the ending state $r(v, a) = \boldsymbol{r}_v, \forall a \in \mathcal{A}_v$. From the average cost infinite horizon optimality criterion as shown by Fercoq et al. [201] we have:

$$\lim_{T\to\infty} \frac{1}{T}\mathbb{E}\Big(\sum_{t=0}^{T-1} r\big(X_t, \nu_t\big)\Big) = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\Big(\sum_{t=0}^{T-1} r_{X_t,j}\overline{\nu}_j\big(X_t\big)\Big) = \sum_{i,j\in[n]} \boldsymbol{\pi}_i \boldsymbol{P}_{ij} r_{i,j} \tag{F.3}$$

where $X_t \in \mathcal{S}$ is a random variable denoting the state of the system at the discrete time $t \geq 0$, and $\nu(h_t)$ is deterministic control strategy determining a sequence of actions and is a function of the history $h_t = (X_0, \nu_0, \dots, X_{t-1}, \nu_{t-1}, X_t)$. For this problem there exists a stationary (feedback) strategy $\overline{\nu}(X_t)$ that does not depend on the history such that for all $t \geq 0, \nu_t(h_t) = \overline{\nu}(X_t)$. Eq. F.3 follows from the ergodic theorem for Markov chains. Here the reward is more general and can be set depending on the edge (i, j). Letting $r_{ij} = \boldsymbol{r}_i, \forall j$ and plugging it in Eq. F.3 we get that the optimality criterion equals $\boldsymbol{r}^T\boldsymbol{\pi}$ since the transion matrix $\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A}$ is row-stochastic. As shown by Hollanders et al. [202] policy iteration is well suited to optimize PageRank and our Algorithm 7.1 corresponds to policy iteration with local budget for the above MDP. For a fixed damping factor α (which is our case) policy iteration always converges in less iterations than value iteration [205] and does so in weakly polynomial time that depends on the number of fragile edges [202]. □

Proof. Proposition 7.2. Eq. 7.6b and Eq. 7.6c correspond to the LP of the unconstrained MDP on the auxiliary graph. Intuitively, the variable x_v maps to the PageRank score of node v, and from the variables x^0_{ij}/x^1_{ij} we can recover the optimal policy: if the variable x^0_{ij} (respectively x^1_{ij}) is non-zero then in the optimal policy the fragile edge (i, j) is turned off (respectively on). Since there exists a deterministic optimal policy, only one of them is non-zero but never both. Eq. 7.6d corresponds to the local budget. Remarkably, despite the variables x^0_{ij}/x^1_{ij} not being integral, since they share the factor $\frac{x_i}{d_i}$ from Eq. 7.6c we can exactly count the number of edges that are turned off or on using only linear constraints. Eq. 7.6e and Eq. 7.6f enforce the global budget. From Eq. 7.6e we have that whenever x^0_{ij} is non-zero it follows that $\beta^1_{ij} = 0$ and $\beta^0_{ij} = 1$ since that is the only configuration that satisfies the constraints (similarly for x^1_{ij}). Intuitively, this effectively makes the $\beta^0_{ij}/\beta^1_{ij}$ variables "counters" and thus, we can utilize them in Eq. 7.6f to enforce the total number of perturbed edges to not exceed B.

We also have to show that solving the MDP on the auxiliary graph solves the same problem as the MDP on the original graph. Recall that whenever we traverse any edge from node i we obtain reward $\boldsymbol{r}_i$. On the other hand, whenever we traverse an edge from the auxiliary node v_{ij} corresponding to a fragile edge (i,j) to the node i (action "off") we get negative reward $-\boldsymbol{r}_i$, and the transition probability is 1. Intuitively, traversing back and forth between node i and node v_{ij} does not change the overall reward obtained (since $\boldsymbol{r}_i$ and $-\boldsymbol{r}_i$ cancel out). That is, we have the same reward as in the original graph with the edge (i,j) excluded. Similarly, when we traverse the edge from auxiliary node v_{ij} to the node j (action "on") we obtain 0 reward, i.e. no additional reward is gained and the transition happens with probability α. Therefore, the overall reward is the same as if the fragile edge (i,j) would be present in the original graph.

Formally, for any given arbitrary policy for the unconstrained MDP on the auxiliary graph, let $k_v = |\{x^{*0}_{vj} \mid x^{*0}_{vj} > 0\}|$ be the current number of "off" fragile edges for node v and let $\mathcal{F}^v_+$ be the current set of "on" fragile edges. From Eq. 7.6b and Eq. 7.6c we have:

$$x_v - \alpha \sum_{(i,v)\in\mathcal{E}_f\cup\mathcal{F}^v_+} x_i d_i^{-1} - k_v x_v d_v^{-1} = (1-\alpha)\boldsymbol{z}_v \tag{F.4a}$$

$$x_v - k_v x_v d_v^{-1} = \alpha \sum_{(i,v)\in\mathcal{E}_f\cup\mathcal{F}^v_+} x_i d_i^{-1} + (1-\alpha)\boldsymbol{z}_v \tag{F.4b}$$

$$x_v(1 - k_v d_v^{-1}) = \boldsymbol{\pi}(\boldsymbol{z})_v \tag{F.4c}$$

where we can see that $\boldsymbol{\pi}(\boldsymbol{z})_v$ is the personalized PageRank for node v for a perturbed original graph corresponding to the current policy, i.e. the graph where all $(v,j) \in \mathcal{F}^v_+$ for all $v \in \mathcal{V}$ are turned "on". Plugging in Eq. F.4c into the objective from Eq. 7.6a and using Eq. 7.6c we have

$$\max \sum\nolimits_{v\in\mathcal{V}} x_v \boldsymbol{r}_v - \sum\nolimits_{(i,j)\in\mathcal{F}} x^0_{ij}\boldsymbol{r}_i = \max \sum\nolimits_{v\in\mathcal{V}} \boldsymbol{\pi}(\boldsymbol{z})_v \boldsymbol{r}_v$$

which exactly corresponds to the objective of Problem 7.2. Since the above analysis holds for any policy it also holds for the optimal policy, and therefore solving the unconstrained MDP on the auxiliary graph is equivalent to solving the unconstrained MDP on the original graph. Combining everything together we have that solving the QCLP is equivalent to solving Problem 7.2.

□

Proof. Proposition 7.3. Using the reformulation-linearization technique (RLT) we relax the quadratic constraints in Eq. 7.6e. In general, from RLT it follows that we add the following four linear constraints for each pairwise quadratic constraint $m_i m_j = M_{ij}$

$$M_{ij} - \underline{m}_i m_j - \underline{m}_j m_i \geq -\underline{m}_i \underline{m}_j \tag{F.5a}$$

$$M_{ij} - \underline{m}_j m_i - \overline{m}_i m_j \leq -\underline{m}_j \overline{m}_i \tag{F.5b}$$

$$M_{ij} - \underline{m}_i m_j - \overline{m}_j m_i \leq -\underline{m}_i \overline{m}_j \tag{F.5c}$$

$$M_{ij} - \overline{m}_i m_j - \overline{m}_j m_i \geq -\overline{m}_i \overline{m}_j \tag{F.5d}$$

where $\underline{m}_i \leq m_i \leq \overline{m}_i$ are lower and upper bounds for m_i.

From Eq. 7.6e we see that our quadratic terms always equal to 0 ($M_{ij} = 0$), and we have the following upper $\overline{\beta^0_{ij}} = \overline{\beta^1_{ij}} = 1$, and $\overline{x^1_{ij}} = \overline{x^0_{ij}} = \frac{\overline{x_i}}{d_i} > 0$, and lower bounds $\underline{\beta^0_{ij}} = \underline{\beta^1_{ij}} = \underline{x^1_{ij}} = \underline{x^0_{ij}} = 0$. Plugging these upper/lower bounds into Eq. F.5 for our quadratic terms $x^0_{ij}\beta^1_{ij} = 0$ and $x^1_{ij}\beta^0_{ij} = 0$ we see that the constraints arising from Eq. F.5a, Eq. F.5b and Eq. F.5c are always trivially fulfilled. Thus, we are left with the constraints arising from Eq. F.5d which for our problem are:

$$x^0_{ij} + \overline{x^0_{ij}}\beta^1_{ij} \leq \overline{x^0_{ij}} \qquad \text{and} \qquad x^1_{ij} + \overline{x^1_{ij}}\beta^0_{ij} \leq \overline{x^1_{ij}} \tag{F.6}$$

There are two cases to consider:

- Case 1:The edge is turned “off”. We have $x^0_{ij} = x_i d_i^{-1}$ and $x^1_{ij} = 0$.

  $$x^0_{ij} + \overline{x^0_{ij}}\beta^1_{ij} \leq \overline{x^0_{ij}} \implies x^0_{ij}(\overline{x^0_{ij}})^{-1} + \beta^1_{ij} \leq 1 \implies$$
  $$\implies x^0_{ij}(\overline{x^0_{ij}})^{-1} \leq \beta^0_{ij} \implies x^0_{ij}(\overline{x_i} d_i^{app_{p}roofs-1})^{-1} \leq \beta^0_{ij}$$

  And trivially: $x^1_{ij} + \overline{x^1_{ij}}\beta^0_{ij} \leq \overline{x^1_{ij}} \implies \overline{x^1_{ij}}\beta^0_{ij} \leq \overline{x^1_{ij}} \implies \beta^0_{ij} \leq 1$.

- Case 2: The edge is turned “on”. We have $x^1_{ij} = x_i d_i^{-1}$ and $x^0_{ij} = 0$.

  $$x^1_{ij} + \overline{x^1_{ij}}\beta^0_{ij} \leq \overline{x^1_{ij}} \implies x^1_{ij}(\overline{x^1_{ij}})^{-1} + \beta^0_{ij} \leq 1 \implies x^1_{ij}(\overline{x_i} d_i^{-1})^{-1} \leq \beta^1_{ij}$$

  And trivially: $\overline{x^0_{ij}} + \overline{x^0_{ij}}\beta^1_{ij} \leq \overline{x^0_{ij}} \implies \overline{x^0_{ij}}\beta^1_{ij} \leq \overline{x^0_{ij}} \implies \beta^1_{ij} \leq 1$

The above two cases are disjoint and we can plug β^0_{ij} and β^1_{ij} into Eq. 7.6f to obtain Eq. 7.7. □

F.4 SDP Relaxation

In this section we show that the SDP-relaxation [78] based on semidefinite programming is not suitable for our problem since the constraints are trivially fulfilled. For convenience, we rename the variables that participate in the quadratic constraints ($\beta^0_{ij}, x^0_{ij}, \dots$) to ($y_1, y_2, \dots$). The SDP relaxation replaces the product terms $y_i y_j$ (e.g. $x^0_{ij}\beta^1_{ij}$) by an element $\boldsymbol{Y}_{ij}$ of an $n \times n$ matrix $\boldsymbol{Y}$ and adds the constraint $\boldsymbol{Y} - \boldsymbol{y}\boldsymbol{y}^T \succeq 0$, where $\boldsymbol{y}$ is the vector of variables. Since in the QCLP there are no terms of the form $y_i y_i$ corresponding to the elements on the diagonal, we can make the diagonal elements $\boldsymbol{Y}_{ii}$ arbitrarily high to make the matrix $\boldsymbol{Y} - \boldsymbol{y}\boldsymbol{y}^T$ positive semidefinite and trivially satisfy the constraint.

F.5 Hardness of PageRank Optimization with Global Budget

The Link Building problem [203, 254] aims at maximizing the PageRank of a single given node v by selecting a set of k optimal edges that point to node v. We use the fact that the Link Building problem is a special case of Problem 7.2 to derive our hardness result.

Problem F.1 (Link Building [203])**.** *Given a graph $G = (\mathcal{V}, \mathcal{E})$, node $v \in \mathcal{V}$, budget $k \in \mathbb{Z}$, and any fixed $\alpha \in (0, 1)$. Find a set $\mathcal{S} \subseteq \mathcal{V} \setminus \{v\}$ with $|S| = k$ maximizing $\boldsymbol{\pi}_{\tilde{G},\alpha}(\boldsymbol{e}/n)_v$ in the perturbed graph $\tilde{G} = (\mathcal{V}, \tilde{\mathcal{E}} := \mathcal{E}_f \cup (\mathcal{S} \times \{v\}))$, where $\boldsymbol{e}/n$ is the teleport vector for the uniform distribution.*

Proposition F.1. *Problem 7.2 with global budget is W[1]-hard and allows no FPTAS.*

Proof. Setting the teleport vector to the uniform distribution $\boldsymbol{z} = \boldsymbol{e}/n$, the reward vector to $\boldsymbol{r} = \boldsymbol{e}_v$, the set of fragile edges to $\mathcal{F} = (\mathcal{V} \setminus \{v\}) \times \{v\}$, the set of fixed edges to $\mathcal{E}_f = \mathcal{E}$, and configuring the budgets as $b_v = 1, \forall v$ and $B = k$ we see that the Problem F.1 is a special case of Problem 7.2. Note that, since we can always increase $\boldsymbol{\pi}_v$ by adding edges pointing to v, the $x \leq B$ global constraint is equivalent to the $x = B$ constraint where x is the expression on the left-hand side in Eq. 7.6f.

Olsen [203] shows that the Link Building problem is W[1]-hard and admits no FPTAS by reducing it to the Regular Independent Set problem which is W[1]-complete [255]. Therefore, Problem 7.2 with global budget is also W[1]-hard and allows no FPTAS since k is preserved in the reduction. □

F.6 Alternative Upper Bound

As an alternative upper bound for x_v we can use the following approach: Assume we have given a fixed set of edges $\mathcal{E}_f$ where every node has at least one fixed edge. From Proposition 7.2 we have $x_v = (1 - k_v d_v^{-1})^{-1} \boldsymbol{\pi}_v$. To maximize this value, we can simply set $\boldsymbol{\pi}(\boldsymbol{z})_v = 1$ (since this is the maximal PageRank score achievable) and $k_v = |\mathcal{F}^v|$. Since every node has at least one fixed edge, we have $d_v > k_v$, i.e. the inverse is always defined.

F.7 Experimental Details

We preprocess each graph and keep only the nodes that belong to largest connected component. The resulting graph for Cora-ML has $N = 2,810, |\mathcal{E}| = 10,138$, for Citeseer $N = 2,110, |\mathcal{E}| = 7,336$ and for Pubmed $N = 19,717, |\mathcal{E}| = 88,648$. Unless otherwise specified we set $\alpha = 0.85$. We compute the certificates with respect to the predicted class label, i.e. we set y_t in $m^*_{y_t,*}(t)$ to the predicted class for node t using the clean graph. Experiments are run on Nvidia 1080Ti GPUs using CUDA and TensorFlow and on Intel CPUs. We use the GUROBI solver to solve the linear programs.

We configure our $\boldsymbol{\pi}$-PPNP model with one hidden layer and choose a latent dimensionality of 64. We randomly select 20 nodes per class for the training/validation set, and use the rest for the testing. The weights θ are regularized with the L_2 norm with strength of

$5e-2$. We train for a maximum of $10,000$ epoch with a fixed learning rate of $1e-2$ and patience of 100 epochs for early stopping. We train the model for five different random splits and report the averaged results.

When reporting results for local budget (e.g. Fig. 7.3, Fig. 7.4c, Fig. F.1a, Fig. F.1c) we evaluate the certifiable robustness for all test nodes, since as we discussed in Sec. 7.4.3 we only need to run Algorithm 7.1 $K \times K$ times to obtain certificates for all nodes. When reporting results for global budget (e.g. Fig. 7.4a and Fig. F.1b) we randomly select 150 test nodes for which we compute the certificate. For all results regarding runtime (e.g. Fig. 7.4b, Fig. F.2b, Fig. F.2c) we report average time across five runs on a machine with 20 CPU cores.

G Efficient Robustness Certificates for Discrete Data

G.1 Proofs

Proof. Proof (Proposition 8.1). First we show that the regions are disjoint. Let $\boldsymbol{z} \in \mathcal{R}_i^{r_a,r_d}$, and $\boldsymbol{z} \in \mathcal{R}_j^{r_a,r_d}$ for some $i \neq j$. From the definition of a region it follows that $\|\boldsymbol{x}_\mathcal{C} - \boldsymbol{z}_\mathcal{C}\|_0 = i$ and $\|\boldsymbol{x}_\mathcal{C} - \boldsymbol{z}_\mathcal{C}\|_0 = j$. This can be true only if $i = j$ which is a contradiction. Therefore, $\boldsymbol{z}$ cannot belong to two different regions. For any $\boldsymbol{z}$ and $\boldsymbol{x}$, $\|\boldsymbol{x}_\mathcal{C} - \boldsymbol{z}_\mathcal{C}\|_0 \in \{0, \dots, r_a + r_d\}$ since the $\|\cdot\|_0$ (Hamming) distance between two $|\mathcal{C}|$-dimensional vectors has the range $\{0, \dots, |\mathcal{C}|\}$. Thus, any $\boldsymbol{z}$ must land in some region $\mathcal{R}_q^{r_a,r_d}$ with $q \leq |\mathcal{C}|$, and for any $q > |\mathcal{C}| = r_a + r_d$ we have $\mathcal{R}_q^{r_a,r_d} = \emptyset$. Therefore, $\mathcal{X} = \bigcup_{q=0}^{q=\infty} \mathcal{R}_q^{r_a,r_d} = \bigcup_{q=0}^{q=r_a+r_d} \mathcal{R}_q^{r_a,r_d}$. $\square$

Proof. Proof (Proposition 8.2). For any $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{S}_{r_a,r_d}(\boldsymbol{x})$, and $\mathcal{R}_q^{r_a,r_d}$:

$$\begin{aligned}&\Pr\big(\phi(\boldsymbol{x}) \in \mathcal{R}_q^{r_a,r_d}\big) = \Pr\big(\|\boldsymbol{x}_\mathcal{C} - \phi(\boldsymbol{x})_\mathcal{C}\|_0 = q\big) = \\ &\Pr\Big(\sum_{i\in\mathcal{C}} \mathbb{I}[\boldsymbol{x}_i \neq \phi(\boldsymbol{x})_i] = q\Big) = \Pr\Big(\sum_{i\in\mathcal{C}} \epsilon_i = q\Big)\end{aligned} \tag{G.1}$$

where $\epsilon_i \sim \text{Ber}(p = p_-^{\boldsymbol{x}_i} p_+^{(1-\boldsymbol{x}_i)})$. The first equality in Eq. G.1 follows from the definition of a region, and the last equality follows from the definition of $\phi(\cdot)$. Since $\boldsymbol{x} \in \mathcal{R}_q^{r_a,r_d}$ we have $\sum_{i\in\mathcal{C}} \boldsymbol{x}_i = r_d$ and $\sum_{i\in\mathcal{C}} 1 - \boldsymbol{x}_i = r_a$. Therefore, $\sum_{i\in\mathcal{C}} \epsilon_i \sim Q$ where $Q = \text{PB}([p_+, r_a][p_-, r_d])$. $\square$

Proof. Proof (Proposition 8.3). For any $\boldsymbol{z} \in \mathcal{R}_q^{r_a,r_d}$, by definition it holds $\|\boldsymbol{x}_\mathcal{C} - \boldsymbol{z}_\mathcal{C}\|_0 = q$. Let $q_- = \sum_{i=1}^d \mathbb{I}(\boldsymbol{x}_i - 1 = \boldsymbol{z}_i)$ and $q_+ = \sum_{i=1}^d \mathbb{I}(\boldsymbol{x}_i + 1 = \boldsymbol{z}_i)$, so $q = q_+ + q_-$. We have:

$$\begin{aligned}\eta_q^{r_a,r_d} &= \frac{\Pr(\phi(\boldsymbol{x}) = \boldsymbol{z})}{\Pr(\phi(\tilde{\boldsymbol{x}}) = \boldsymbol{z})} = \frac{\prod_{i\in\tilde{\mathcal{C}}} \Pr(\phi(\boldsymbol{x})_i = \boldsymbol{z}_i) \prod_{j\in\mathcal{C}} \Pr(\phi(\boldsymbol{x})_j = \boldsymbol{z}_j)}{\prod_{i\in\tilde{\mathcal{C}}} \Pr(\phi(\tilde{\boldsymbol{x}})_i = \boldsymbol{z}_i) \prod_{j\in\mathcal{C}} \Pr(\phi(\tilde{\boldsymbol{x}})_j = \boldsymbol{z}_j)} \\ &= \frac{\prod_{j\in\mathcal{C}} \Pr(\phi(\boldsymbol{x})_j = \boldsymbol{z}_j)}{\prod_{j\in\mathcal{C}} \Pr(\phi(\tilde{\boldsymbol{x}})_j = \boldsymbol{z}_j)} = \frac{p_-^{q_-}(1-p_-)^{r_d - q_-} p_+^{q_+}(1-p_+)^{r_a - q_+}}{p_-^{r_a - q_+}(1-p_-)^{q_+} p_+^{r_d - q_-}(1-p_+)^{q_-}} \\ &= p_-^{q - r_a}(1-p_-)^{r_d - q} p_+^{q - r_d}(1-p_+)^{r_a - q} \\ &= \left[\frac{p_+}{1-p_-}\right]^{q - r_d} \left[\frac{p_-}{1-p_+}\right]^{q - r_a}\end{aligned}$$

Where the second equality holds since ϕ is independent per dimension, and the third equality holds since $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ agree on $\tilde{\mathcal{C}}$. Plugging in the definition of ϕ and rearranging we obtain $\eta_q^{r_a,r_d}$. Thus, the ratio is constant for any $\boldsymbol{z} \in \mathcal{R}_q^{r_a,r_d}$. Now we show that the ratio is a monotonic function of q:

$$\eta_q^{r_a,r_d} = \left[\frac{p_+}{1-p_-}\right]^{q-r_d}\left[\frac{p_-}{1-p_+}\right]^{q-r_a} = C \cdot \left[\frac{p_+p_-}{p_+p_- + \underbrace{1-(p_+-p_-)}_{:=u}}\right]^q \tag{G.2}$$

$C = \left[\frac{p_+p_-}{(1-p_+)(1-p_-)}\right]^{-(r_a+r_d)} \geq 0$ is a non-negative constant that does not depend on q since $p_+, p_- \in [0,1]$, and hence does not change the monotonicity. We have three cases: (i) if $p_+ + p_- < 1$ then $u > 0$ in the denominator of Eq. G.2, the ratio is < 1 and a decreasing function of q; (ii) if $p_+ + p_- = 1$ then $u = 0$ and the ratio becomes constant $C \cdot 1^q$; (iii) if $p_+ + p_- > 1$ then $u < 0$, the ratio is > 1 and an increasing function of q. □

G.2 Multi-class Certificates

For the multi-class certificate our goal is to solve the following optimization problem:

$$\begin{aligned} &\mu_{\boldsymbol{x},\tilde{\boldsymbol{x}}}(p_1(\boldsymbol{x}), \ldots, p_{\mathcal{Y}}(\boldsymbol{x}), y^*) \\ &= \min_{h \in \mathcal{H}} \Pr(h(\phi(\tilde{\boldsymbol{x}})) = y^*) - \max_{y \neq y^*} \Pr(h(\phi(\tilde{\boldsymbol{x}})) = y) \\ &\text{s.t.}\quad \Pr(h(\phi(\boldsymbol{x})) = y^*) = p_{y^*} \\ &\text{and}\ \ \Pr(h(\phi(\boldsymbol{x})) = y) = p_y, \quad y \neq y^* \end{aligned} \tag{G.3}$$

where y^* is the (predicted or ground-truth) class we want to certify. Similar to before computing $p_y(\boldsymbol{x})$ exactly is difficult, thus we compute a lower bound $\underline{p_{y^*}(\boldsymbol{x})}$ for y^* and an upper bound $\overline{p_y(\boldsymbol{x})}$ for all other y. Since we are conservative in the estimates, the solution to Eq. G.3 using these bounds yields a valid certificate. Estimating the lower and upper bounds from Monte Carlo samples such that they hold simultaneously with confidence level α requires some care. Specifically, we have to correct for multiple testing error. Similar to Jia et al. [256] we estimate each bound individually using a Clopper-Pearson Bernoulli confidence interval with confidence $\frac{\alpha}{C}$ where $C = |\mathcal{Y}|$ is the number of classes and use Bonferroni correction to guarantee with confidence of α that the estimates hold simultaneously.

The problem in Eq. G.3 is valid if $\underline{p_{y^*}(\boldsymbol{x})} + \overline{p_{\tilde{y}}(\boldsymbol{x})} < 1$. The binary-class certificate assumes that $\overline{p_{\tilde{y}}(\boldsymbol{x})} = 1 - \underline{p_{y^*}(\boldsymbol{x})}$. From here we can directly conclude that the multi-class certificate is in principle always equal or better than the binary certificate, and in particular the improvement can only occur when $\underline{p_{y^*}(\boldsymbol{x})} + \overline{p_{\tilde{y}}(\boldsymbol{x})} < 1$. Note that, however, the value of $\underline{p_{y^*}(\boldsymbol{x})}$ will be lower for the multi-class certificate compared to the binary-class certificate due to the Bonferroni correction. This implies that in some cases the binary-class certificate can yield a higher certified radius. For the majority of our experiments the multi-class certificate was better.

Now, given an input $\boldsymbol{x}$ and a perturbation set $\mathcal{B}_{r_a,r_d}(\boldsymbol{x})$ if it holds that:

$$\min_{\tilde{\boldsymbol{x}} \in \mathcal{B}(\boldsymbol{x})} \mu_{\boldsymbol{x},\tilde{\boldsymbol{x}}}(p_1(\boldsymbol{x}), \ldots, p_{\mathcal{Y}}(\boldsymbol{x}), y^*) > 0$$

we can guarantee that classification margin for the worst-case classifier is always bigger than 0 for all $\tilde{\boldsymbol{x}} \in \mathcal{B}(\boldsymbol{x})$. This implies that $g(\boldsymbol{x}) = g(\tilde{\boldsymbol{x}}) = y^*$ for any input within the ball, i.e. $\boldsymbol{x}$ is certifiably robust. Compare this to the previous certificate where we had to verify whether $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}(p^*, y^*) > 0.5$ which was not tight for $|\mathcal{Y}| > 2$.

Similar to before, Eq. G.3 is equivalent to the following LP:

$$\begin{aligned} &\min_{\boldsymbol{h},\boldsymbol{t}} \boldsymbol{h}^T\tilde{\boldsymbol{r}} - \boldsymbol{t}^T\tilde{\boldsymbol{r}} \\ \text{s.t.}\quad &\boldsymbol{h}^T\boldsymbol{r} = \underline{p_{y^*}(\boldsymbol{x})}, \qquad \boldsymbol{t}^T\boldsymbol{r} = \overline{p_{\tilde{y}}(\boldsymbol{x})}, \\ &0 \le \boldsymbol{h} \le 1, \qquad 0 \le \boldsymbol{t} \le 1 \end{aligned} \tag{G.4}$$

where $\tilde{y} = \max_{y \neq y^*} \overline{p_y(\boldsymbol{x})}$ is the class with the second highest number of majority votes after y^*. The proof is analogous to the proof of Lemma 2 in Lee et al. [220].

The exact solution to the LP is easily obtained with another greedy algorithm: first sort the regions such that $c_1 \ge c_2 \ge \dots \ge c_I$, then iteratively assign $\boldsymbol{h}_i = 1$ in decreasing order for all regions $\mathcal{R}_i$ until the constraint $\underline{p_{y^*}(\boldsymbol{x})}$ is met. Finally, iteratively assign $\boldsymbol{t}_j = 1$ now in increasing order for all regions $\mathcal{R}_j$ until the constraint $\overline{p_{\tilde{y}}(\boldsymbol{x})}$ is met.

G.3 Special Cases for Flipping Probabilities

We derive the regions of constant likelihood ratio for the case $p_+ = 0$ and $p_- > 0$. There are only three regions which we have to consider. First note that there is only one set of vectors $\boldsymbol{z}$ which can be reached by both $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ when applying the randomization ϕ and these are the vectors which have all valid (reachable via deletion) configurations of ones and zeros in $\tilde{\mathcal{C}}$ and all zeros in $\mathcal{C}$. This holds since $\boldsymbol{x}_\mathcal{C}$ and $\tilde{\boldsymbol{x}}_\mathcal{C}$ are complementary and we can only delete edges. See Fig. 8.2 for an illustration. Denoting this region with $\mathcal{R}_1$ we have that $\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_1) = p_-^{r_d}$ and $\Pr(\phi(\tilde{\boldsymbol{x}}) \in \mathcal{R}_1) = p_-^{r_a}$ since we need to successfully delete all edges.

The second region $\mathcal{R}_2$ corresponds to the case where we flip less than r_d bits in $\boldsymbol{x}$ and this happens with probability $\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_2) = 1 - p_-^{r_d}$. By definition the vectors in the intersection reachable by both $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ are all in $\mathcal{R}_1$, thus $\Pr(\phi(\tilde{\boldsymbol{x}}) \in \mathcal{R}_2) = 0$. Finally, the third region $\mathcal{R}_3$ corresponds to the case where we flip less than r_a bits in $\tilde{\boldsymbol{x}}$, we have $\Pr(\phi(\tilde{\boldsymbol{x}}) \in \mathcal{R}_3) = 1 - p_-^{r_a}$ and $\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_3) = 0$. For the binary-class certificate we can ignore any regions $\mathcal{R}_i$ where $\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_i) = 0$, so the only two valid regions are $\mathcal{R}_1$ and $\mathcal{R}_2$. However, for our multi-class certificate all three regions are necessary. The case for $p_+ > 0, p_- = 0$ is analogous. We have: $\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}'_1) = p_+^{r_a}$ and $\Pr(\phi(\tilde{\boldsymbol{x}}) \in \mathcal{R}'_1) = p_+^{r_d}$ for the first region; $\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}'_2) = 1 - p_+^{r_a}$ and $\Pr(\phi(\tilde{\boldsymbol{x}}) \in \mathcal{R}'_2) = 0$ for the second region; $\Pr(\phi(\tilde{\boldsymbol{x}}) \in \mathcal{R}'_3) = 1 - p_+^{r_d}$ and $\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}'_3) = 0$ for the third region.

G.4 Region Traversal

As we discussed in Sec. 8.4.3 we can efficiently compute $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}$ by directly visiting the regions $\mathcal{R}_q^{r_a,r_d}$ in decreasing order w.r.t. the ratio $\eta_q^{r_a,r_d}$ without sorting. The pseudo-code

Algorithm G.1 Compute $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}$ # special cases omitted

```
Input: $p_+, p_-, r_a, r_d, \underline{p_{y^*}(\boldsymbol{x})}$
if $p_+ + p_- < 1$ then
    start $= r_a + r_d$,    end $= 0$
else
    start $= 0$,    end $= r_a + r_d$
end if
Initialize $p = 0$,    $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}} = 0$.
for $q =$ start to end do
    Compute $\eta_q^{r_a,r_d}$ ratio using Proposition 8.3
    Compute $\mathrm{PB}(q;\cdot) = \Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_q^{r_a,r_d})$ as in Sec. 8.4.4
    $\Pr(\phi(\tilde{\boldsymbol{x}}) \in \mathcal{R}_q^{r_a,r_d}) = \mathrm{PB}(q;\cdot)/\eta_q^{r_a,r_d}$
    if  $p = \Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_q^{r_a,r_d}) > \underline{p_{y^*}(\boldsymbol{x})}$ then
        break
    else
        $p = p + \Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_q^{r_a,r_d})$
        $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}} = \rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}} + \Pr(\phi(\tilde{\boldsymbol{x}}) \in \mathcal{R}_q^{r_a,r_d})$
    end if
end for
if $\underline{p_{y^*}(\boldsymbol{x})} - p > 0$ then
    $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}} = \rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}} + (\underline{p_{y^*}(\boldsymbol{x})} - p)/\eta_q^{r_a,r_d}$
end if
Output: $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}$
```

is given in Algorithm G.1 and corresponds to the greedy algorithm for solving the LP in Eq. 8.4 and thus Eq. 8.3. Once $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}$ is computed we simply have to check whether $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}} > 0.5$ to certify the input $\boldsymbol{x}$ w.r.t. the given radii r_a and r_d. The algorithm for the multi-class certificate $\mu_{\boldsymbol{x},\tilde{\boldsymbol{x}}}$ is similar.

G.5 Joint Certificates

It may be beneficial to specify different flip probabilities and radii for the graph and attributes. Let $\boldsymbol{x}^{\boldsymbol{A}} = \mathrm{vec}(\boldsymbol{A}) \in \{0,1\}^{n\times n}$ and $\boldsymbol{x}^{\boldsymbol{F}} = \mathrm{vec}(\boldsymbol{F}) \in \{0,1\}^{n\times m}$ denote the flattened adjacency and feature matrix respectively. Let $\boldsymbol{x} = [\boldsymbol{x}^{\boldsymbol{A}}, \boldsymbol{x}^{\boldsymbol{F}}] \in \mathcal{X}^{\boldsymbol{A},\boldsymbol{F}}$ where $\mathcal{X}^{\boldsymbol{A},\boldsymbol{F}} = \{0,1\}^{n\times n + n\times m}$. We apply the randomization schemes independently: for the graph $\phi(\boldsymbol{x}^{\boldsymbol{A}})$ with $p_+^{\boldsymbol{A}}, p_-^{\boldsymbol{A}}$, and for the attributes $\phi(\boldsymbol{x}^{\boldsymbol{F}})$ with $p_+^{\boldsymbol{F}}, p_-^{\boldsymbol{F}}$. We define the region:

$$\mathcal{R}_{q,q'}^{r_a^{\boldsymbol{A}},r_d^{\boldsymbol{A}},r_a^{\boldsymbol{F}},r_d^{\boldsymbol{F}}} = \{\boldsymbol{z} = [\boldsymbol{z}^{\boldsymbol{A}}, \boldsymbol{z}^{\boldsymbol{F}}] \in \mathcal{X}^{\boldsymbol{A},\boldsymbol{F}} : \boldsymbol{z}^{\boldsymbol{A}} \in \mathcal{R}_q^{r_a^{\boldsymbol{A}},r_d^{\boldsymbol{A}}}, \boldsymbol{z}^{\boldsymbol{F}} \in \mathcal{R}_{q'}^{r_a^{\boldsymbol{F}},r_d^{\boldsymbol{F}}}\}$$

where $\mathcal{R}_q^{r_a^{\boldsymbol{A}},r_d^{\boldsymbol{A}}}$ and $\mathcal{R}_{q'}^{r_a^{\boldsymbol{F}},r_d^{\boldsymbol{F}}}$ are defined similar to before. We have that the regions $\{\mathcal{R}_{0,0}^{r_a^{\boldsymbol{A}},r_d^{\boldsymbol{A}},r_a^{\boldsymbol{F}},r_d^{\boldsymbol{F}}}, \ldots, \mathcal{R}_{r_a^{\boldsymbol{A}}+r_d^{\boldsymbol{A}},r_a^{\boldsymbol{F}}+r_d^{\boldsymbol{F}}}^{r_a^{\boldsymbol{A}},r_d^{\boldsymbol{A}},r_a^{\boldsymbol{F}},r_d^{\boldsymbol{F}}}\}$ partition the space $\mathcal{X}^{\boldsymbol{A},\boldsymbol{F}}$. This follows directly due to the independence and the fact that the regions w.r.t. graph/attributes partition their respective spaces. The total number of regions is thus $(r_a^{\boldsymbol{A}} + r_d^{\boldsymbol{A}} + 1)(r_a^{\boldsymbol{F}} + r_d^{\boldsymbol{F}} + 1)$.

As before we can compute

$$\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_{q,q'}^{r_a^{\boldsymbol{A}}, r_d^{\boldsymbol{A}}, r_a^{\boldsymbol{F}}, r_d^{\boldsymbol{F}}}) = \Pr(\phi(\boldsymbol{x}^{\boldsymbol{A}}) \in \mathcal{R}_q^{r_a^{\boldsymbol{A}}, r_d^{\boldsymbol{A}}}) \cdot \Pr(\phi(\boldsymbol{x}^{\boldsymbol{F}}) \in \mathcal{R}_{q'}^{r_a^{\boldsymbol{F}}, r_d^{\boldsymbol{F}}})$$

Similarly we have for the ratio:

$$\eta_{q,q'}^{r_a^{\boldsymbol{A}}, r_d^{\boldsymbol{A}}, r_a^{\boldsymbol{F}}, r_d^{\boldsymbol{F}}} = \frac{\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_{q,q'}^{r_a^{\boldsymbol{A}}, r_d^{\boldsymbol{A}}, r_a^{\boldsymbol{F}}, r_d^{\boldsymbol{F}}})}{\Pr(\phi(\tilde{\boldsymbol{x}}) \in \mathcal{R}_{q,q'}^{r_a^{\boldsymbol{A}}, r_d^{\boldsymbol{A}}, r_a^{\boldsymbol{F}}, r_d^{\boldsymbol{F}}})} = \eta_q^{r_a^{\boldsymbol{A}}, r_d^{\boldsymbol{A}}} \cdot \eta_{q'}^{r_a^{\boldsymbol{F}}, r_d^{\boldsymbol{F}}}$$

The above directly follows from the definition of the regions and because $\phi(\boldsymbol{x}^{\boldsymbol{A}})$ is independent of $\phi(\boldsymbol{x}^{\boldsymbol{F}})$. Given the values of $\eta_{q,q'}$ and $\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_{q,q'})$ for all q, q' we can again apply the greedy algorithm to compute $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}$. Note that this can be trivially extended to certify arbitrary groupings of $\boldsymbol{x}$ into subspaces with different radii/flip probabilities per subspace, however, the complexity quickly increases and in general the number of regions will be $\mathcal{O}((r_a^{\max} + r_d^{\max} + 1)^v)$ where v is the number of groupings and $r_a^{\max}, r_d^{\max}$ are the maximum radii across the groupings.

G.6 Existing Graph Certificates Comparison

We compare our certificates with the only two existing works for certifying GNNs: Zügner and Günnemann [215]'s certificate which can only handle attacks on $\boldsymbol{F}$ and works for the GCN model [19]; and Bojchevski and Günnemann [5]'s certificate which can only handle attacks on $\boldsymbol{A}$ and works for a small class of models where the predictions are a linear function of (personalized) PageRank.

Both certificates specify local (per node) and global budgets/constraints, while our radii correspond to having only global budget. Therefore, to ensure a fair comparison we set their local budgets to be equal to their global budget which is equal to one of our radii, i.e. $q = Q = r_*$ for Zügner and Günnemann [215]'s certificate, and $b_v = B = r_*$ for Bojchevski and Günnemann [5]'s certificate. As we discussed in Sec. 8.6.1 we can only compare the certified robustness of the *base* classifier (existing certificates) versus the *smoothed* variant of the same classifier (our certificate).

Zügner and Günnemann [215]'s certificate does not distinguish between adding/deleting bits in the attributes so we compute a single radius corresponding to the total number of perturbations. For our certificate we evaluate two cases: (i) $r_d = 0$ and r_a varies; (ii) $r_a = 0$ and r_d varies. We use a different configuration of flip probabilities for each case. The certified ratio for all test nodes is shown on figure Fig. G.1. We see that our certificate is slightly better w.r.t. deletion and worse w.r.t. addition.

For Bojchevski and Günnemann [5]'s certificate we randomly select 50 test nodes to certify since solving their relaxed QCLP with global budget is computationally expensive. We evaluate the robustness of the (A)PPNP model, and we focus on edge removal since their global budget certificate for edge addition took more than 12h to complete. That

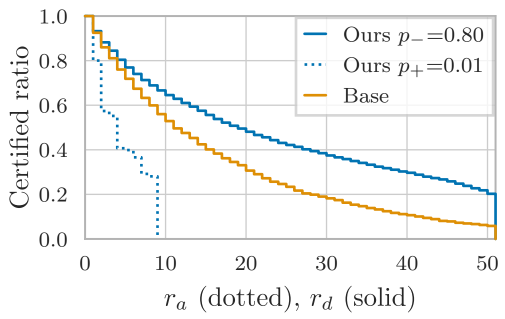


Figure G.1: Comparison between our certificate of the smoothed GCN classifier and Zügner and Günnemann [215]'s certificate of the base GCN classifier. We are certifying w.r.t. the attributes on Cora-ML. Solid lines denote r_d (with $r_a = 0$) and dotted lines denote r_a (with $r_d = 0$).

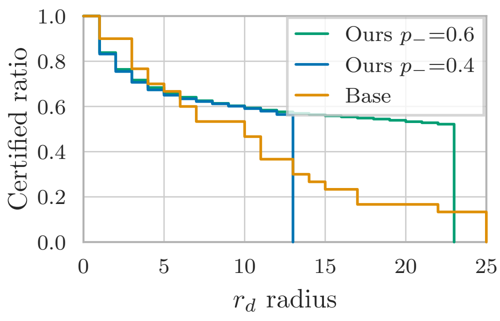


Figure G.2: Comparison between our certificate of the smoothed PPNP classifier and Bojchevski and Günnemann [5]'s certificate of the base PPNP classifier. We are certifying edge deletion on Cora-ML. Our certificate is significantly better despite the fact that we are certifying undirected edges.

is, we configure the set of fragile edges $\mathcal{F}$ to contain only the existing edges (except the edges along the minimum spanning tree which are fixed). The results for different values of p_- (for $p_+ = 0$) are show in Fig. G.2. We see that we can certify significantly more nodes, especially as we increase the radius. Note that the effective certified radius for our approach is double of what is shown in Fig. G.2 since we are certifying undirected edges, while Bojchevski and Günnemann [5]'s certificate is w.r.t. directed edges.

G.7 Training

To investigate the effect of smooth training [228] on certified robustness we approximate the smoothed probability $\boldsymbol{g}_y(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{x}' \sim \phi(\boldsymbol{x})}[f(\boldsymbol{x}')_y]$ for class y with m Monte Carlo samples $\boldsymbol{g}_y(\boldsymbol{x}) \approx \sum_{i=1}^{m} f(\boldsymbol{x}^{(i)})_y$, and we compute the cross-entropy loss with $l(\boldsymbol{g}(\boldsymbol{x}), y)$. Note that $m = 1$ is equivalent to training f with noisy inputs. We vary the number of Monte Carlo samples m we use during training for a fixed value of $p_+ = 0.01, p_- = 0.6$. Fig. G.3 shows the results when perturbing the attributes on Cora-ML using GCN as a base classifier. Specifically, we show the difference (Δ) in the certified ratio relative to standard (non-smoothed) training, i.e. $m = 0$. We see that including the perturbations during training ($m > 0$) is consistently better than standard training ($m = 0$). The difference for different values of m is relatively small overall, with $m = 1$ being the best. Therefore, for all experiments we set $m = 1$.

G.8 Graph Classification

For most experiments we focused on the node-level classification task. However, our certificate can be trivially adapted for the graph-level classification task. Currently, there are no other existing certificate that can handle this scenario. Given any classifier f that

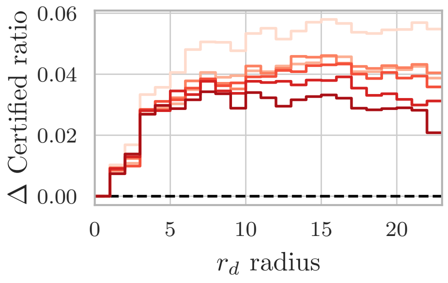


Figure G.3: The difference (Δ) in the certificate ratio relative to $m = 0$ (standard training, dashed black line). The color gradient denotes $m \in \{1, 5, 10, 25, 50, 100\}$ with darker colors corresponding to higher m. The difference is relatively small overall, and $m = 1$ (lightest color) is best.

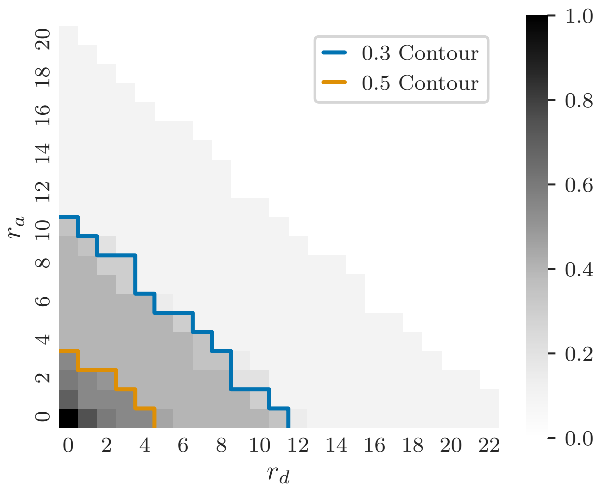


Figure G.4: Certifying graph-level classification w.r.t. perturbations of the graph structure on the MUTAG dataset. We set $p_+ = 0.2$ and $p_- = 0.4$. We can certify a high ratio of graphs for r_a and r_d.

takes a graph G_i as an input and outputs (a distribution over) graph-level classes, we can form the smoothed classifier g by randomly perturbing G_i, e.g. by applying ϕ on $\boldsymbol{x} = \text{vec}(\boldsymbol{A}_i)$ where $\boldsymbol{A}_i$ is the adjacency matrix of the graph G_i. Then, we certify g simply by calculating $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}$ or $\mu_{\boldsymbol{x},\tilde{\boldsymbol{x}}}$. The certificates are still efficient to compute and independent of the graph size.

To demonstrate the generality of our certificate we train GIN on the MUTAG dataset, which consists of 188 graphs corresponding to chemical compounds. The graphs are divided into two classes according to their mutagenic effect on bacteria. The results are shown in Fig. G.4. We see that we can certify a high ratio of graphs for both r_a and r_d. Similar results hold when perturbing the node features.

G.9 Hyperparameters

For node classification, for all GNN models we randomly select 20 nodes from each class for the training set, and 20 nodes for the validation set. We train the models for a maximum of 3000 epochs with a fixed learning rate of 10^{-3} and patience of 50 epochs for early stopping. We optimize the parameters with Adam and use a weight decay of 10^{-3}. For GCN and APPNP we use a single hidden layer of size 64, and we set the hidden size for GAT to 8 and use 8 heads to match the number of trainable parameters. For MNIST and ImageNet we use the standard train/validation/test split, and we train a CNN classifier with the same configuration as described in Lee et al. [220]. We set the confidence level $\alpha = 0.01$ and the number of samples to 10^6 (10^5 for MNIST and ImageNet). For all experiments, we use our multi-class certificate since it yields slightly

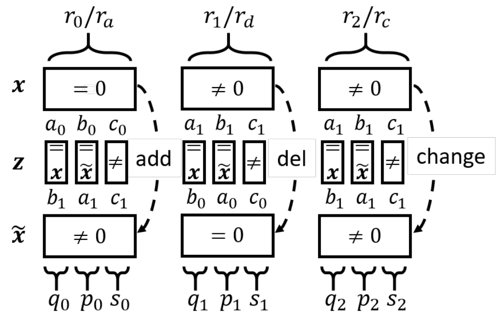


Figure G.5: Illustration of the regions for the general sparsity-aware discrete certificate. We only show the dimensions $\mathcal{C}$ where $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ disagree. The triplets (q_j, p_j, s_j) are used to parametrize the regions. The variables a_0, b_0, c_0, and a_1, b_1, c_1 depend on the flip probabilities p_+, p_- and the number of categories K (see text).

higher certified ratios compared to the binary-class certificate (see Sec. 8.8.1). Note that to certify an input w.r.t. $\mathcal{B}_{r_a,r_d}(\boldsymbol{x})$ we can simply compute the certificate w.r.t. $\mathcal{S}_{i,j}(\boldsymbol{x})$ for all $0 \leq i \leq r_a, 0 \leq j \leq r_d$. In practice, we compute the maximum r_a and r_d for a given $\underline{p_{y^*}(\boldsymbol{x})}$ and $\overline{p_{\tilde{y}}(\boldsymbol{x})}$ such that the input is certifiably robust. Whenever the number of majority votes is the same for several inputs, they have the same $\underline{p_{y^*}(\boldsymbol{x})}$ and $\overline{p_{\tilde{y}}(\boldsymbol{x})}$ so we only need to compute the maximum radii once to certify all of them.

G.10 Limitations

The main advantage of the randomized smoothing technique is that we can utilize it without making any assumptions about the base classifier f since to compute the certificate we need to consider only the output of f for each sample. This is also one of its biggest disadvantages since it does not take into account any properties of f, e.g. smoothness. More importantly, when applied for certifying graph data we can additionally leverage the fact that the predictions for neighboring nodes are often highly correlated, especially when the graph exhibits homophily. Extending our certificate to account for these aspects is a viable future direction.

Moreover, to accurately estimate $\underline{p_y(\boldsymbol{x})}$ we need a large number of samples (e.g. we used 10^6 samples in our experiments). Even though one can easily parallelize the sampling procedure developing a more sample-efficient variant is desirable. Finally, the guarantees provided are probabilistic, the certificate holds with probability $1 - \alpha$, and as shown in previous work [217, 220] the number of samples necessary to certify at a given radius grows as we increase our confidence, i.e. decrease α.

G.11 Certificate for Discrete Data

As before, since the randomization scheme which we defined in Sec. 8.5 is applied independently per dimension w.l.o.g. we can focus only on those dimensions $\mathcal{C}$ where

$\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ disagree. We omit all proofs for the discrete case since they are analogous to the binary case. The only difference is in how we partition the space $\mathcal{X}_K$ and how we compute the respective regions. Once we obtain the regions the computation of $\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}$ or $\mu_{\boldsymbol{x},\tilde{\boldsymbol{x}}}$ and hence the certificate is the same.

Intuitively, we have variables q_0, q_1, q_2 corresponding to the dimensions where $\boldsymbol{z}_\mathcal{C}$ matches $\boldsymbol{x}_\mathcal{C}$, variables p_0, p_1, p_2 corresponding to the dimensions where $\boldsymbol{z}_\mathcal{C}$ matches $\tilde{\boldsymbol{x}}_\mathcal{C}$, and variables s_0, s_1, s_2 corresponding to the dimensions where $\boldsymbol{z}_\mathcal{C}$ matches neither $\boldsymbol{x}_\mathcal{C}$ nor $\tilde{\boldsymbol{x}}_\mathcal{C}$ (see illustration in Fig. G.5). The fourth-case where $\boldsymbol{z}_\mathcal{C}$ matches both $\boldsymbol{x}_\mathcal{C}$ and $\tilde{\boldsymbol{x}}_\mathcal{C}$ is not possible since by definition $\boldsymbol{x}_i \neq \tilde{\boldsymbol{x}}_i$ for all $i \in \mathcal{C}$. We define the region parametrized by (q_j, p_j, s_j) triplets:

$$\begin{aligned}
\mathcal{R}_{\substack{q_0,q_1,q_2\\p_0,p_1,p_2\\s_0,s_1,s_2}} &= \{\boldsymbol{z} \in \mathcal{X}_K : \\
q_0 &= \sum_{i\in\mathcal{C}} \mathbb{I}(\boldsymbol{z}_i = \boldsymbol{x}_i)\mathbb{I}(\boldsymbol{x}_i = 0),\\
q_1 &= \sum_{i\in\mathcal{C}} \mathbb{I}(\boldsymbol{z}_i = \boldsymbol{x}_i)\mathbb{I}(\tilde{\boldsymbol{x}}_i = 0),\\
q_2 &= \sum_{i\in\mathcal{C}} \mathbb{I}(\boldsymbol{z}_i = \boldsymbol{x}_i)\mathbb{I}(\tilde{\boldsymbol{x}}_i \neq 0)\mathbb{I}(\boldsymbol{x}_i \neq 0),\\
p_0 &= \sum_{i\in\mathcal{C}} \mathbb{I}(\boldsymbol{z}_i = \tilde{\boldsymbol{x}}_i)\mathbb{I}(\boldsymbol{x}_i = 0),\\
p_1 &= \sum_{i\in\mathcal{C}} \mathbb{I}(\boldsymbol{z}_i = \tilde{\boldsymbol{x}}_i)\mathbb{I}(\tilde{\boldsymbol{x}}_i = 0),\\
p_2 &= \sum_{i\in\mathcal{C}} \mathbb{I}(\boldsymbol{z}_i = \tilde{\boldsymbol{x}}_i)\mathbb{I}(\boldsymbol{x}_i \neq 0)\mathbb{I}(\tilde{\boldsymbol{x}}_i \neq 0),\\
s_0 &= \sum_{i\in\mathcal{C}} \mathbb{I}(\boldsymbol{z}_i \neq \tilde{\boldsymbol{x}}_i)\mathbb{I}(\boldsymbol{z}_i \neq \boldsymbol{x}_i)\mathbb{I}(\boldsymbol{x}_i = 0),\\
s_1 &= \sum_{i\in\mathcal{C}} \mathbb{I}(\boldsymbol{z}_i \neq \tilde{\boldsymbol{x}}_i)\mathbb{I}(\boldsymbol{z}_i \neq \boldsymbol{x}_i)\mathbb{I}(\tilde{\boldsymbol{x}}_i = 0),\\
s_2 &= \sum_{i\in\mathcal{C}} \mathbb{I}(\boldsymbol{z}_i \neq \tilde{\boldsymbol{x}}_i)\mathbb{I}(\boldsymbol{z}_i \neq \boldsymbol{x}_i)\mathbb{I}(\boldsymbol{x}_i \neq 0)\mathbb{I}(\tilde{\boldsymbol{x}}_i \neq 0)\}
\end{aligned}$$

for a given clean $\boldsymbol{x} \in \mathcal{X}_K$ and adversarial $\tilde{\boldsymbol{x}} \in \mathcal{S}_{r_0,r_1,r_2}(\boldsymbol{x})$ which is defined subsequently.

We use $a_0 = 1 - p_+$ as a shorthand for the probability to keep (not flip) a zero, $b_0 = \frac{p_+}{K-1}$ for the probability to flip a zero to some other value, and $c_0 = 1 - a_0 - b_0$. Similarly we define $a_1 = 1 - p_-$, $b_1 = \frac{p_-}{K-1}$, and $c_1 = 1 - a_1 - b_1$ for the non-zero values. We can easily verify from the definitions that given a specific configuration of q_j, p_j, s_j

variables the ratio for the corresponding $\mathcal{R}_{\substack{q_0,q_1,q_2\\p_0,p_1,p_2\\s_0,s_1,s_2}}$ region equals:

$$\begin{aligned}\eta &= \Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_{\substack{q_0,q_1,q_2\\p_0,p_1,p_2\\s_0,s_1,s_2}}) / \Pr(\phi(\tilde{\boldsymbol{x}}) \in \mathcal{R}_{\substack{q_0,q_1,q_2\\p_0,p_1,p_2\\s_0,s_1,s_2}})\\ &= \left(\frac{a_0}{b_1}\right)^{q_0-p_1}\left(\frac{b_0}{a_1}\right)^{p_0-q_1}\left(\frac{c_0}{c_1}\right)^{s_0-s_1}\left(\frac{a_1}{b_1}\right)^{q_2-p_2}\end{aligned} \tag{G.5}$$

Furthermore, we define $r_j = q_j + p_j + s_j$ for $j = 0, 1, 2$. Now, we can compute the probability for $\phi(\boldsymbol{x})$ to land in the respective region as a product of Multinomials:

$$\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_{\substack{q_0,q_1,q_2\\p_0,p_1,p_2\\s_0,s_1,s_2}}) = \prod_{j=0}^{2} \Pr(\boldsymbol{u}_j = [q_j, p_j, s_j]) \tag{G.6}$$

where $\boldsymbol{u}_j$ are the following Multinomial random variables:

$$\begin{aligned}\boldsymbol{u}_0 &\sim \mathrm{Mul}([a_0, b_0, c_0], r_0)\\ \boldsymbol{u}_1 &\sim \mathrm{Mul}([a_1, b_1, c_1], r_1)\\ \boldsymbol{u}_2 &\sim \mathrm{Mul}([a_1, b_1, c_1], r_2)\end{aligned}$$

These variables have only 3 categories regardless of the number of discrete categories in the input space. This is due to the fact that we only need to keep track of 3 states: $\boldsymbol{z}_i = \boldsymbol{x}_i$, $\boldsymbol{z}_i = \tilde{\boldsymbol{x}}_i$, and $\boldsymbol{x}_i \neq \boldsymbol{z}_i \neq \tilde{\boldsymbol{x}}_i$ for all $i \in \mathcal{C}$.

This construction suggests that we should parametrize our threat model with three radii: r_0/r_a which counts the number of added non-zeros, r_1/r_d which counts the number of removed non-zeros, and r_2/r_c which counts how many non-zeros changed to another non-zero value. We have:

$$\begin{aligned}\mathcal{S}_{r_0,r_1,r_2}(\boldsymbol{x}) = \{\tilde{\boldsymbol{x}} \in \mathcal{X}_K : &\sum_{i=1}^{d} \mathbb{I}(\boldsymbol{x}_i = 0)\mathbb{I}(\boldsymbol{x}_i \neq \tilde{\boldsymbol{x}}_i) = r_0,\\ &\sum_{i=1}^{d} \mathbb{I}(\tilde{\boldsymbol{x}}_i = 0)\mathbb{I}(\boldsymbol{x}_i \neq \tilde{\boldsymbol{x}}_i) = r_1,\\ &\sum_{i=1}^{d} \mathbb{I}(\boldsymbol{x}_i \neq 0)\mathbb{I}(\tilde{\boldsymbol{x}}_i \neq 0)\mathbb{I}(\boldsymbol{x}_i \neq \tilde{\boldsymbol{x}}_i) = r_2\}\end{aligned}$$

Similarly, we define the respective ball $\mathcal{B}_{r_0,r_1,r_2}(\boldsymbol{x})$ by replacing equalities with inequalities.

We can directly verify that for the binary case ($K = 2$), r_2 necessarily has to be equal to 0. We recover the definition of our threat model for binary data. Moreover, all s_i's, as well as $c_0 = \frac{(K-2)\cdot p_+}{K-1}$ and $c_1 = \frac{(K-2)\cdot p_-}{K-1}$ also have to be zero.

In order to partition the entire space $\mathcal{X}_K$ we have to generate all unique (q_j, p_j, s_j) triplets where $q_j + p_j + s_j = r_j$. There are $T_j = (r_j+1)(r_j+2)/2$ unique (q_j, p_j, s_j) triplets for $j = 0, 1, 2$. Therefore, the total number of regions is upper bounded by $T_0 \cdot T_1 \cdot T_2$. Note that this is an upper bound since the ratio in Eq. G.5 is the same for certain combinations of q_j's, p_j's, and s_j's, e.q. when $q_0 - p_1 = 1 - 3 = 2 - 4$ and similarly for $p_0 - q_1$, $s_0 - s_1$, and $q_2 - p_2$. In these cases we can merge these regions into one region.

The overall computation of the regions is efficient and it consists of: (i) generating all unique (q_j, p_j, s_j) triplets; (ii) computing the ratio defined in Eq. G.5; and (iii) computing the probability for $\phi(\boldsymbol{x})$ to land in the respective region using Eq. G.6. Since the number of regions is small the overall runtime is less than a second. We provide a reference implementation in Python with further details.

For the special case of $p_+ = p_-$ we have that $a_0 = a_1$, $b_0 = b_1$, and $c_0 = c_1$. Then the ratio in Eq. G.5 simplifies to:

$$\eta = \left(\frac{a_0}{b_1}\right)^{q_0+q_1+q_2-p_0-p_1-p_2} = \left(\frac{a_0}{b_1}\right)^{q'-p'} \tag{G.7}$$

where we set $q' = q_0 + q_1 + q_2$ and $p' = p_0 + p_1 + p_2$. This directly implies that in this case we do not need to keep track of the different (q_j, p_j, s_j) triplets, but rather it is sufficient to parametrize the region with two variables, namely q' and p'. The probability that $\phi(\boldsymbol{x})$ lands in the respective $\mathcal{R}_{q',p'}$ region also simplifies (see Fig. G.5):

$$\Pr(\phi(\boldsymbol{x}) \in \mathcal{R}_{q',p'}) = \Pr(\boldsymbol{u} = [q', p', r - q' - p']) \tag{G.8}$$

where $\boldsymbol{u} \sim \text{Mul}([a_0, b_0, c_0], r)$. Moreover, we have that $q' \in \{0, \ldots, r_0 + r_1 + r_2\} = \{0, \ldots, r\}$, where $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_0 = r$. Similarly, $p' \in \{0, \ldots, r\}$. It follows that $(q' - p') \in \{-r, \ldots, r\}$, and thus there are only $2r + 1$ regions in total.

G.12 Further Analysis of Joint Certificates

On Fig. G.6 we show our method's ability to certify robustness against combined perturbations on the graph and the attributes. The configuration of flip probabilities is the same as in Sec. 8.8.1. Specifically to show different aspects of the 4D heatmap (certified ratio w.r.t. the 4 different radii) we plot all pairwise heatmaps, e.g. $r_a^{\boldsymbol{A}} = r_d^{\boldsymbol{F}} = 0$ and varying $r_d^{\boldsymbol{A}}, r_a^{\boldsymbol{A}}$. The figure is symmetric w.r.t. the diagonal, which shows the certified ratio as we fix all radii except one to 0. Similar to before we observe that we can certify more easily w.r.t. r_a compared to r_d. Since we are perturbing both features and structure at the same time we can obtain only modest certified radii. We leave it for future work to design models that are robust to such joint perturbations.

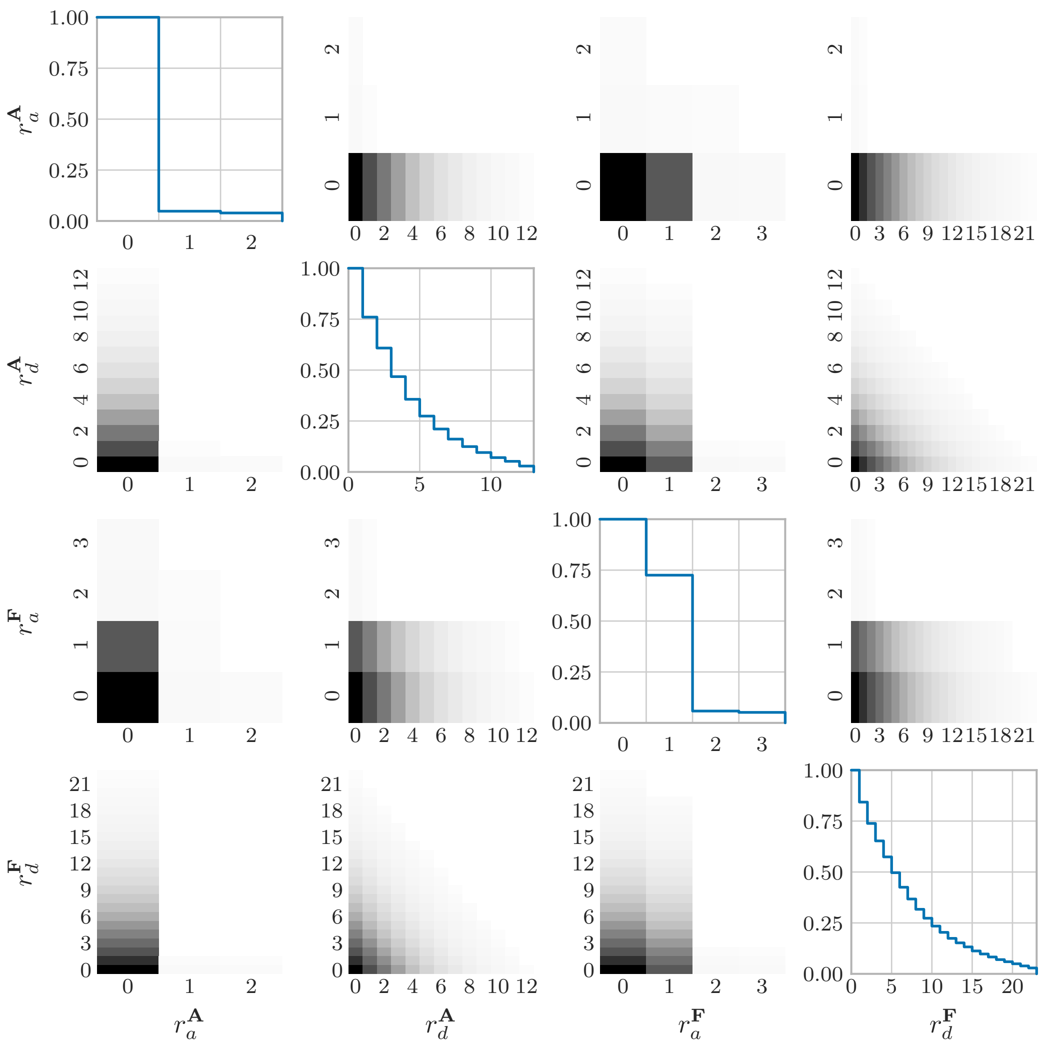


Figure G.6: Joint certificate for both graph and attributes on Cora-ML. We show all pairwise heatmaps, e.g. $r_a^{\boldsymbol{A}} = r_d^{\boldsymbol{F}} = 0$ and varying $r_d^{\boldsymbol{A}}, r_a^{\boldsymbol{A}}$. The figure is symmetric w.r.t. the diagonal, which shows the certified ratio as we fix all radii except one to 0.