CHALMERS

Graph Classification with Differential Privacy

Master of Science Thesis in Complex Adaptive Systems

OTTO FROST
CARL RETZNER

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, November 2014

Graph Classification and Differential Privacy

OTTO FROST,
CARL RETZNER

Examiner: DEVDATT DUBHASHI
Supervisor: FREDRIK JOHANSSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden November 2014

Abstract

With increasing usage of online services such as email, social networks and online shopping more data than ever before is being gathered. Many types of data for which structure, flow and relationships are important may naturally be represented by graphs. Services may want to use machine learning algorithms to gain insight about their customers or users, and to successfully use the concepts of machine learning they must take the structure and properties of graph data into account. Seemingly innocuous data could potentially be used to infer sensitive information about individuals and should be kept private, the algorithms and techniques used must therefore take privacy into consideration.

In this thesis we consider the combination of machine learning and privacy by bringing together the concepts of support vector machines and differential privacy. We examine the classification of graphs by means of kernel methods and present a framework for constructing private representations of two well known graph kernels, the random walk kernel and graphlet kernels. Furthermore, to allow for classification of large graphs we present a novel sampling scheme for approximation of subgraph counts. We evaluate both sampling and classification using four real world datasets consisting of social, road and protein networks.

Keywords: support vector machines, differential privacy, classification, sampling

Acknowledgements

We want to express our gratitude for all the helpful discussions and support from our supervisor Fredrik Johansson. We also recognize the valuable insights of Professor Devdatt Dubhashi without whom this work would not have been possible.

Otto Frost and Carl Retzner
Göteborg, November 28, 2014

Contents

Chapter 1

Introduction

As society adapts to new technology a lot of previously private data is recorded by various services such as social networks, email hosts and other online agents. In this digital revolution many users of these services are eager to accept terms and conditions that may have unexpected consequences. Social networking sites keep records of known affiliations, email services may keep tabs on who receives certain kinds of emails and online stores knows what products that are bought and browsed. Today, online privacy has become a hot topic and users have been alerted to the potential dangers of sharing certain information. Seemingly innocuous data can in some cases be used to infer highly sensitive and personal information about the individuals.

Since these services have been widely adopted by the general public, the task of analyzing the gathered data has gained importance. Anticipating the user's behavior and needs has become an important part of the ever changing online marketplace. Functions such as "people you may know", "spam filtering" and "products you may be interested in" are all heavily based on algorithms for statistical classification and regression. Social networks, email correspondence and product browsing may be naturally represented as graphs, and to successfully use machine learning algorithms to gain insight about the users and customers one needs to take the structure and property of graph data into account. However, information used to enhance the service may also reveal private information about the users. This highlights the questions and problems we treat in this thesis.

The idea of combining privacy and machine learning for graphs is of increasing importance and has no obvious solutions. While preserving privacy, the output of a machine learning algorithm must still be useful for the data curator. Demands on scalability and accuracy of the data analysis are also increasing as the area of application is growing. In this thesis we examine the possibility of combining graph classification and privacy, we provide a novel sampling scheme to allow classification of large graphs and perform an extensive evaluation of our findings.

1.1 Introduction to Privacy

We highlight the importance of sufficient privacy constraints by two examples of inadequate privacy. First, we must introduce the concept of *auxiliary information* which will be present throughout this thesis. Auxiliary, or external, information is the information an adversary has about some record of a database without access to the database itself. For instance, an adversary knows the age of a certain person in a medical database without access to the medical database itself. We must also introduce the concept of *quasi identifiers* to analyze the examples. Quasi identifiers are pieces of information that are not by themselves unique identifiers, but are correlated to an extent where they can be combined to uniquely identify entities such as records in a database.

- **Medical database [Sweeney, 2013]**
  In medical databases patients' personal information and medical diagnoses are stored, but quasi identifiers such as names and residence addresses are not included. Medical diagnoses are considered to be personal pieces of information and should not be disclosed without consent. Sweeney [2013] showed that by combining anonymized hospital records on sale for research purposes and newspaper stories containing the word "hospitalized", it was possible to exactly match 35 of 81 medical records to the names of individuals. This information could be used by anyone with interest in the medical situation of individuals e.g. insurance companies or creditors. In this case the medical data is considered to be a private database since quasi identifiers have been removed and the newspaper stories are considered to be auxiliary information.

- **Netflix Prize database [Narayanan and Shmatikov, 2008]**
  Online services such as video streaming strive to predict the user's next choice. The video streaming service Netflix released a subset of their user submitted data as part of a programming challenge known as the "Netflix Prize". The database consisted of unique user reviews of video content where quasi identifiers were removed, and some random noise added. The aim of the challenge was to improve the recommender system, but accidentally Netflix had compromised the privacy for a subset of their customers. Narayanan and Shmatikov [2008] showed that by

using auxiliary information, from public online profiles of the International Movie Database (IMDb), it was possible to uniquely identify users in the Netflix database.

Even though steps were taken in both cases to preserve privacy of the data by removing quasi identifiers, the integrity of records in the databases was compromised with the aid of auxiliary information. It is clear that there is a need for privacy and privacy preserving mechanisms in the field of statistical data gathering and analysis. For a curator of data this means carefully choosing what data to publish and also how to compute statistics in order to maintain privacy.

Considering privacy for databases, there are two different approaches to publishing statistics. The non-interactive setting is what is described in the examples above, when the data curator publishes a sanitized version of the database. In contrast, the interactive setting only allows an adversary to ask questions and get some answer in return.

We can intuitively think of the two extreme cases, complete privacy and no guarantee at all on privacy. Complete privacy can be achieved by giving completely random answers, or equivalently no answers at all. The case of no privacy guarantees is realized when the true answer is always returned unperturbed. The aim of privacy preserving mechanisms is to find a middle ground, where the answers to questions on the database are *useful* but do not reveal too much information. We will throughout this thesis consider this sense of useful answers given by a private mechanism as the *utility* of the mechanism.

In this thesis we consider a mechanism for privacy preservation in statistical databases known as *differential privacy*. We can think of differential privacy intuitively as a way of making sure that the presence or absence of a single record in a database will not change the answer of a question on that database by very much [Dwork et al., 2006]. We will describe and thoroughly define this concept later on.

Consider a study that was to be carried out in a differentially private setting, in which the participants are required to provide sensitive data about themselves. Alice is asked to join the study, but being concerned about providing her data, she is hesitant whether to participate or not. The researchers behind the study can argue that whether Alice participates in the study or not will not affect the analysis significantly. Considering Alice alone, it will not matter much if she is a part of the study or not. Differential privacy guarantees that the answer of a question on the database will not differ by much regardless of whether Alice joins or not.

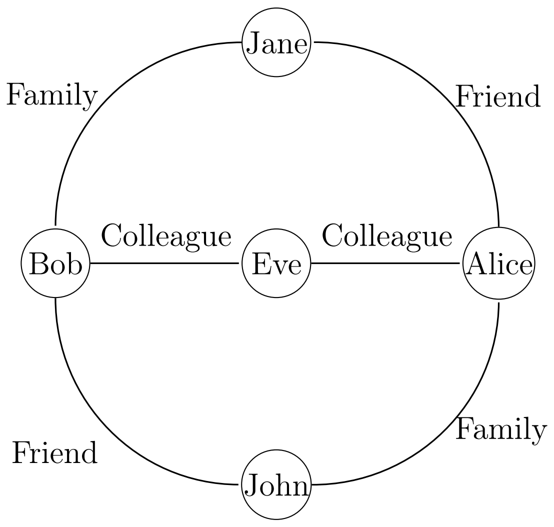


Figure 1.1: A social network represented as a graph, node labels indicate identities and edge labels indicate type of relationship.

1.2 Introduction to Graph Classification

The cost of data storage and computational power has decreased dramatically in the last years and the techniques for distributed computing has become available to the general public. As the sheer volume of data stored by various service providers reaches unprecedented levels, a need for efficient and accurate algorithms to analyze this huge amount of data emerges. This setup makes algorithms for machine learning in general, and classification in particular, more important than ever for companies and researchers.

In this thesis we consider data represented as graphs consisting of vertices and edges. There are many types of data that can be represented as graphs such as social networks, airline connections, city road networks, molecules and other systems where structure, flow and connections are of importance. They may provide representations of relations, distances, flows, importance and many other attributes which arise naturally from certain types of data.

The vertices in a graph typically represents units or entities of some kind, be it persons, molecules, atoms or locations. Each unit may be connected to any other unit by friendship, atomic bonds or distances. These connections are often represented by edges in the graph. To further enrich the representation of data, labels can be added to each edge and node to specify importance, strength or identity. A toy example of graph representation for a social network is presented in Figure 1.1.

For simple measurements which gives a sequence of numbers as results one might be interested in statistics as the mean value and the standard error. However, when the data is represented as a graph, other quantities are of interest since graphs are more difficult to compare. Graphs can exhibit patterns on both local and global scale and there are many interesting features of graphs such as diameter, minimum and maximum spanning trees, clustering and centrality to only mention a few.

There exists methods for classifying data expressed as vectors [Schölkopf and Smola, 2002] as well as graphs [Gärtner et al., 2003, Borgwardt and Kriegel, 2005, Shervashidze et al., 2009]. In both cases, an essential choice which impacts performance and accuracy of classification is the choice in feature representation of the data.

1.3 Problem formulation

Publishing statistics of datasets may compromise the privacy of the entities represented even if quasi identifiers have been removed and the contents of the dataset seems innocuous. The methods for graph classification and frameworks for privacy can be combined, to enable classification that does not compromise the privacy of the entities considered. This thesis investigates the possibility of constructing differentially private graph kernels and aims at presenting a general framework for constructing private counterparts to existing graph kernels. This thesis is to the best of our knowledge the first research on the combination of differential privacy and graph kernels.

Computational complexity increases dramatically for large graphs, and exact kernels become prohibitively expensive to compute. To remedy this we aim at presenting a sampled version of the graphlet kernel and extensively evaluate the performance of said sampling.

1.4 Related work

This thesis cover topics in the fields of privacy and kernel methods and thus we consider the research and related work for both of these fields. First we present related work on graph classification and then privacy.

Gärtner et al. [2003] introduces the concept of graph kernels in general and path kernels in particular. The random walk kernel [Gärtner et al., 2003] was the first graph kernel, followed by the shortest path kernel [Borgwardt and Kriegel, 2005]. Later Vishwanathan et al. [2010] showed a more efficient implementation of the random walk kernel using Sylvester equations.

Borgwardt [2007], Vacic [2008] introduced the graphlet kernels which are based on subgraph counting. Shervashidze et al. [2009] implements a sampled version to deal with larger graphs.

The family of Weisfeiler-Lehman kernels were proposed by Shervashidze et al. [2011] based on the Weisfeiler-Lehman test of isomorphism.

The concept of differential privacy was introduced by Dwork et al. [2006]. The application of differential privacy for graphs was first brought up by Nissim et al. [2007] who considers private triangle counting. Hay et al. [2009] considers private release of degree distribution for graphs.

Further analysis of graphs have been studied by Karwa et al. [2011] focusing on private release of counts of various subgraphs. Blocki et al. [2013] proposes the concept of restricted sensitivity for private subgraph counts and Kasiviswanathan et al. [2013] considers an LP-based technique for achieving the same goal.

1.5 Our contribution

In this thesis we bring together graph classification and differential privacy in order to define differentially private graph kernels. We will perform an extensive evaluation of the private kernels by classifying a well known benchmark dataset as well as three real world datasets compiled for this thesis.

The aim of the thesis is to give empirical results and valuable insights for further development of differentially private graph kernels, which to the best of our knowledge have not been implemented before. We also examine how the expressiveness of kernels relate to private classification and evaluate levels of privacy in relation to utility.

In light of recent developments within the field of data storage and gathering we also present a sampling scheme for classification of large, sparse graphs. The novel sampling algorithm is used to approximate subgraph counts and is both theoretically and practically evaluated.

1.6 Scope of thesis

We only consider differential privacy in the interactive setting as mentioned in Section 1.1. The algorithms and techniques used allow an adversary to ask questions about the database and get an answer in return but will not be able to output a sanitized version of the database in question.

For the evaluation and construction of private kernels we consider the connected graphlet kernels and p-random walk kernels in particular. The framework for differentially private kernels presented however is applicable for any graph kernel that has an explicit feature representation.

Evaluation of sampling and classification is only considered for the four datasets described in the thesis.

1.7 Thesis Outline

In Chapter 2 we introduce the definitions and concepts of graph theory, differential privacy and classification methods. This will serve as a theoretical base upon which the definitions and algorithms of Chapter 3 are built.

We define the framework of differentially private graph kernels and use this framework to construct private versions of the random walk kernel and graphlet kernels in Chapter 3. We also present a novel sampling scheme for approximation of subgraph counts by sampling edges.

We start Chapter 4 by presenting the datasets used for this thesis. Experiments and evaluation of the proposed kernels and sampling scheme are also described and presented in Chapter 4.

Lastly, in Chapter 5 we discuss the results of all experiments and comment on possible future work.

Chapter 2

Preliminaries

In this chapter we present basic concepts regarding graph theory, privacy, graph classification and sampling. The outline is as follows: in Section 2.1 basic concepts from graph theory are introduced. In Section 2.2 we introduce the notion of differential privacy. We provide general considerations regarding privacy for databases and graphs in particular but also the rigorous definitions of differential privacy and sensitivity. In Section 2.3 we survey some of the graph specific applications of differential privacy that exists. Finally, in Section 2.4 we present the problem of graph classification together with some of the existing results and applications.

2.1 Basic Graph Theory

Let $G = (V, E)$ be a graph with a set of vertices, $V(G) = V$, edges, $E(G) = E$ and adjacency matrix A. We denote the number of vertices by $n = |V|$ and the number of edges by $m = |E|$. We let $\mathcal{G}$ denote the set of all graphs. The degree of a vertex corresponds to the number of edges adjacent to the vertex and the degree of vertex i is denoted as $\deg(i)$. The set $\{1, 2, ..., n\}$ is denoted by $[n]$. We use the following definitions of distance and graph adjacency.

Definition 2.1 (Rewiring distance)**.** The rewiring or vertex distance, $d_{vertex}(G, G')$, between two graphs G and G' is the minimum number of vertices that needs to be rewired in G to obtain G'

Rewiring of a vertex includes addition of a new vertex with arbitrary edges, deletion of a vertex and all of its edges and arbitrary change to a vertex's adjacency list.

Definition 2.2 (Vertex neighbors)**.** Two graphs G and G' are vertex neighbors if $d_{vertex}(G, G') = 1$.

Table 2.1: The number of connected graphlets (N_k) and the total number of graphlets (${N_k}'$) for each graphlet size, k.

k	N_k	${N_k}'$
3	2	4
4	6	11
5	21	34

Definition 2.3 (Edge distance)**.** The edge distance, $d_{edge}(G, G')$, between two graphs G and G' is the minimum number of edges that needs to be added or deleted from G to obtain G'.

Definition 2.4 (Edge neighbors)**.** Two graphs G and G' are edge neighbors if $d_{edge}(G, G') = 1$.

Where applicable, we drop the subscript of $d_{vertex}(G, G')$ or $d_{edge}(G, G')$ and let $d(G, G')$ denote vertex or edge distance.

A subgraph $H = (V_H, E_H)$ of $G = (V, E)$ is a graph where $V_H \subset V$ and $E_H \subset E$. If all of the edges that connect the vertices V_H in G are in H, we say that H is induced. That is, H is induced if $H = G[V(H)]$. We say that G is the *supergraph* of H.

A special class of subgraphs are the graphlets. A graphlet is a small induced subgraph, usually with 3-5 vertices. The size of a graphlet is denoted by k and the number of unique graphlet patterns of size k is denoted by N_k for the connected graphlets and ${N_k}'$ for the total number of graphlets. The relationship between k, N_k and ${N_k}'$ is displayed in Table 2.1. We denote the set of graphlets of size k by $\mathcal{H}^{(k)} = \{H_i^{(k)}\}_i^a$, where a corresponds to either N_k or ${N_k}'$. $H_i^{(k)}$ is a graphlet pattern of size k and type i. For a graph $G = (V, E) \in \{G \in \mathcal{G} : |V| > k\}$, the number of graphlets $f_G^{(k)}(i)$ of type i is

$$f_G^{(k)}(i) = |\{U \subset V : |U| = k, G[U] \cong H_i^{(k)}\}| \ , \tag{2.1}$$

where $G[U] \cong H_i^{(k)}$ corresponds to $G[U]$ and $H_i^{(k)}$ being isomorphic. Two graphs G and G' are isomorphic if they only differ in enumeration and not in structure. Mathematically, an isomorphism of two graphs G and G' is the bijective function on the vertex sets of G and G', $f : V(G) \to V(G')$ such that $(i, j) \in E(G) \iff (f(i), f(j)) \in E(G')$. If such a function exists, G and G' are said to be isomorphic [Bollobás, 1998, pp. 3].

2.2 Differential Privacy

Consider a database which is representative for an underlying population. Statistics of this database are then representative of the underlying population. The task of preserving privacy for such a database while still maintaining a certain utility is to enable a learner to learn properties of the underlying population while protecting the privacy of individuals in the database. We consider a type of privacy as defined by semantic security for cryptosystems: *"access to a statistical database should not enable one to learn anything about an individual that could not be learned without access"* [Goldwasser and Micali, 1984, Dwork, 2006].

Dwork [2006] show that this type of privacy is impossible to achieve because of auxiliary information. It can be intuitively thought of as a toy example: Assume that an individuals salary is a piece of sensitive data and suppose an adversary has the auxiliary information "Alice's salary is twice that of the average American woman". If an adversary gains access to the statistical database of average salaries by nation and gender, the adversary knows Alice's salary and privacy is thus compromised.

The fact is that privacy in this case can be compromised regardless of whether or not Alice is in the statistical database. This leads to a change in the privacy goals and to the definitions of *differential privacy*. The notion of differential privacy can be described as: *"the risk to one's privacy should not substantially increase as a result of participating in a statistical database"* [Dwork, 2006].

To grasp the notion of differential privacy we consider two databases, $\mathcal{D}$ and $\mathcal{D}'$, which differ on a single record. A differentially private algorithm will behave approximately the same for both databases, and thus the presence or absence of a single record will not effect the output by very much.

We give the definition of differential privacy introduced by Dwork et al. [2006]. We will restrict this definition to graphs, but the definition holds with slight adjustment for any type of database.

Definition 2.5 (Differential privacy)**.** A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private if for all of the possible outcomes S of $\mathcal{A}$ and for all neighboring graphs G and G' we have:

$$\Pr[\mathcal{A}(G) \in S] \leq \exp(\epsilon) \Pr[\mathcal{A}(G') \in S] + \delta \ .$$

If $\delta = 0$ we have ϵ-differential privacy. The neighboring graphs can be either *vertex* (Definition 2.2) or *edge* neighbors (Definition 2.4) resulting in a vertex or edge differentially private algorithm.

We can construct an algorithm that satisfies the inequality in Definition 2.5 and thereby obtain differential privacy. Say that we are interested in the output of some function $f : \mathcal{G} \rightarrow \mathbb{R}^k$ that operates on our data. We can construct the randomized algorithm $\mathcal{A}(G) = f(G) + \eta$, where η is a random number drawn from some distribution. If the distribution of the noise, η, is correctly chosen, $\mathcal{A}$ will be differentially private. We call this randomized algorithm a *differentially private release mechanism* for f. However, for $\mathcal{A}(G)$ to be differentially private, we need to find both the magnitude and class of distribution for η. Next, we describe how to find the magnitude of the noise.

2.2.1 The sensitivity of a function

Consider a function $f : \mathcal{G} \rightarrow \mathbb{R}^k$. We want to construct a release mechanism for f and need to find the magnitude of the noise to be added. Remember, that we want to make two neighboring graphs G and G' indistinguishable in terms of function output. Intuitively, we can consider to scale the noise according to the difference of function output of G and G'. Since there are many different combinations for which G and G' are neighbors, we should scale the noise according to the worst case, i.e. the maximum difference in function output considering all possible neighbor pairs of G and G'. We call this difference the global sensitivity of f denoted as GS_f. The concept of global sensitivity was first introduced by Dwork et al. [2006].

Definition 2.6 (Global sensitivity [Dwork et al., 2006])**.** The global sensitivity of a function $f : \mathcal{G} \rightarrow \mathbb{R}^k$, for all neighboring graphs G and G' is:

$$GS_f = \max_{G,G':d(G,G')=1} ||f(G) - f(G')||_1 \ .$$

We note that the global sensitivity is independent of any database that contains G. Instead it depends on the set to which G and every other entity in the considered database belongs to, in our case $\mathcal{G}$. For many queries, such as subgraph counting queries, the global sensitivity can be very large or unbounded. As an example, consider the set $\{G \in \mathcal{G} : |V| < n'\}$ and a triangle counting query $f_\Delta(G)$. The global sensitivity of $f_\Delta(G)$ is $GS_{f_\Delta} = n' - 2$ [Nissim et al., 2007]. In these cases, using a release mechanism scaled with the global sensitivity might result in very noisy answers. Another useful sensitivity measure is the *local sensitivity*, which captures the maximum difference in query output of a graph and all of its neighbors.

Definition 2.7 (Local sensitivity [Nissim et al., 2007])**.** The local sensitivity of a function $f : \mathcal{G} \rightarrow \mathbb{R}^k$ for a given graph G and all of its neighbors G' is:

$$LS_f(G) = \max_{G':d(G,G')=1} ||f(G) - f(G')||_1 \ .$$

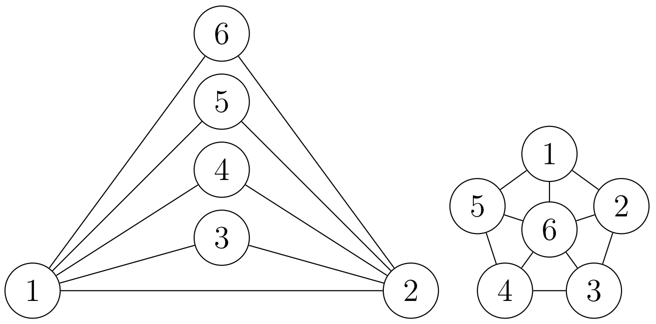


Figure 2.1: Two graphs with large local sensitivity for a triangle counting function. In the graph to the left we have large edge and node sensitivity. Consider removing either edge (1,2) or vertex 1 or vertex 2. That will destroy all of the triangles in the graph. In the right graph, we can destroy at most two triangles by the removal of one edge but all of the triangles by removing vertex 6.

We note that $GS_f = \max_G LS_f(G)$. In Figure 2.1, we see illustrate the difference of edge and vertex differential privacy. In general, the local sensitivity is much lower than the global sensitivity. It is however not possible to scale the noise in a release mechanism using the local sensitivity as it could leak information [Nissim et al., 2007]. In the next two sections, we present methods of computing lower bounds of the sensitivity.

2.2.1.1 Restricted sensitivity

Restricted sensitivity, first proposed by Blocki et al. [2013], is based on the idea that if we can make some assumption about our data, we could reduce the global sensitivity by making the right assumption. More precisely we let our assumption be the set $\mathcal{S} \subset \mathcal{G}$. We call $\mathcal{S}$ the *hypothesis space*. It is evident that the global sensitivity of a query on $\mathcal{S}$ is lower than or equal the global sensitivity on $\mathcal{G}$. By careful selection of hypothesis space we can reduce the sensitivity significantly.

Definition 2.8 (Restricted sensitivity [Blocki et al., 2013])**.** The restricted sensitivity of a function f under a hypothesis $\mathcal{S} \subset \mathcal{G}$ is:

$$RS(\mathcal{S}) = \max_{G,G' \in \mathcal{S}} \left(\frac{||f(G) - f(G')||_1}{d(G, G')} \right) .$$

We can construct a function $f_{\mathcal{S}} : \mathcal{S} \to \mathbb{R}^k$ so that $f_{\mathcal{S}}(G) = f(G)$ for every $G \in \mathcal{S}$.

Theorem 2.9 (Restricted sensitivity [Blocki et al., 2013]). *Given a function f and a hypothesis $\mathcal{S} \subset \mathcal{G}$, we can construct a function $f_{\mathcal{S}}$ such that:*

1. *$\forall G \in \mathcal{S}$: $f_{\mathcal{S}}(G) = f(G)$ and*
2. *$GS_{f_{\mathcal{S}}} = RS_f(\mathcal{S})$.*

While general, the construction of Theorem 2.9 is not very efficient. This because we only deal with a single function at a time. For any different f, we would have to construct a new $f_{\mathcal{S}}$. A more efficient way to construct $f_{\mathcal{S}}$ is to use a mapping $\mu : \mathcal{G} \to \mathcal{S}$ that maps every $G \notin \mathcal{S}$ to $\mathcal{S}$. For any $G \in \mathcal{S}, \mu(G) = G$. $f_{\mathcal{S}}$ can then be constructed as the composition of f and μ: $f_{\mathcal{S}} = f \circ \mu$.

For many functions, such as subgraph counting functions, the global sensitivity is often unbounded with respect to the graph size. This is due to the fact that in the worst case, a vertex might be connected to all other vertices of the graph, making the sensitivity proportional to graph size. For a set of graphs that have a maximum degree D, the sensitivity is instead proportional to D. Blocki et al. [2013] introduce the hypothesis space of degree bounded graphs,

$$\mathcal{G}_D = \{G = (V, E) \in \mathcal{G} :\ \forall v \in V,\ \deg(v) \leq D\}\ . \tag{2.2}$$

One way of characterize this projection is through its smoothness. We have the following definition:

Definition 2.10 (c-smoothness [Blocki et al., 2013]). A projection $\mu : \mathcal{G} \to \mathcal{S}$ is called c-smooth if for any two graphs G and G' where $d(G, G') = 1$ we have that $d(\mu(G), \mu(G')) \leq c$

We can use this projection to construct functions for the hypothesis space:

Lemma 2.11 (Composition [Blocki et al., 2013]). *Let $\mu : \mathcal{G} \to \mathcal{S}$ be a c-smooth projection. Then, for every function f, the function $f_{\mathcal{S}} = f \circ \mu$ satisfies that $GS_{f_{\mathcal{S}}} \leq c \cdot RS_f(\mathcal{S})$*

Considering edge neighbors, we also have the following claim:

Claim 2.12 (3-smooth projection [Blocki et al., 2013]). *For edge neighboring graphs, there exists an efficiently computable 3-smooth projection $\mu_D : \mathcal{G} \to \mathcal{G}_D$.*

Proof. The projection can be constructed as follows [Blocki et al., 2013]: Fix a canonical ordering of all edges and denote the edges adjacent to vertex i as ${e^{(i)}}_1, {e^{(i)}}_2, ..., {e^{(i)}}_t,$. For each edge $e = (i, j)$, we delete e if and only if: (a)

$e = e_k^{(i)}$ for $k > D$ or (b) $e = e_k^{(j)}$ for $k > D$. Let two graphs G and G' differ by one edge $e = (m, n)$, i.e. they are neighboring. W.l.o.g let e be in G. All the nodes $v \neq m, n$ in G and G' will be affected equally by the projection. There can be at most one edge e_m (adjacent to m) and one edge e_n (adjacent to n) that are deleted from G without being deleted from G'. Hence, $d(\mu(G), \mu(G')) \leq 3$. □

A naïve approach to the canonical ordering is to order the adjacent vertices according to their index. There might exist a more suitable ordering of the edges that involves e.g. fewer edges being deleted or the graphlet distribution being better preserved under the transformation. However, it is far from obvious how to construct such a ordering without the loss of generality. In Figure 2.2, we show an example of the projection and it's smoothness property. In Theorem 2.17 a differentially private release mechanism that utilizes the restricted sensitivity is described.

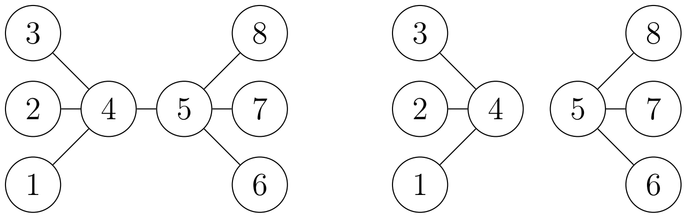


Figure 2.2: A pair of edge neighboring graphs G (left) and G' (right) that differ only in edge $(4, 5)$. Say we want to bound these graphs to a maximum degree of 3. In G, we have to remove one edge incident to both vertex 4 and vertex 5 while in G' we do not have to remove any edge at all. Say that we remove edge $(1, 4)$ and edge $(5, 6)$ from G. Then, the distance between the truncated G and G' is 3 in accordance with Claim 2.12.

2.2.1.2 Smooth sensitivity

Nissim et al. [2007] propose the notion of *smooth sensitivity*. Here, the noise is scaled according to the sensitivity of a region around each graph G instead of being scaled with the global sensitivity. A region around G can be interpreted as all graphs G' that satisfies $d(G, G') \leq s$, where s is a distance characterizing the size of the region. Since we cannot apply the local sensitivity from Definition 2.7 directly in a release mechanism [Nissim et al., 2007], we consider a smooth upper bound on $LS_f(G)$:

Definition 2.13 (Smooth bounds [Nissim et al., 2007])**.** For $\beta > 0$ the function $S : \mathcal{G} \to \mathbb{R}$ is a β-smooth upper bound on the local sensitivity if:

$$\begin{aligned} \forall G \in \mathcal{G} : \qquad & S(G) \geq LS_f(G) \\ \forall G, G', d(G, G') = 1 : \qquad & S(G) \leq \exp(\beta) S(G') \end{aligned}$$

Here, the distance metric $d(G, G')$ can refer to either $d_{vertex}(G, G')$ or $d_{edge}(G, G')$ depending on which type of privacy we seek. In Theorem 2.18, a release mechanism that utilizes these smooth bounds is presented. Nissim et al. [2007] show one way to compute such a smooth bound through the smooth sensitivity:

Definition 2.14 (Smooth sensitivity [Nissim et al., 2007])**.** For $\beta > 0$ the β-smooth sensitivity is:

$$S^*_{f,\beta}(G) = \max_{G'} \left(LS_f(G') e^{-\beta d(G,G')} \right)$$

The expression of Definition 2.14 might be difficult to compute. To simplify, we introduce the local sensitivity of the region of size s around G.

Definition 2.15 (Local sensitivity at distance s [Nissim et al., 2007])**.** The sensitivity of f at distance s is:

$$LS_f^{(s)}(G) = \max_{G' \in \mathcal{G} : d(G,G') \leq s} LS_f(G')$$

We can now express the smooth sensitivity from Definition 2.14 in terms of $LS_f^{(s)}$:

$$S^*_{f,\beta}(G) = \max_{s=1,2,\ldots,n} e^{-s\beta} LS_f^{(s)} \qquad (2.3)$$

The challenge is to find an expression for $LS_f^{(s)}(G)$ from which we can compute $S^*_{f,\beta}$. The expression will obviously depend on f and may be intractable for some f. We will also need to decide a value of β, which will depend on ϵ and must be correctly chosen for $S^*_{f,\beta}$ to be a smooth upper bound to the local sensitivity.

2.2.2 Differentially private mechanisms

In this section we present some of the proposed release mechanisms used for the release of graph statistics. First, we introduce some probability distributions needed. We denote Laplacian random numbers with mean zero, shape parameter λ and density

$$p(x) = \frac{1}{2\lambda} \exp\left(-\frac{|x|}{\lambda} \right) ,$$

as $\mathrm{Lap}(\lambda)$. We denote Cauchy random numbers with mean zero, shape parameter γ and density

$$p(x) = \frac{1}{\pi\gamma[1 + (x/\gamma)^2]} \ ,$$

as $\mathrm{Cauchy}(\gamma)$.

Using the results from Section 2.2.1 we can construct mechanisms that give differentially private outputs to queries according to Definition 2.5. We will present three mechanisms, although there exists many more. One of the most commonly used and straight forward implementations is the Laplace mechanism.

Theorem 2.16 (Laplacian Mechanism [Dwork et al., 2006])**.** *Let $f : \mathcal{G} \to \mathbb{R}^k$ be a function with global sensitivity GS_f. The randomized algorithm $\mathcal{A}(G) = f(G) + \mathrm{Lap}^k\left(\frac{GS_f}{\epsilon}\right)$ is ϵ-differentially private.*

The term $\mathrm{Lap}^k(\cdot)$ corresponds to a vector with k elements of i.i.d. Laplacian random numbers. Blocki et al. [2013] provides a release mechanism to use with the restricted sensitivity on degree bounded graphs. Based on the Laplacian mechanism, it utilizes the properties of the projection to the hypothesis space from Lemma 2.11 and Claim 2.12

Theorem 2.17 (Edge Differential Privacy w.r.t Restricted Sensitivity [Blocki et al., 2013])**.** *Let $f : \mathcal{G} \to \mathbb{R}^k$ and μ be the projection from Claim 2.12. Using the hypothesis $\mathcal{G}_D$, the randomized algorithm $\mathcal{A}(G) = f(\mu(G)) + \mathrm{Lap}^k(3RS_f(\mathcal{G}_D)/\epsilon)$ is ϵ-differentially private.*

Nissim et al. [2007] presents a release mechanism that utilizes the notion of smooth sensitivity of Definition 2.13.

Theorem 2.18 (Calibrating noise to Smooth bounds [Nissim et al., 2007])**.** *Let $f : \mathcal{G} \to \mathbb{R}$ have a β-smooth bound S on LS_f. If $\beta \leq \epsilon/\sqrt{2}$ the randomized algorithm $\mathcal{A}(G) = f(G) + \mathrm{Cauchy}\left(\frac{\sqrt{2}S(G)}{\epsilon}\right)$ is ϵ-differentially private.*

All of the above mechanisms are designed for a single query. For multiple queries, the ϵ- or (ϵ, δ)-privacy can not be guaranteed. However, we can use the following Lemma for a multiple query setting.

Lemma 2.19 (Composition and Post Processing [McSherry and Mironov, 2009, Dwork and Lei, 2009])**.** *Given a randomized algorithm $\mathcal{A}$ that runs t randomized algorithms $\mathcal{A}_i$, all of which are (ϵ, δ)-differentially private, and then applies some algorithm g on the composition of all $\mathcal{A}_i$'s, that is $\mathcal{A}(G) = g(\mathcal{A}_1(G), ..., \mathcal{A}_t(G))$. Then $\mathcal{A}$ is $(t\epsilon, t\delta)$-differentially private.*

There are two main implications of Lemma 2.19. The first one states that if we want to give answers to multiple queries at a given privacy level we must increase the levels of noise. Otherwise, it would be possible to gain private information about the noise distribution and thereby about the original data. The second implication states that we can employ any post processing technique on the output of a differentially private algorithm and still be able to provide the same privacy guarantees. The post processing techniques could include various noise reduction methods [Hay et al., 2009] and the composition could for instance include composing features into a feature vector as input to a learning algorithm.

2.3 Edge Private Release of Graph Statistics

In this section we present a number of queries for graphs and their respective sensitivities in the edge privacy model. Given these sensitivities and assuming they are not too large, there exists differentially private random algorithms for releasing their values.

2.3.1 Degree sequence

Hay et al. [2009] Provide a framework for releasing the degree sequence privately.

Claim 2.20 (Global sensitivity of the degree sequence [Hay et al., 2009]). *The global sensitivity GS_{DS} of the degree sequence in the edge differentially private setting is:*

$$GS_{DS} = 2 \ .$$

Proof. If one edge is removed from a graph with degree sequence DS then two vertices will have their degree reduced by 1. The new degree sequence DS' will either have two elements reduced by 1 or one element reduced by 2. Thus, $||DS - DS'||_1 = 2$. The case of an added edged is treated analogously. □

2.3.2 Edge count

It is evident that for a function that that counts the number of edges in a graph, that output can change at most by ± 1 by adding or removing one edge. We give the global sensitivity without the proof.

Claim 2.21 (Global sensitivity of edge counting function). *The global sensitivity of a edge counting function f_E is:*

$$GS_{f_E} = 1 \ .$$

2.3.3 Subgraph counts

There are various methods that can be employed to release counts of subgraphs. Some require very specific subgraphs such as triangles or k-stars while some do not assume any pattern. First we describe a general method, using the notion of restricted sensitivity. Remember the hypothesis of (2.2) that included all graphs with a maximum degree bound on all nodes. Under this hypothesis we get the restricted sensitivity of a subgraph counting query:

Claim 2.22 (Restricted sensitivity of subgraph counting [Blocki et al., 2013])**.** *Let $f^{(k)}$ be a subgraph counting function for some template graph $H = (E_H, V_H)$ and $k = |V_H|$. Then, $RS_{f^{(k)}}(\mathcal{G}_D) \leq kD^{k-1}$ in both the edge and vertex differentially private setup.*

Using $RS_{f^{(k)}}(\mathcal{G}_D)$, we can use the mechanism of Theorem 2.17 to release counts of subgraphs of size k under differential privacy.

2.3.3.1 Triangles

A triangle is a fully connected graph consisting of three vertices as show in Figure 2.3. Nissim et al. [2007] provide a method of releasing counts of triangles using the notion of smooth sensitivity. We let $f_\Delta(G)$ denote the number of triangles in the graph. Let the matrix B represent the number of triangles involving a potential edge: $B_{ij} = \sum_k A_{ik}A_{kj}$ and the matrix C the number of half-built triangles involving a potential edge: $C_{ij} = \sum_k A_{ik} \oplus A_{kj}$. We note that with the notation above, $LS_{f_\Delta} = \max_{(i,j)} B_{ij}$.

Claim 2.23 ([Nissim et al., 2007])**.** *The local sensitivity of f_Δ at distance s is:*

$$LS^{(s)}(G) = \max_{i \neq j; i,j \in [n]} D_{ij}(s), \text{ where}$$
$$D_{ij}(s) = \min\left(B_{ij} + \left\lfloor \frac{s + \min(s, C_{ij})}{2} \right\rfloor, n-2\right) .$$

From this expression, the smooth sensitivity can be computed from (2.3) to be used with the release mechanism of Theorem 2.18.

2.3.3.2 k-stars

A k-star is a graph in which all vertices are connected to a single vertex and not to any other vertices, show in Figure 2.4. To compute the smooth sensitivity of a k-star counting function $f_{k\star}$ we first need the local sensitivity at distance s.

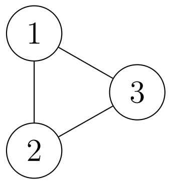


Figure 2.3: A triangle.

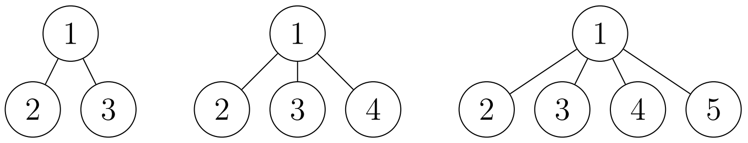


Figure 2.4: A 2-star (left), 3-star (middle) and 4-star (right).

Lemma 2.24 ([Karwa et al., 2011])**.** *Let $c(a) = \binom{a}{k-1}$, $d'_i = d_i - A_{ij}$, $b_i = n - 2 - d'_i$ and define d'_j and b_j analogously. Then the local sensitivity at distance s is:*

$$LS^{(s)}_{k\star}(G) = \max_{(i,j):d_i<d_j} LS^{(s)}_{ij}(G) \ ,$$

where $LS^{(s)}_{ij}(G)$ is the local sensitivity of $f_{k\star}$ at distance s over an edge (i,j). If $d_i \geq d_j$ then $LS^{(s)}_{ij}(G)$ is

$$\begin{cases} c(d'_i + t) + c(d'_j) & \text{if } s \leq b_i, \\ c(n-2) + c(d'_i + t - b_j) & \text{if } s \in (b_i, b_i + b_j) \\ 2c(n-2) & \text{if } s \geq b(i) + b(j) \ . \end{cases}$$

Using the results from Lemma 2.24 together with (2.3) we can compute the smooth sensitivity of $f_{k\star}$ to use with the mechanism of Theorem 2.18.

2.4 A primer on SVM and kernel methods

Classification can be considered as the categorical equivalence of regression. That is, given some input value x we want to predict the value $f(x) = y$. For regression problems y corresponds to real valued numbers and for classification y is a categorical identifier. The main task of any machine learning algorithm is to learn the function f so we can predict the value for any input value x.

We will only consider supervised learning in this thesis. In a supervised setting we have a dataset, $\mathcal{X}$, for which the labeling is known. Using the data in $\mathcal{X}$ we can learn the function f with the aim to predict categories for data from the same underlying distribution which is not present in $\mathcal{X}$.

One way of classifying data is to train a classifier that searches for a hyperplane in the feature space of the data which correctly divides the data into classes. These classifiers are commonly known as hyperplane classifiers. Training models for these classifiers include logistic regression, the perceptron and the support vector machine (SVM). In this thesis we will only consider the SVM classifier.

For the SVM framework to handle complicated data where the classes are not linearly separable in the feature space one can use *kernel methods*. Using kernel methods a SVM can learn a non linear function or hyperplane. Depending on the data we want to classify a suitable kernel function is chosen. We define a (positive semi-definite) kernel as following:

Definition 2.25 (Positive semi-definite kernel [Schölkopf and Smola, 2002])**.** Let $\mathcal{X}$ be a nonempty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a semi-definite kernel if and only if it is symmetric, that is $k(x, x') = k(x', x)$ for any $x, x' \in \mathcal{X}$, and positive semi-definite, that is

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j k(x_i, x_j) \geq 0 \ ,$$

for any $N \in \mathbb{N}$, any choice of objects $x_1, ..., x_N \in \mathcal{X}$, and any choice of real numbers $c_1, ..., c_N \in \mathbb{R}$.

This definition lets us think of kernels as inner products which convey a notion of similarity between x and x'. In the next sections we will introduce the concept of kernels for graphs. All further mentions of kernels or kernel functions will implicitly assume that these are positive semi-definite.

2.4.1 Graph classification

To classify graphs we must qualitatively be able to determine whether two graphs are similar. A widely studied topic is that of graph comparison, which we can formalize as:

Definition 2.26 (Graph comparison problem)**.** For the set of all graphs $\mathcal{G}$ and graphs $G, G' \in \mathcal{G}$, the graph comparison problem is defined as finding the function

$$k(G, G') : \mathcal{G} \times \mathcal{G} \to \mathbb{R} \ ,$$

such that k quantifies the similarity between G and G'.

A simple application of graph comparison is to investigate whether two graphs are equal or equivalently if there exists an isomorphism between G and G'. While this application is intuitive and straight forward, deciding whether there exists an isomorphism between two graphs is not. The problem of graph isomorphism is known to be in NP however it is still unknown whether or not it belongs to any of the subsets P or NP-complete [Gärtner et al., 2003].

Another graph comparison approach is to find the graph edit distance. This distance tells us the minimum amount of operations needed to transform a graph G into another graph G'. The operations might include vertex addition or deletion (Definition 2.1), edge addition or deletion (Definition 2.3) and vertex or edge relabeling. As a similarity measure, this approach is more flexible than the isomorphism which can only tell if two graphs are the same but the edit distance defines a scale of similarity. However, the problem of computing the graph edit distance between two graphs is NP-hard [Zeng et al., 2009].

Both graph isomorphism and edit distances are hard to compute. One approach to the problem of computing similarity between graphs is to find a characteristic feature representation for each graph and compute similarity of the representations. This approach is discussed further in the next section where we introduce the concept of *graph kernels*.

2.4.2 Graph kernels

In order to utilize the SVM framework for graphs we must first represent the graph by real valued features. Deciding what features to use is not obvious and is highly dependent on the specific application. The following definition formalize the problem of representing a graph by some features.

Definition 2.27 (Graph representation problem)**.** Given a graph $G \in \mathcal{G}$, the graph representation problem is defined as finding the function

$$\varphi : \mathcal{G} \to \mathbb{R}^k, \text{where } k \in \mathbb{N} \ ,$$

such that $\varphi(G)$ captures the structure of G.

We can construct a kernel according using the feature representations of Definition 2.27:

$$k(G, G') = \langle \varphi(G), \varphi(G') \rangle \ . \tag{2.4}$$

Gärtner et al. [2003] show that if φ is injective, computing k is at least as hard as deciding whether G and G' is isomorphic. Iff φ is injective the kernel of (2.4) is called a *complete* graph kernel.

To avoid this costly computation, we must rely on approximate graph representations. The challenge is to find an approximate graph representation that is still expressive enough to allow for accurate classification. In the next few sections we present a number of popular graph kernels based on various approximate representations.

2.4.2.1 Graphlet kernels

The graphlet kernels, proposed by Borgwardt [2007], Vacic [2008] uses counts of graphlets as a similarity measure between graphs. If two graphs contains graphlets that are of the same type, they are considered to be similar.

Definition 2.28 (Graphlet frequency vector)**.** The graphlet frequency vector $\mathbf{f}_G^{(k)}$ for graphlets of size k is defined as:

$$\mathbf{f}_G^{(k)} = (f_G^{(k)}(1), ..., f_G^{(k)}(a)) \ ,$$

where $f_G^{(k)}(i)$ is calculated according to (2.1) and $a = |\mathcal{H}^{(k)}|$.

Definition 2.29 (Graphlet kernel [Borgwardt, 2007, Vacic, 2008])**.** For two graphs G and G' of size $n > k$, and their corresponding frequency vectors $\mathbf{f}_G^{(k)}$ and $\mathbf{f}_{G'}^{(k)}$, the graphlet kernel is defined as:

$$k_G(G, G') = {\mathbf{f}_G^{(k)}}^\top \mathbf{f}_{G'}^{(k)} \ .$$

Usually the frequency vectors are normalized to compensate for different sizes of G and G'. Let $\mathbf{D}_G^{(k)}$ be the normalized frequency vector,

$$\mathbf{D}_G^{(k)} = \frac{\mathbf{f}_G^{(k)}}{\sum f_G^{(k)}(i)} \ . \tag{2.5}$$

A kernel using the normalized frequency vector representation can be expressed as

$$k_g(G, G') = {\mathbf{D}_G^{(k)}}^\top \mathbf{D}_G^{(k)} \ .$$

Typically, values of $k \in \{3, 4, 5\}$ are used [Borgwardt, 2007, Vacic, 2008, Shervashidze et al., 2009]. The time complexity of enumerating all graphlets is $O(nD^{k-1})$, where D is the maximum degree of the graph [Shervashidze, 2012]. The kernel can be constructed by enumerating either all of the graphlets or the connected graphlets only. In Figures 2.5, 2.6 and 2.7, graphlets of size 3, 4 and 5 are displayed.

Shervashidze et al. [2009] also considers a sampled version of the graphlet kernel. Exahustive enumeration of graphlets in large graphs is prohibitively

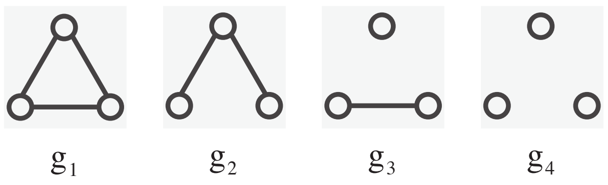


Figure 2.5: The graphlets of size three. Image gathered from Shervashidze and Borgwart. [2012].

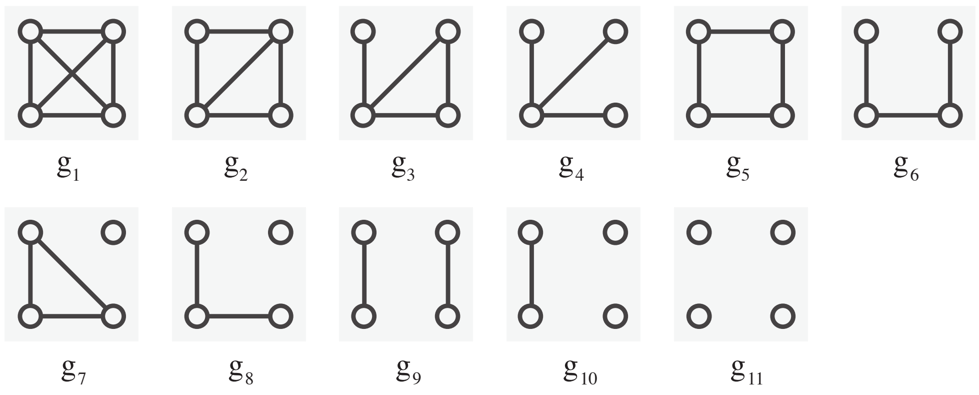


Figure 2.6: The graphlets of size four. Image gathered from Shervashidze and Borgwart. [2012]

expensive and by using a sampling algorithm one can resolve this issue. The sampling scheme proposed is to repeatedly select k vertices and identify the corresponding graphlet pattern the k vertices represent. The distribution of all graphlets, connected and disconnected, of size k can be approximated with L_1-error smaller than γ with probability $p \geq 1 - \rho$ by

$$s = \left\lceil \frac{2(\log 2 \cdot a + \log(\frac{1}{\gamma}))}{\rho^2} \right\rceil , \tag{2.6}$$

where a is the number of graphlets according to Table 2.1 and s is the sample complexity.

2.4.2.2 Random Walk kernel

The Random Walk kernel uses counts of random walks on the graphs as a measure of similarity. More specifically: given two graphs $G = (V, E)$ and $G' = (V', E')$, it performs random walks on both the graphs and count the number of similar walks. Instead of counting the number of random walks of the two graphs separately, it is possible to count the walks simultaneously by counting the number walks of the direct product graph or the Kronecker graph of G and G' [Vishwanathan et al., 2010]. The direct product graphs of the two graphs, $G_\times = (V_\times, E_\times)$ is a graph with the vertex set

$$V_\times = \{(v, v') : v \in V, v' \in V'\} ,$$

and edge set

$$E_\times = \Big\{\big((v_i, {v'}_k), (v_j, {v'}_l)\big) : (v_i, v_j) \in E \wedge ({v'}_k, {v'}_l) \in E')\Big\} .$$

An edge will appear in $G_\times$ if edges appear in both G and G' for the corresponding vertices. The adjacency matrix $A_\times$ of $G_\times$ can be expressed as:

$$A_\times = A \otimes A' ,$$

where A and A' is the adjacency matrices of G and G' respectively. $\otimes$ denotes the Kronecker product.

Definition 2.30 (Random walk kernel [Gärtner et al., 2003])**.** For two graphs G and G' with the direct product graph $G_\times = G \otimes G'$, adjacency matrix $A_\times$ of $G_\times$, vertex set $V_\times$ of $G_\times$ and weights $\lambda = \{\lambda_i\}_{i=0}^{\infty}$, where $\lambda_i \geq 0, \lambda_{i+1} > \lambda_i$ the random walk kernel is:

$$k_{rw}(G, G') = \sum_{i,j=1}^{|V_\times|} \left[\sum_{k=0}^{\infty} \lambda_k {A_\times}^k\right]_{i,j}$$

If we let $\lambda_n = \alpha^n$ for some constant α, we can rewrite the kernel in Definition 2.30, using $\lambda = \alpha$, for simplicity, to:

$$k_{rw}(G, G') = \sum_{k=0}^{\infty} (\lambda^{\frac{k}{2}} u_k)(\lambda^{\frac{k}{2}} u'_k) \tag{2.7}$$

Where, $u_k = \mathbf{e}A^k\mathbf{e}^\top$, $u'_k = \mathbf{e}A'^k\mathbf{e}^\top$ and $\mathbf{e} = (1, 1, ..., 1)^\top$. The Random Walk kernel has a runtime of $O(n^6)$ [Gärtner et al., 2003] or $O(n^3)$ [Vishwanathan et al., 2010]. Borgwardt and Kriegel [2005] argue that the performance of the kernel does not improve for large k due the increasing amount of tottering

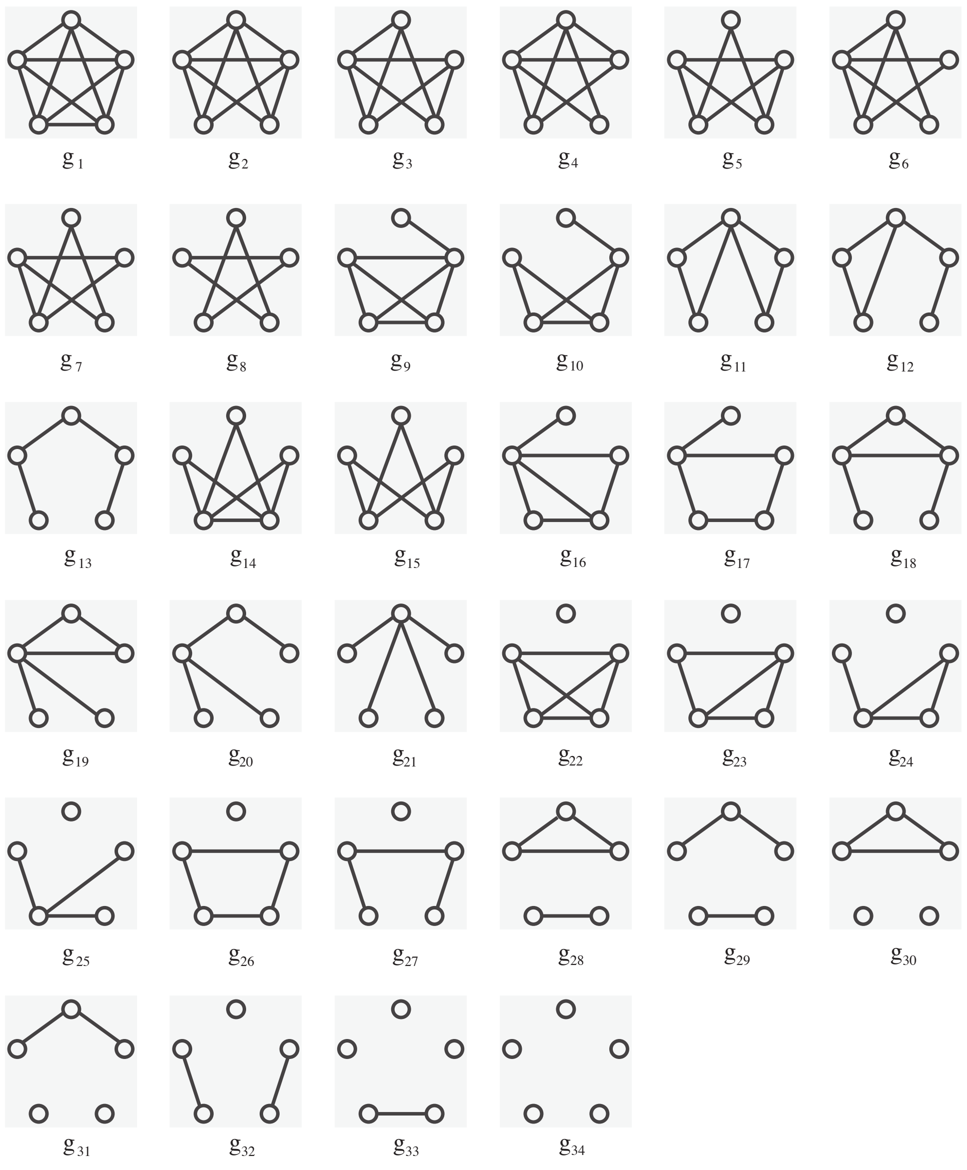


Figure 2.7: The graphlets of size five. Image gathered from Shervashidze and Borgwart. [2012]

walks that occur for larger values of k. A tottering walk is a self repeating walk, i.e. a walk that visits the same vertices multiple times. When the number of self repeating walks become large, the kernel value will be distorted, resulting in a loss of performance [Vishwanathan et al., 2010]. This deficiency can be compensated for by choosing λ_i to be very small for large i. Another solution is to consider only the first p terms of the sum as proposed by Gärtner et al. [2003].

Definition 2.31 (Truncated random walk kernel [Gärtner et al., 2003])**.** For two graphs G and G' with the direct product graph $G_\times = G \otimes G'$, adjacency matrix $A_\times$ of $G_\times$, vertex set $V_\times$ of $G_\times$ and weights $\lambda = \{\lambda_i\}_{i=0}^{p}$, where $\lambda_i \geq 0$ the truncated random walk kernel is:

$$k_{prw}(G, G') = \sum_{i,j=1}^{|V_\times|} \left[\sum_{k=0}^{p} \lambda_k {A_\times}^k \right]_{i,j}$$

This kernel has a time complexity of $O(pm)$ [Hermansson et al., 2013] which, compared to $O(n^3)$ and $O(n^6)$, allow for much faster computation, especially for sparse graphs. Just like the random walk kernel, the truncated random walk kernel can be formulated as in (2.7).

2.4.2.3 Shortest Path kernel

Next, we define the shortest path kernel [Borgwardt and Kriegel, 2005]. To do this, we must also introduce the Floyd transform. The Floyd transform transforms an input graph G to a shortest distance graph S where the label of each edge $e = (i, j)$ corresponds to the shortest path between nodes i and j. An algorithm for the Floyd transform can be found in [Borgwardt and Kriegel, 2005].

Definition 2.32 (Shortest path kernel [Borgwardt and Kriegel, 2005])**.** For two graphs G and G' with corresponding shortest path graphs S and S' the shortest path kernel is defined as:

$$k_{SP}(S, S') = \sum_{e \in E(S)} \sum_{e' \in E(S')} {k^{(1)}}_{walk}(e, e')$$

where ${k^{(1)}}_{walk}(e, e')$ is a walk kernel on paths of length 1.

The shortest path kernel has a runtime complexity of $O(n^4)$ [Borgwardt and Kriegel, 2005].

2.4.2.4 Weisfeiler-Lehman kernels

Shervashidze et al. [2011] construct a family of kernels based on the Weisfeiler Lehman test of graph isomorphism. Here, we will only consider the Weisfeiler-Lehman subtree kernel. The goal is to compare the structure of two graphs based on their node labels. Let $G = (V, E, \ell^{(0)})$ be a vertex labeled graph. That is, each vertex $v \in V$ has a corresponding label $l_v^{(0)}$ from the set $\ell^{(0)}$ denoted as $l_v^{(0)} = \ell^{(0)}(v)$. For unlabeled graphs, we can use $\ell^{(0)} = \{1\}$.

In the Weisfieler-Lehman test of isomorphism we iteratively relabel each vertex in both of the graphs and terminate either when (i) a label present in one graph but not present in the other is detected or (ii) after a given number of iterations. In the case of (i) the graphs are not isomorphic and in the case of (ii) the graphs are either isomorphic or the algorithm is unable to determine whether the graphs are not isomorphic [Shervashidze et al., 2011]. The relabeling is described below.

For a node $v \in V$ we consider the set of labels of all the adjacent vertices to v, $\mathcal{L}^{(v)} = \{(u, v) \in E : l_u^{(i)}\}$. For each node we sort its set of adjacent labels, $\mathcal{L}^{(v)}$, and concatenate the set into a single string. We set this string to be the new label of v and obtain a new set of labels, $\ell^{(i+1)}$. For instance, if a vertex v has three neighbors x, y, z with corresponding labels b, a, b, we get $\mathcal{L}^{(v)} = \{b, a, b\}$, which we sort and concatenate to $l_v^{(i+1)} = abb$. This relabeling is carried out for both graphs and for each iteration we check if there exist a label in G that does not exist in G' .

Based on this test, we can construct a kernel that counts the number of matching node labels in $\ell^{(i)}$ and $\ell^{(i)'}$ for the two graphs $G = (V, E, \ell^{(i)})$ and $G' = (V', E', \ell^{(i)'})$.

Definition 2.33 (Weisfeiler-Lehman subtree kernel [Shervashidze et al., 2011])**.** For two vertex labeled graphs $G = (V, E, \ell^{(i)})$ and $G' = (V', E', \ell^{(i)'})$, the Weisfeiler-Lehman subtree kernel at iteration i is

$$k_{WL}(G, G') = \sum_{v \in V} \sum_{v' \in V'} \delta(\ell^{(i)}(v), \ell^{(i)'}(v')) \ ,$$

where $\delta(x, y)$ is a Dirac kernel, i.e. $\delta(x, y) = 1$ if $x = y$ and 0 otherwise.

It should be noted that a Weisfieler-Lehman kernel can be constructed using any kernel that utilizes vertex labels, such as the shortest path kernel [Shervashidze et al., 2011].

Chapter 3

Private classification and sampling

In Section 3.1 we present a framework for a family of differentially private graph kernels. We use this framework to define the differentially private graphlet and p-random walk kernels in Sections 3.1.1 and 3.1.2 respectively. Lastly a novel sampling scheme for approximating the number of connected graphlets in a graph is presented in Section 3.2.

3.1 Differentially private kernels

We introduce a framework for releasing feature vectors of either subgraph counts or counts of random walks under differential privacy. Using these features, we construct kernels that by the properties of Lemma 2.19 also are differentially private. To enforce the privacy constraints this framework makes use of the Laplacian mechanism of Theorem 2.17.

The projection of graphs onto the lower degree hypothesis space is done using the projection μ_D of Claim 2.12. For graphs with vertices bounded to a lower maximum degree we have not only lower sensitivity but also a decrease in computational complexity [Shervashidze, 2012].

A description of this framework can be found in Algorithm 1. As seen in this algorithm, the noise added by the Laplacian mechanism has a shape parameter proportional to not only the restricted sensitivity of the feature $f_G(i)$ but also the number of features a.

Proposition 3.1 (Differentially private framework). *The output of Algorithm 1 is* $(\epsilon, 0)$*-differentially private.*

Proof. Apply Theorem 2.17 and Lemma 2.19 □

Algorithm 1: Edge private graph features.

```
  Input: Graphs $G = (V, E)$, truncation level $D$
  Input: Privacy level, $\epsilon$
  Input: Queries $f_G(i) : \mathcal{G} \to \mathbb{R},\ i = 1, ..., a$
1 $G_D := \mu_D(G)$;
2 $\mathbf{f}_{G_D} := [f_{G_D}(1), ..., f_{G_D}(a)]^\top$;
3 $\hat{\mathbf{f}}_G := \mathbf{f}_{G_D} + \mathbf{e},\ \mathbf{e}(i) \sim \text{Lap}\big(3a \cdot RS_{f(i)}(\mathcal{G}_D)/\epsilon\big)$;
  Output: Private counts, $\hat{\mathbf{f}}_G$
```

3.1.1 The Private Graphlet kernels

In this section, we use the framework presented above to construct a differentially private graphlet kernel of connected graphlets.

Definition 3.2 (Edge Differentially Private Connected Graphlet kernel)**.** For two graphs G, G' and a given value of D, the edge differentially private graphlet kernels are defined as the output of Algorithm 1 applied to counts of connected graphlets, $f_G^{(k)}(i)$ of (2.1), using the the restricted sensitivity of Claim 2.22,

$$k(G, G') = \hat{\mathbf{f}}_G \hat{\mathbf{f}}_{G'}^\top \ .$$

It follows naturally from Proposition 3.1 that the output of this kernel is ϵ-differentially private.

3.1.2 The Private p-Random walk kernel

Using the same reasoning as for the subgraph counting, we can find an expression for the restricted sensitivity of the counts of random walks.

Lemma 3.3 (Restricted sensitivity of Random Walks)**.** *The restricted sensitivity in the edge private model of counts of random walks of length t on graphs $G \in \mathcal{G}_D$, is:*

$$RS_{f_{RW}}(\mathcal{G}_D) = 2tD^{t-1} \ .$$

Proof. Consider two neighboring graphs $G, G' \in \mathcal{G}_D : d_{edge}(G, G') = 1$. Denote the edge the two graphs do not share by $e = (u, v)$. From vertex u to vertex v, the edge can be part of at most tD^{t-1} walks of length t. Similarly, from vertex v to vertex u, the edge can be part of at most tD^{t-1} walks. e can thus be part of at most $2tD^{t-1}$ walks. $\square$

Using the sensitivity of Lemma 3.3 and the framework of Algorithm 1, we use the feature vector representation of the p-random walk kernel to construct a differentially private p-random walk kernel.

Definition 3.4 (Edge Differentially Private p-random walk kernel)**.** For two graphs $G, G' \in \mathcal{G}_D$, the edge differentially private p-random walk kernel is defined by the output of Algorithm 1 applied to counts of random walks, with the sensitivity of Lemma 3.3,

$$k(G, G') = \sum_{n=0}^{p} (\lambda^{\frac{n}{2}} \hat{\mathbf{u}}_n)(\lambda^{\frac{n}{2}} \hat{\mathbf{u}}'_n)^\top .$$

where $\hat{\mathbf{u}}_n = \hat{\mathbf{u}}_G(n)$ and $\hat{\mathbf{u}}'_n = \hat{\mathbf{u}}_{G'}(n)$ as seen in Algorithm 1.

From Proposition 3.1, it follows that the above kernel is ϵ-differentially private.

3.2 Approximating connected graphlet counts

While efficient methods of approximating counts of all graphlets exists [Shervashidze et al., 2009], there exists to our knowledge no method that approximates counts of connected graphlets. Note that a variety of methods for sampling graphlet distributions have been proposed [Lu and Bressan, 2012, Hubler et al., 2008, Rahman et al., 2012] but they do not approximate the count of graphlets.

To approximate counts of connected graphlets, one possible solution is to use the sampling scheme proposed by Shervashidze et al. [2009] and reject all samples that are not connected. We call this method *rejection sampling.* While simple and straight forward, it is inefficient for sparse graphs as the probability of picking sets of vertices that are connected decrease with sparsity and graphlet size. Using this sampling scheme would result in a large number of rejected samples and a time complexity highly dependent on the graph sparsity.

We propose a novel sampling algorithm based on sampling of edges rather than nodes. We call this method *edge sampling* and it is listed in Algorithm 2. Let m_{H_i} be the number of edges in graphlet $H_i^{(k)}$, $m_{H_i} = |E(H_i^{(k)})|$ and s be the number of samples. We sample edges, $e_j \in E$ for $j = 1, 2, ..., s$, uniformly at random with replacement from the edge set of G. For each sampled edge we calculate all the graphlets that contains e_j and update the counts in a vector $\widetilde{\mathbf{f}}$. The total count for graphlets of type i is approximated as $\widetilde{f}_G^{(k)}(i) = \widetilde{f}(i) \frac{m}{s m_{H_i}}$.

3.2.1 Properties of the Edge Sampling Algorithm

Let Z be the number of connected graphlets of a certain type, $f_G^{(k)}(i)$ as defined in (2.1), embedded in the connected graph $G(V, E)$. We denote the

Algorithm 2: Approximate k-graphlet count, $\widetilde{\mathbf{f}}_G^{(k)}$.

```
   Input: Graph, $G = (V, E)$ with $m = |E|$
   Input: Sample size $s$
   Input: Graphlet size, $k$
1  $\widetilde{\mathbf{f}} = 0$, Sampled count for each graphlet type;
2  for $j = 1, 2, 3, ..., s$ do
3      Sample edge $e_j \in E$ uniformly at random;
4      $\widetilde{\mathbf{f}} \leftarrow \widetilde{\mathbf{f}}$ + counts of $k$-graphlets containing $e_j$;
5  end
6  $\widetilde{f}_G^{(k)}(i) \leftarrow \frac{\widetilde{f}(i)}{s} \frac{m}{m_{H_i}}$;
   Output: $\widetilde{\mathbf{f}}_G^{(k)}$ approximate $k$-graphlet counts.
```

number of graphlets that contains an edge $e \in E$ by Z_e and the set of s sampled edges as $\{e_j\}_{j=1}^s$. For each sampled edge, Z_{e_j} is the number of unique graphlets $H_i^{(k)}$ in which edge e_j is present.

For the sampled edges $\{e_j\}_{j=1}^s$ we compute

$$\hat{Z}_s = \frac{1}{s} \sum_{j=1}^{s} \frac{m}{m_{H_i}} Z_{e_j} \ . \tag{3.1}$$

Lemma 3.5. *$\hat{Z}_s$ is an unbiased estimator for the number of graphlets of size k and type i, $f_G^{(k)}(i)$, in a graph G.*

Proof. The expected value of $\hat{Z}_s$ is because of linearity of expectation

$$\mathbb{E}[\hat{Z}_s] = \frac{1}{s} \frac{m}{m_{H_i}} \sum_{j=1}^{s} \mathbb{E}[Z_{e_j}] \ . \tag{3.2}$$

Because a single graphlet will be counted m_{H_i} times and the edges are picked uniformly at random we know that

$$\mathbb{E}[Z_{e_j}] = \sum_{\text{all edges}} \Pr[\text{picking } e_j] Z_{e_j} = \frac{1}{m} \sum_{\text{all edges}} Z_{e_j} = \frac{m_{H_i}}{m} Z \ . \tag{3.3}$$

Now we combine (3.2) and (3.3) to obtain

$$\mathbb{E}[\hat{Z}_s] = \frac{1}{s} \frac{m}{m_{H_i}} \sum_{j=1}^{s} \mathbb{E}[Z_{e_j}] = \frac{m}{m_{H_i}} \mathbb{E}[Z_{e_j}] = Z \ . \tag{3.4}$$

□

We show an upper bound for the sample complexity s using Chernoff bounds. The definition is as following:

$$\Pr\left[X \geq (1+\gamma)\mathbb{E}[X]\right] \leq \exp\left(-\frac{s\mathbb{E}[X]\gamma^2}{3}\right) ,$$

which can be re-written as

$$\Pr\left[\left|X - \mathbb{E}[X]\right| \geq \gamma\mathbb{E}[X]\right] \leq 2\exp\left(-\frac{s\mathbb{E}[X]\gamma^2}{3}\right) \tag{3.5}$$

for $X = X_1 + X_2 + ... + X_n$ with $X_i \in [0,1]$ $\forall i$ and $\gamma \geq 0$.

Consider a transformation of the random variable Z_{e_i} by:

$$X_i = \frac{m}{m_{H_i}} \cdot Z_{e_i} ,$$

then we get $X_i \in [0, C]$ where $C = \frac{m}{m_{H_i}} \max_e Z_e$. We can now rewrite (3.1) in terms of X as:

$$\hat{Z}_s = \frac{1}{s}\sum_{i=1}^{s} X_i .$$

The corresponding Chernoff bounds according to (3.5) becomes:

$$\Pr\left[\left|\hat{Z}_s - \mathbb{E}[\hat{Z}_s]\right| \geq \gamma\mathbb{E}[\hat{Z}_s]\right] \leq 2\exp\left(-\frac{s\mathbb{E}[\hat{Z}_s]\gamma^2}{3C}\right) = \rho ,$$

which gives us a sample complexity of

$$s = \frac{3C\log(2/\rho)}{\mathbb{E}[\hat{Z}_s]\gamma^2} .$$

We note that $\mathbb{E}[\hat{Z}_s] = \frac{m}{m_{H_i}}\mathbb{E}[Z_{e_i}]$ from (3.4) and we can now give more descriptive formulation of the sample complexity as

$$s = 3\frac{\max_{e_i} Z_{e_i}}{\mathbb{E}[Z_{e_i}]}\frac{\log(2/\rho)}{\gamma^2} .$$

Introducing the notation

$$s = 3\alpha_i^{(k)}\log(2/\rho)/\gamma^2 , \tag{3.6}$$

we get sample complexity proportional to

$$\alpha_i^{(k)} = \frac{\max_{e_j} Z_{e_j}}{\mathbb{E}[Z_{e_j}]} \leq \frac{Z}{\frac{m_{H_i}}{m}Z} = \frac{m}{m_{H_i}} . \tag{3.7}$$

This $\alpha_i^{(k)} = m/m_{H_i}$ is the worst case, it represents a situation where one edge is contained in all connected graphlets of the graph. Note also that the i and k correspond to a type and size of graphlet. The worst case sample complexity for any size of graphlet corresponds to $\max_i \alpha_i^{(k)}$, i.e. the graphlet type with the least number of edges. As we will see in the results often a much lower value for α than m/m_{H_i} can be empirically determined.

For each sample, given an edge e, we can compute the number of graphlets containing this edge in $O(D^{k-2})$ [Shervashidze, 2012]. This gives an overall complexity of $O(sD^{k-2})$ for the approximation algorithm. We also note that $\max \alpha \sim m \leq nD$ and we can rewrite the complexity as $O(nD^{k-1})$ which is equivalent to exact enumeration of all graphlets.

We have now shown that the proposed sampling scheme of Algorithm 2 is an unbiased estimator for $f_G^{(k)}(i)$ with a worst case sample complexity equal to that of exact enumeration. However, in practice a significantly smaller amount of samples are needed to obtain accurate approximations of the graphlet counts.

Chapter 4

Experiments

In this chapter, we evaluate the different methods that together enable private classification. We perform this evaluation on various datasets, described in Section 4.1. In Section 4.2 we evaluate the effects of bounding the degree using the projection scheme μ_D. We evaluate our novel sampling method in detail in Section 4.3, where we first compare our sampling method to an other method and secondly, evaluate the effects of sampling on our datasets. We compare the private classification with the non private classification and evaluate the performance of the sampled kernels in Section 4.4.

4.1 Data sets

Many of the datasets commonly used for evaluation of graph classification [Shervashidze et al., 2011] are not suitable for the scope of this thesis. The graphs represented in these datasets are too small, corresponding to feature vectors having too low values compared to the noise magnitude. Because of this three datasets that consists of large sparse graphs were compiled for the purpose of this thesis. The datasets are called PROTO, ROADS and SOCIAL.

PROTO is a set of connected induced subgraphs sampled using random vertex expansion from synthesized population interaction networks[1] of either Portland or Montgomery County [Swarup et al., 2014]. ROADS is a set of connected induced subgraphs sampled using a MCMC scheme [Hubler et al., 2008], from road network graphs of Texas and California available from SNAP[2]. The class of each graph corresponds to the county or state representing the network from which it has been sampled. SOCIAL is a collection

[1]http://www.vbi.vt.edu/ndssl

[2]http://snap.stanford.edu/

Table 4.1: Statistics of datasets. N corresponds to the number of graphs, n the number of nodes, m number of edges, $\alpha_*^{(k)} = \max_i \alpha_i^{(k)}$ with $\alpha_i^{(k)}$ as in (3.7) and d the average or maximum degree of the graphs. ‡ correpsponds to computations that did not finish within 2 days.

	D & D	PROTO	ROADS	SOCIAL
N	1178	200	200	262
Pos./Neg.	691/487	100/100	100/100	131/131
$\alpha_*^{(3)}$	4.3	14.9	33.9	‡
$\alpha_*^{(4)}$	50.6	1689	2291	‡
n_{max}	5748	1000	10000	4938
n_{avg}	284.3	1000	10000	1072.2
m_{max}	14267	10308	13973	1473709
m_{avg}	715.7	4321.4	13283.7	104026.1
d_{max}	19	176	12	2971
d_{avg}	5.5	9.5	2.7	80.6

of graphs from social networking sites Twitter and Google+ available from SNAP[2]. We divide the set into two classes of equal size: Google+ graphs in one and Twitter graphs in the other. We use only the largest (by number of nodes) Twitter networks to obtain equally large classes, we also remove the ego-nodes of the networks and direction of edges. In addition to these datasets, we also include one of the benchmark datasets, D & D, commonly used for classification. We consider D & D to represent a lower boundary with respect to graph size. D & D consists of graphs representing proteins classified according to whether they are enzymes or not. A set of statistics for the datasets mentioned above is presented in Table 4.1.

4.2 Degree bounding

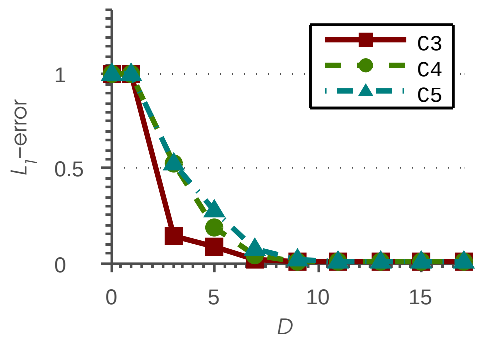
We evaluate the effects of the bounding of degrees on our datasets by computing the L_1-error between the graphlet distributions of original and degree bounded graphs. We define the L_1-error as

$$L_1\text{-error} = \left\langle ||\mathbf{D}_G^{(k)} - \mathbf{D}_{G_D}^{(k)}||_1 \right\rangle_G \ , \ G_D = \mu_D(G) \ , \tag{4.1}$$

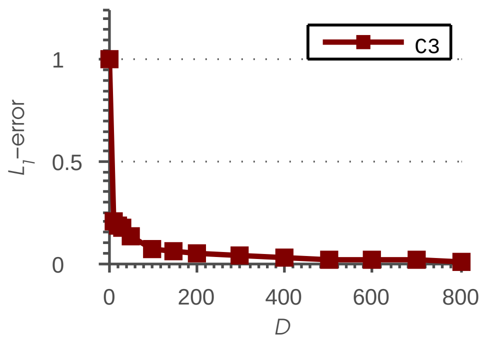
where $\mathbf{D}_G^{(k)}$ corresponds to the normalized graphlet distribution of graph G from (2.5) and $\langle\cdot\rangle_i$ the mean with respect to i.

We compute the L_1-error for different values of $D < d_{\text{max}}$ and for the distributions of size k graphlets for $k \in \{3, 4, 5\}$. We use the naïve approach described in Section 2.2.1.1 for the projection μ_D. The results are presented in Figure 4.1. For PROTO and SOCIAL the computation of the true graphlet counts for $k = 5$ and $k = 4, 5$, respectively, exceeded two days and the results are not available.

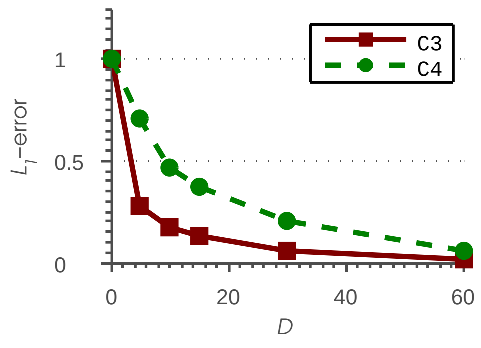
For $D = 0$ and $D = 1$, we see that the L_1-error $= 1$, due to the fact that no connected subgraphs of size $k > 2$ can exist if the maximum degree of the graph is $D \leq 1$. We note that for many of the data sets, the L_1-error drops to a low value (> 0.1) for a relatively low value of D compared to $d_{\max}$. Intuitively, we can think of these values of D as a first choice when selecting the bounding degree for the private kernels. The final value is determined with respect to classification accuracy.



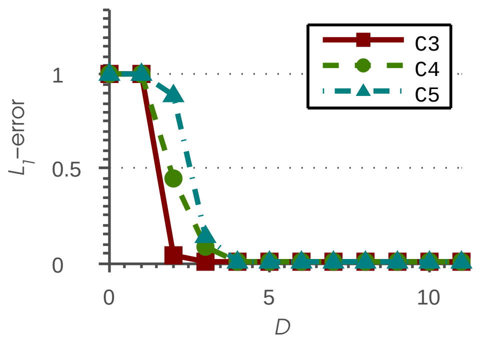
(a) L_1-error for the D & D data set



(b) L_1-error for the SOCIAL data set



(c) L_1-error for the PROTO data set



(d) L_1-error for the ROADS data set

Figure 4.1: L_1-error calculated according to (4.1) for the datasets D & D, ROADS, PROTO and SOCIAL. Distributions for $k \in \{3, 4, 5\}$

4.3 Sampling Connected Graphlets

We evaluate our novel sampling algorithm of Section 3.2 in two steps. First, we evaluate the effects of graph sparsity on the performance of the sampling scheme in comparison to the rejection sampling. Secondly, we empirically evaluate the sample complexity of the edge sampling scheme.

The effects of sparsity on sampling performance are evaluated by sampling from a set of random Erdős-Rényi graphs with incresing densities. The density is defined as

$$c = \frac{m}{n(n-1)/2} \ ,$$

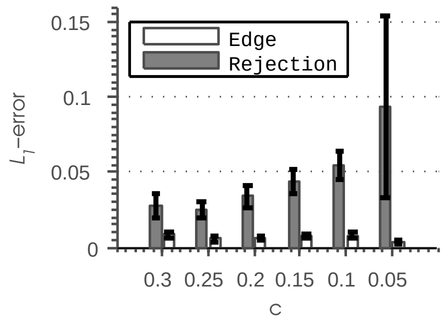
and we consider graphs with densities in the set $c \in \{0.05, 0.1, 0.15, ..., 0.3\}$. For each density 10 random graphs are generated, and for each graph we approximate the graphlet counts by sampling with two different sampling methods. We sample 100 edges using the edge sampling scheme from each graph and determine the number of graphlets encountered. For the rejection sampling we use this number as the number of samples to take for the corresponding graph. We measure the performance in terms of average L_1-errors of the sampled and true graphlet distributions. We define

$$L_1\text{-error} = \left\langle ||\mathbf{D}_G^{(k)} - \widetilde{\mathbf{D}}_G^{(k)}||_1 \right\rangle_G \ , \tag{4.2}$$

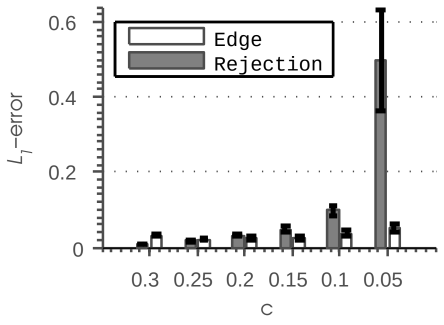
where $\mathbf{D}_G^{(k)}$ and $\widetilde{\mathbf{D}}_G^{(k)}$ are the normalized graphlet distributions of true and sampled counts respectively. The results for sampling of 3- and 4-graphlets are found in Figure 4.2. Notably, the rejection sampling only achieves low L_1-error for graphs that are dense while the novel algorithm achieves low L_1-error regardless of density. We observe that for high densities and large graphlet sizes the rejection sampling outperforms the edge sampling.

To evaluate the edge sampling scheme for the datasets presented in Section 4.1 we compute the L_1-error defined in (4.2), for $s_i = 10, 20, ..., 100$ samples for each dataset and $k \in \{3, 4, 5\}$. The true graphlet counts for PROTO and SOCIAL are unavailable for $k = 4$ and $k = 4, 5$ respectively due to the exhaustive enumeration becoming prohibitively expensive.

The results are found in Figure 4.3. We see that the number of samples actually needed for the L_1-error to become small, is typically much lower than the worst case sample complexity defined in (3.7). We observe that the number of counted graphlets needed to reach a low L_1-error are of the same order as proposed by Shervashidze et al. [2009].



(a) L_1-error of sampling 3-graphlets for Erdős-Rényi graphs.



(b) L_1-error of sampling 4-graphlets for Erdős-Rényi graphlets.

Figure 4.2: L_1-error relative to density of graphs. We sampled 100 edges with the edge sampling scheme and the corresponding amount of graphlets for the rejection sampling. The L_1-error is given with the standard error. Note the different scales of L_1-error.

4.4 Classification

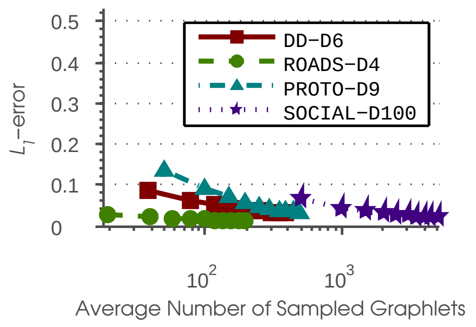
In this section we present the classification results. We present the results considering two aspects, privacy and sampling. First, we provide results for the baselines and the private kernels. We compare the accuracy of the private kernels to various baselines. Secondly, we provide classification results for the sampled of the connected kernels. We compare the accuracy of the sampled kernels to the accuracy of the baseline kernels which we refer to as the *full* kernels. The results are presented in Tables 4.2 and 4.3.

Setup

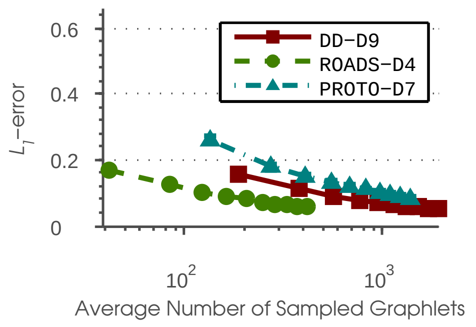
We use a binary C-SVM with 10-fold cross validation. For each fold, we optimize C with respect to classification accuracy. We repeat this experiment 10 times and report the average accuracy and standard error.

As baselines, we use the p-random walk (PRW) kernel [Gärtner et al., 2003], the Weisfeiler-Lehman (WL) kernel [Shervashidze et al., 2011] and the graphlet (GK-Ck or GK-Ak, $k \in \{3, 4, 5\}$) kernels [Shervashidze et al., 2009]. We use the two private kernels from Section 3.1, denoted as DPPRW and DPGK-C4, $k \in \{3, 4\}$. We use the unnormalized graphlet counts in all of the graphlets kernels as we note that it yields better results in general.

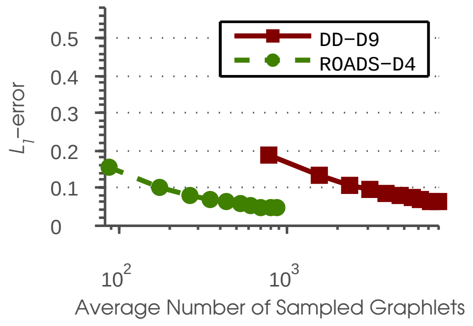
For the PRW kernel, we choose λ from the set $\{1, 0.1, 0.01, 0.001\}$ with respect to classification accuracy. For D & D, SOCIAL and ROADS, $\lambda = 0.1$



(a) L_1-error of sampling 3-graphlets for D & D, ROADS, PROTO and SOCIAL.



(b) L_1-error of sampling 4-graphlets for D & D, ROADS and PROTO.



(c) L_1-error of sampling 5-graphlets for D & D and ROADS.

Figure 4.3: L_1-error relative to number of samples taken using the edge sampling algorithm. Sample sizes $s = 10, 20, 30, \ldots 100$ were used in all plots.

gave the best results. For PROTO, $\lambda = 0.01$ gave the best results.

We select D with respect to the classification accuracy of the private kernels. We chose the following values, in the order of D & D, PROTO, ROADS and SOCIAL; for PRW and DPPRW: $(9, 9, 4, 50)$, for DPGK-C3 and GK-C3: $(6, 9, 4, 100)$, for DPGK-C4 and GK-C4: $(9, 7, 4, 100)$.

For the experiments of Table 4.3, we use the edge sampling described in Section 3.2 to compute the sampled verison of GK-Ck, denoted as GK-Sk. We select the number of samples based on the empirical results of Section 4.3, see Figure 4.3. We selected the number samples, s, from the set $s \in \{50, 100, 200\}$. For the computation of the GK-Ak kernels we sample according to the sampling of Shervashidze et al. [2009] with the number of samples calculated according to (2.6) with $\gamma = \rho = 0.1$.

Results

From Table 4.2 it is obvious that only WL and GK-C5 are able to achive classification accuracy higher than label distribution for ROADS, for all other datasets we can classify better than the label distributions with at least one of the private kernels. The DPGK-C3 kernel perform increasingly better with respect to graph size, from D & D to PROTO to SOCIAL. DPPRW do not show this size dependency, however. On all applicable datasets, DPGK-C4 fail to classify better than the label distributions. This is likely caused by the very large amounts of noise added to the features due to both graphlet size and composition.

For the PROTO dataset the accuracy of DPPRW compared to PRW actually increases. Although this might seem counterintuitive, we note that the accuracy of PRW for the truncated graphs is significantly higher than its baseline counterpart. This explains the increase in accuracy of DPPRW as in fact the comparison between the baselines and their private counterparts is more fair when the kernels are computed on identical data, i.e. the truncated graphs. There is also a similar increase in accuracy between the GK-C3 kernel for PROTO when computed with truncated graphs. In general, we note that the truncation alone makes little difference to the accuracy.

In Table 4.3 we note that in most cases, the performance of the sampled connected kernels are close to the performance of the full kernels. Notable exceptions is the accuracy of GK-S5 for D & D and ROADS where we for D & D observe a small but significant drop of accuracy and for ROADS are unable to classify better than the label distribution for any of the sample sizes.

The sample sizes used are significantly smaller than the sample complexity of (3.6). Even for these relatively small sample sizes, we are able to classify

Table 4.2: Classification accuracy. We give the accuracy and the standard error. The first part of the table represents the baselines, the second part the classification of the truncated graphs and the third part the classification using the private kernels. ‡ denotes computations that took longer than two days. † denotes classification using private kernels with $\epsilon = 0.5$.

Kernel	D & D	PROTO	ROADS	SOCIAL
PRW	75.35 ± 0.6	83.75 ± 1.2	55.45 ± 3.7	82.96 ± 0.4
WL	74.88 ± 0.6	93.70 ± 5.1	90.30 ± 1.2	79.81 ± 1.8
GK-C3	74.42 ± 1.0	98.35 ± 1.1	44.35 ± 2.3	88.96 ± 0.7
GK-C4	73.31 ± 1.0	99.90 ± 0.2	55.00 ± 2.3	‡
GK-C5	74.11 ± 0.7	‡	86.25 ± 1.9	‡
PRW	75.23 ± 0.9	89.00 ± 1.1	43.55 ± 3.9	82.96 ± 0.4
GK-C3	74.21 ± 1.3	97.75 ± 2.9	42.40 ± 3.2	88.65 ± 0.7
GK-C4	73.80 ± 0.7	99.50 ± 0.0	54.40 ± 2.2	‡
GK-C5	73.58 ± 1.2	‡	85.55 ± 2.2	‡
DPPRW†	68.44 ± 1.1	86.85 ± 3.0	48.95 ± 2.8	68.88 ± 1.9
DPGK-C3†	59.26 ± 0.5	73.30 ± 1.6	45.20 ± 2.5	76.96 ± 0.5
DPGK-C4†	58.57 ± 0.1	50.95 ± 3.1	44.65 ± 2.0	‡

with accuracy close to the baseline.

For the sampled kernel that computed all graphlets, GK-Ak, we observe that as the density of the graphs increases, from ROADS to PROTO to SOCIAL, the kernels performs better. For the most sparse dataset, ROADS, the kernels cannont classify better than the label distribution. For D & D, the kernels perform close to the baseline accuracy of their connected counterparts.

Table 4.3: Classification accuracy. We give the accuracy and the standard error. s is the number of samples taken corresponding to edges for GK-Sk and graphlets for GK-Ak. ‡ denotes computations that took longer than two days.

Kernel	D & D	PROTO	ROADS	SOCIAL	s
GK-C3	74.42 ± 1.0	98.35 ± 1.1	44.35 ± 2.3	88.96 ± 0.7	-
GK-S3	74.06 ± 0.8	96.75 ± 1.1	49.50 ± 4.5	87.35 ± 1.0	50
GK-S3	74.99 ± 0.8	97.65 ± 1.6	45.10 ± 1.2	86.96 ± 0.4	100
GK-S3	74.19 ± 0.7	96.80 ± 1.3	48.55 ± 3.5	87.31 ± 0.7	200
GK-C4	73.31 ± 1.0	99.90 ± 0.2	55.00 ± 2.3	‡	-
GK-S4	74.37 ± 0.7	98.90 ± 0.7	44.25 ± 3.0	‡	50
GK-S4	73.13 ± 1.2	99.25 ± 0.9	51.90 ± 4.0	‡	100
GK-S4	73.79 ± 0.8	99.70 ± 0.3	45.95 ± 3.9	‡	200
GK-C5	74.11 ± 0.7	‡	86.25 ± 1.9	‡	-
GK-S5	72.12 ± 1.2	98.85 ± 0.3	46.65 ± 2.6	‡	50
GK-S5	72.25 ± 1.0	99.95 ± 0.2	50.15 ± 3.2	‡	100
GK-S5	72.80 ± 1.2	99.20 ± 0.3	53.25 ± 3.7	‡	200
GK-A3	74.36 ± 0.4	73.95 ± 1.1	40.65 ± 2.6	71.81 ± 1.7	1016
GK-A4	74.68 ± 0.5	83.75 ± 0.9	51.20 ± 2.9	76.23 ± 2.0	1986
GK-A5	74.61 ± 0.5	85.00 ± 1.9	42.00 ± 2.6	81.50 ± 1.8	5174

Chapter 5

Discussion

We discuss the results of the experiments presented in Chapter 4, each of the experiments and their results are discussed in Sections 5.1, 5.2 and 5.3. In Section 5.4 we present some criticism of differential privacy with respect to privacy of the data generating process. In Section 5.5 we discuss the general implications of the thesis and novel kernels and sampling scheme. Lastly, we comment on future work and what we think is the next step in this line of research in Section 5.6.

5.1 Degree bounding, edge truncation

While simple and unrefined, the naïve projection μ_D preserves the distributions relatively well, as seen in Figure 4.1. There might exist projections that removes fewer edges and better preserves the distributions, but it is not obvious how to construct such a projection that is independent of graph structure.

For our datasets, the degree bounding affects the classification accuracy only marginally (see Table 4.2). We observe this behavior even for relatively small values of the bounding degree, D. The separating features of the graphs can therefore not be highly dependent on the high degree vertices of the graphs. For graphs generated by e.g. the preferential attachment model [Barabási and Albert, 1999], this might not be true as these graphs to a large extent are characterized by their large degree vertices or hubs. For private analysis of these types of networks, sensitivity measures other than the restricted sensitivity of degree bounded graphs might be more suitable.

5.2 Sample complexity and quality of approximation

Figure 4.2 show that the approximation of edge sampling achieves a low L_1-error for all the considered densities. Most notably it outperforms the rejection sampling for sparse graphs. The results also show that rejection sampling improves as the graphs grow more dense and even beats the edge sampling for certain densities. An adaptive choice of sampling algorithm based on the density of graphs would yield the lowest L_1-errors.

From Figure 4.3 it is clear that by sampling 100 edges, the edge sampling approximation gives graphlet counts with L_1-errors lower than 0.1 for graphlets of all considered sizes. For example, consider a graph with $m = 10000$ edges, the worst case sample complexity can be calculated from (3.6) with $\alpha = m/m_{H_i}$. The worst case sample complexity for $k \in \{3, 4, 5\}$ is of the order 10^6 edges, but as we can see in the results only sampling 10^2 edges gives good approximations.

We can see from Table 4.1 that the actual values of α implies much lower worst case sample complexities. However, the results from Figure 4.3 imply even lower complexity. By computing the average number of sampled graphlets we can compare edge sampling to the rejection sampling. Shervashidze et al. [2009] presents sampling levels in the range $[1016, 21251]$, which are the same orders of magnitude as presented in Figure 4.3.

5.3 Classification

By selecting D to maximize the classification accuracy we naturally capture the trade-off between retaining as much information as possible versus keeping the noise level as low as possible. We observe that the optimal D is close to the value for which the L_1-error drops and approaches zero.

For some datasets, the private kernels fail to perform better than the label distribution of the datasets. This corresponds to the features being randomly scattered in the feature space with no possible separating hyperplane. Intuitively we can think of the signal-to-noise ratio (SNR) of the features as a measure that quantifies this behavior. When the SNR is low, we cannot separate the classes of the datasets due to the high amounts of noise. When the SNR is high, the amounts of noise is relatively low and the separating hyperplane will be similar to the separating hyperplane of the non-private kernel. Since the shape parameter of the noise is constant, the only way to achieve high values of the SNR is through large feature values which require graphs that are either dense or large. From this, a natural size constraint arises which states that we cannot classify graphs with differential privacy if the graphs are too small or contains too few edges. We observe this behavior for the graphlet kernels which performs closer and closer to the baseline as the sizes of the graphs increases. We do not observe this lower bound for the p-random walk kernel which performs relatively well on even the smallest graphs, in the D & D set.

For sparse graphs the graphlet rejection sampling does not perform well. Due to the sparsity, the probability of sampling connected components of these graphs becomes very small. It is therefore likely that the feature vector will be dominated by the counts of the empty graphlet, resulting in a close to inseparable set.

Such is the case for ROADS where GK-Ak kernels are unable to separate the set. PROTO contains graphs that are denser than ROADS and GK-Ak kernels are able to capture a larger part of the separating structure of the classes. The dataset is, however, still relatively sparse and compared to GK-Ck kernels, the GK-Ak kernels perform poorly. SOCIAL is denser and GK-Ak kernels are closer in performance to GK-Ck kernels.

5.4 Criticism of Differential Privacy

While the Laplacian mechanism is independent of the data, some queries are not. Such queries can be related to the data generating process as described by Kifer and Machanavajjhala [2011].

Consider a social network with two distinct communities; If there exists a single edge between the two communities, the probability of more cross-community edges to form as time evolves is increased in comparison to the case where no cross-community edges are present from the start. The initial edge between the communities impacts the data generating process.

Kifer and Machanavajjhala [2011] considers a situation where an adversary knows about the existence of two communities along with the fact that ”if there exists an edge between the communities it is between Bob (who is a member of the first community) and someone in the second community”. If the adversary learns about Bob’s edge, privacy is considered to be compromised. Kifer and Machanavajjhala [2011] note that the sensitivity of this query and thus the amount of noise to add is unknown unless assumptions on the time evolution of the network are made.

This highlights a complication for differential privacy: certain queries and certain statistics may have complicated or indeterminable sensitivity. As noted by Kifer and Machanavajjhala [2011], privacy is only preserved if the sensitivity (and thus the magnitude of the noise) can be determined accurately. Thus the issue can be formulated as: *differential privacy does not take the data generating process into account.*

If the privacy formulation is such that all effects of a certain node or edge in the data generating process must be private then noise must be scaled to this notion of time dependent sensitivity. An equivalent formulation is that queries on the databases generated with or without this node or edge should result in indistinguishable results.

Even though the sensitivity may be easily found, it may induce sufficiently large noise to drown out the signal i.e. a signal to noise ratio such that answers to queries are rendered useless for all practical purposes.

5.5 Concluding remarks

By using the framework of Section 3.1 we can reformulate any graph kernel that can be represented explicitly by feature vectors as a private graph kernel. However, the sensitivity of a given feature may be unbounded and thus the corresponding private feature may be useless w.r.t. classification accuracy.

The implicit trade-off for private graph kernels lies in the compromise between expressiveness of the features versus their sensitivity. More expressive features imply a more accurate kernel but these features are also more sensitive and will thus result in larger amounts of noise being added by private mechanisms.

Furthermore, the choice of kernel will naturally impact the accuracy and performance of classification. As privacy through noise addition always worsen the classification accuracy, the kernel must provide adequate accuracy already before noise addition. Differential privacy will not improve a kernel but rather learn from distorted training data and make classification less accurate.

As expected the private kernels are not as accurate as their non-private counterparts. The DPPRW and DPGK-C3 are the top performers and beats the label distribution for all datasets except ROADS. We can also see that the sampled GK-Sk kernels perform well. There is a slight drop in classification accuracy early on but as the number of samples increase this becomes negligible.

5.6 Future work

As noted in the introduction and problem formulation, this thesis is to the best of our knowledge the first research on differentially private graph kernels.

There are many unexplored topics and the possible applications become more and more frequent as the social networks and other online services increase in popularity.

In this paper we only consider the private versions of the p-random walk kernel and graphlet kernels. As noted in 3.1, every kernel that allows a explicit feature representation could potentially be reformulated as a private kernel by making use of Algorithm 1. One future development would be to consider other graph kernels and apply the differentially private framework.

The concept of differential privacy is only considered in the notion of edge privacy. Node differential privacy enforces stronger guarantees on privacy than edge edge differential privacy [Kasiviswanathan et al., 2013] and is an interesting possible extension to this thesis.

There is also a need for larger real world datasets. As mentioned in Section 4.1, the benchmark datasets often used for graph classification are not large enough for reasonable signal to noise ratios. An agreed upon battery of large graph benchmark datasets is needed to compare the accuracy and utility of different kernels.

Bibliography

A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):pp. 509–512, 1999.

J. Blocki, A. Blum, A. Datta, and O. Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *ITCS*, pages 87–96, 2013.

B. Bollobás. *Modern graph theory*, volume 184. Springer, New York, 1998.

K. Borgwardt. *Graph Kernels.* PhD thesis, Ludwig-Maximilians-University Munich, 2007.

K. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *Data Mining, Fifth IEEE International Conference on*, Nov 2005.

C. Dwork. Differential privacy. In *Automata, Languages and Programming*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2006.

C. Dwork and J. Lei. Differential privacy and robust statistics. In *Proceedings of the 41th Annual ACM Symposium on Theory of Computing (STOC)*, Bethesda, Maryland, May 2009. Association for Computing Machinery, Inc.

C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pages 265–284, 2006.

S. Goldwasser and S. Micali. Probabilistic Encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

T. Gärtner, P Flach, and S Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Conference on learning theory*, pages 129–143, 2003.

M. Hay, Chao Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *Data Mining, 2009. ICDM '09. Ninth IEEE International Conference on*, pages 169–178, Dec 2009.

L. Hermansson, T. Kerola, F. Johansson, V. Jethava, and D. Dubhashi. Entity disambiguation in anonymized graphs using graph kernels. In *Proc. of the 22nd ACM International Conference on Information and Knowledge Management (CIKM)*, 2013.

C. Hubler, H.-P. Kriegel, K. Borgwardt, and Z. Ghahramani. Metropolis algorithms for representative subgraph sampling. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, pages 283–292, Dec 2008.

V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev. Private analysis of graph structure. *PVLDB*, 4(11):1146–1157, 2011.

S. P. Kasiviswanathan, K. Nissim, S Raskhodnikova, and A. Smith. Analyzing graphs with node-level differential privacy. In A. Sahai, editor, *Theory of Cryptography*, pages 457–476, 2013.

D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 193–204. ACM, 2011.

X. Lu and S. Bressan. Sampling connected induced subgraphs uniformly at random. In *Proceedings of the 24th International Conference on Scientific and Statistical Database Management*, SSDBM'12, pages 195–212, 2012.

F. McSherry and I. Mironov. Differentially private recommender systems: Building privacy into the net. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 627–636, 2009.

A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. *2013 IEEE Symposium on Security and Privacy*, 0:111–125, 2008.

K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 75–84, 2007.

M. Rahman, M. Bhuiyan, and M. Al Hasan. Graft: An approximate graphlet counting algorithm for large graph analysis. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1467–1471, 2012.

B. Schölkopf and A. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT Press, 2002.

N. Shervashidze. *Scalable Graph Kernels.* PhD thesis, University of Tübingen, 2012.

N. Shervashidze and K. Borgwart. Graph kernels: Code and data, 2012. URL `\webdav.tuebingen.mpg.de/u/karsten/Forschung/research.html?\page=research&topic=JMLR10_graphkernels&html=JMLR10`. Collected 06-12-2014.

N. Shervashidze, S. V. N. Vishwanathan, T. H. Petri, K. Mehlhorn, and K. Borgwart. Efficient graphlet kernels for large graph comparison. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5, pages 488–495, 2009.

N. Shervashidze, P. Schweitzer, E. van Leeuwen, K. Mehlhorn, and K. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.

S. Swarup, S. G. Eubank, and M. V. Marathe. Computational epidemiology as a challenge domain for multiagent systems. In *AAMAS '14*, pages 1173–1176, Richland, SC, 2014.

L. Sweeney. Matching known patients to health records in washington state data. *CoRR*, abs/1307.1370, 2013.

V. Vacic. *Computational methods for discovery of cellular regulatory mechanisms.* PhD thesis, University of California Riverside, 2008.

S. V. N. Vishwanathan, N. Schraudolph, R. Kondor, and K. Borgwardt. Graph kernels. *J. Mach. Learn. Res.*, 11:1201–1242, aug 2010. ISSN 1532-4435.

Z. Zeng, A. Tung, J. Wang, J. Feng, and L. Zhou. Comparing stars: On approximating graph edit distance. *Proc. VLDB Endow.*, 2(1):25–36, aug 2009.